

# Développement avec VMware vRealize Orchestrator

vRealize Orchestrator 7.6



vmware®

Vous trouverez la documentation technique la plus récente sur le site Web de VMware, à l'adresse :

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

Si vous avez des commentaires à propos de cette documentation, envoyez-les à l'adresse suivante :

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

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

**VMware France SAS.**  
Tour Franklin  
100-101 Terrasse Boieldieu  
92042 Paris La Défense 8 Cedex  
France  
[www.vmware.com/fr](http://www.vmware.com/fr)

Copyright © 2008-2019 VMware, Inc. Tous droits réservés. [Informations relatives aux copyrights et marques commerciales.](#)

# Table des matières

Développement avec VMware vRealize Orchestrator 10

## 1 Développement de workflows 11

Concepts clés des workflows	14
Paramètres du workflow	14
Attributs de workflow	14
Schéma de workflow	15
Présentation d'un workflow	15
Jetons de workflow	15
Phases du processus de développement de workflow	15
Meilleures pratiques concernant le développement de workflows	16
Droits d'accès au client Orchestrator	16
Tester des workflows au cours du développement	16
Création et modification d'un workflow	17
Créer un workflow	17
Modifier un workflow	18
Modifier un workflow à partir de la bibliothèque standard	18
Onglets de l'éditeur de workflow	19
Fourniture des informations générales sur le workflow	20
Définition des attributs et des paramètres	21
Définir les paramètres du workflow	22
Définir les attributs de workflow	23
Limitations de nommage des attributs et des paramètres	23
Schéma de workflow	24
Visualiser le schéma de workflow	25
Créer un workflow dans le schéma de workflow	26
Éléments du schéma	29
Propriétés de l'élément du schéma	34
Liens et liaisons	37
Décisions	44
Gestion des exceptions	47
Utilisation des gestionnaires d'erreurs	49
Éléments Foreach et types composites	50
Ajouter une activité de commutation à un workflow	54
Développement de plug-ins	54
Présentation des plug-ins	55
Contenu et structure d'un plug-in	63
Guide de référence de l'API de plug-in Orchestrator	69

Éléments du fichier de définition du plug-in vso.xml	80
Recommandations pour le développement de plug-ins Orchestrator	99
Obtention des paramètres d'entrée auprès des utilisateurs lors du démarrage d'un workflow	114
Création de la boîte de dialogue des paramètres d'entrée dans l'onglet Présentation	115
Définition des propriétés du paramètre	117
Demander des interactions utilisateurs pendant l'exécution d'un workflow	119
Ajouter une interaction utilisateur à un workflow	120
Définir l'attribut security.group de l'interaction utilisateur	121
Définir l'attribut timeout.date sur une date absolue	122
Calculer un délai d'expiration relatif pour les interactions utilisateur	123
Définir l'attribut timeout.date sur une date relative	125
Définir les entrées externes d'une interaction utilisateur	125
Définir le comportement d'exception dans l'interaction utilisateur	126
Créer la boîte de dialogue des paramètres d'entrée pour l'interaction utilisateur	128
Répondre à une demande d'interaction utilisateur	129
Appeler des workflows dans des workflows	130
Éléments du workflow appelant des workflows	131
Appeler un workflow de manière synchrone	134
Appeler un workflow de manière asynchrone	135
Planifier un workflow	135
Conditions préalables pour appeler un workflow distant au sein d'un autre workflow	137
Appeler plusieurs workflows simultanément	137
Exécuter un workflow sur une sélection d'objets	139
Implémenter les workflows Démarrer les workflows en série et Démarrer les workflows en parallèle	140
Développement de workflows au long cours	141
Définir une date et une heure relatives pour les workflows s'appuyant sur un temporisateur	142
Créer un workflow au long cours basé sur un temporisateur	143
Créer un objet déclencheur	145
Créer un workflow au long cours basé sur un déclencheur	147
Éléments de configuration	148
Créer un élément de configuration	148
Autorisations de l'utilisateur de workflow	150
Définir les autorisations des utilisateurs sur un workflow	150
Validation des workflows	151
Valider un workflow et corriger les erreurs de validation	152
Débogage de workflows	153
Déboguer un workflow	153
Exemple de débogage de workflow	154
Workflows en cours d'exécution	155
Exécuter un workflow dans l'éditeur de workflow	156
Exécuter un workflow	156

Reprise d'une exécution de workflow ayant échoué	158
Définir le comportement pour la reprise d'une exécution de workflow ayant échoué	158
Définir des propriétés personnalisées pour permettre la reprise des exécutions de workflows ayant échoué	159
Reprendre une exécution de workflow ayant échoué	159
Générer la documentation sur les workflows	160
Utiliser l'historique des versions du workflow	160
Restaurer des workflows supprimés	161
Développer un exemple de workflow simple	162
Créer l'exemple de workflow simple	164
Créer le schéma de l'exemple de workflow simple	165
Créer les zones de l'exemple de workflow simple	167
Définir les paramètres de l'exemple de workflow simple	169
Définir les liaisons de décision de l'exemple de workflow simple	170
Relier les éléments d'action de l'exemple de workflow simple	171
Relier les éléments Tâche inscriptible de l'exemple de workflow simple	175
Définir les liaisons d'exception de l'exemple de workflow simple	183
Définir les propriétés en lecture seule des attributs de l'exemple de workflow simple	184
Définir les propriétés du paramètre de l'exemple de workflow simple	185
Définir l'agencement de la boîte de dialogue des paramètres d'entrée d'un exemple de workflow simple	187
Valider et exécuter l'exemple de workflow simple	188
Développer un workflow complexe	189
Créer l'exemple de workflow complexe	191
Créer une action personnalisée pour l'exemple de workflow complexe	192
Créer le schéma de l'exemple de workflow complexe	193
Créer les zones de l'exemple de workflow complexe	196
Définir les paramètres de l'exemple de workflow complexe	197
Définir les liaisons de l'exemple de workflow complexe	198
Définir les propriétés d'attribut d'un exemple de workflow complexe	209
Créer la mise en forme des paramètres d'entrée de l'exemple de workflow complexe	210
Valider et exécuter l'exemple de workflow complexe	211

## 2 Scripts 213

Éléments d'Orchestrator réclamant l'écriture de scripts	213
Limites de l'implémentation Mozilla Rhino dans Orchestrator	214
Utilisation de l'API de scripts d'Orchestrator	215
Accéder au moteur de scripts depuis l'éditeur de workflow	216
Accéder au moteur de scripts depuis l'éditeur d'actions ou de stratégies	216
Accéder à l'explorateur d'API Orchestrator	217
Servez-vous de l'Explorateur d'API Orchestrator pour trouver des objets.	217
Rédaction de scripts	219

Ajouter des paramètres aux scripts	220
Accès au système de fichiers Orchestrator à partir de JavaScript et des workflows	221
Accès aux catégories Java à partir de JavaScript	221
Accès aux commandes du système d'exploitation à partir de JavaScript	222
Utilisation des expressions XPath avec le plug-in vCenter Server	222
Utilisation des expressions XPath avec le plug-in vCenter Server	222
Directives de gestion des exceptions	223
Exemples JavaScript d'Orchestrator	225
Exemples de scripts de base	225
Exemples de scripts d'e-mails	227
Exemples de scripts du système de fichiers	229
Exemples de scripts LDAP	229
Exemples de scripts de journalisation	230
Exemples de scripts de mise en réseau	230
Exemples de scripts de workflow	230
<b>3 Développement d'actions</b>	<b>233</b>
Réutilisation d'actions	233
Accéder à la vue Actions	234
Composants de la vue Actions	234
Création d'actions	234
Créer une action	235
Rechercher des éléments qui implémentent une action	236
Directives de codage des actions	236
Utiliser l'historique des versions des actions	238
Restaurer des actions supprimées	238
<b>4 Créer des éléments de ressources</b>	<b>240</b>
Afficher un élément de ressource	240
Importer un objet externe pour utiliser un élément de ressource	241
Modifier les informations et les droits d'accès d'un élément de ressource	241
Enregistrer un élément de ressource dans un fichier	242
Mettre à jour un élément de ressource	243
Ajouter un élément de ressource à un workflow	244
<b>5 Création de modules</b>	<b>245</b>
Créer un module	245
Définir les autorisations des utilisateurs sur un module	246
<b>6 Développement de plug-ins</b>	<b>248</b>
Présentation des plug-ins	248

Structure d'un plug-in Orchestrator	249
Exposer une API externe à Orchestrator	251
Composants d'un plug-in	251
Rôle du fichier vso.xml	252
Rôles de l'adaptateur de plug-in	253
Rôles de la fabrique de plug-in	254
Rôle des objets de l'outil de recherche	255
Rôle des objets de scripts	256
Rôle des gestionnaires d'événements	256
Contenu et structure d'un plug-in	257
Définition du mappage des applications dans le fichier vso.xml	258
Format du fichier de définition du plug-in vso.xml	259
Attribution de noms aux objets de plug-in	260
Conventions de nommage des objets des plug-ins	261
Structure de fichiers du plug-in	262
Guide de référence de l'API de plug-in Orchestrator	263
Interface IAop	263
Interface IDynamicFinder	264
Interface IPluginAdaptor	264
Interface IPluginEventPublisher	265
Interface IPluginFactory	266
Interface IPluginNotificationHandler	267
Interface IPluginPublisher	268
Interface WebConfigurationAdaptor	268
Classe PluginTrigger	269
Classe PluginWatcher	270
Classe QueryResult	270
Classe SDKFinderProperty	271
Classe PluginExecutionException	272
Classe PluginOperationException	273
Énumération HasChildrenResult	273
Type d'annotation ScriptingAttribute	274
Type d'annotation ScriptingFunction	275
Type d'annotation ScriptingParameter	275
Éléments du fichier de définition du plug-in vso.xml	275
Élément module	275
Élément obsolète	276
Élément obsolète	277
Élément url	277
Élément installation	277
Élément d'action	278

Élément finder-datasource	278
Élément finder-datasource	279
Élément inventaire	280
Élément Fonctions de recherche	280
Élément Fonction de recherche	280
Élément propriétés	282
Élément propriété	282
Élément relations	283
Élément relation	283
Élément id	284
Élément inventaire-enfants	284
Élément lien-relation	284
Élément Événements	285
Élément trigger	285
Élément trigger-properties	285
Élément trigger-property	286
Élément Indicateur	286
Élément scripting-objects	287
Élément objet	287
Élément constructeurs	288
Élément constructeur	288
Élément Paramètres du constructeur	288
Élément Paramètre du constructeur	288
Élément attributs	289
Élément attribut	289
Élément méthodes	290
Élément méthode	290
Exemple d'élément	291
Élément de code	292
Élément paramètres de la méthode	292
Élément paramètre de la méthode	292
Élément singleton	292
Élément Énumérations	293
Élément Énumération	293
Élément Entrées	294
Élément Entrée	294
Recommandations pour le développement de plug-ins Orchestrator	294
Méthodes de création de plug-ins Orchestrator	295
Types des plug-ins Orchestrator	297
Implémentation des plug-ins	301
Recommandations pour le développement de plug-ins Orchestrator	306



[Documentation des chaînes et des API de l'interface utilisateur du plug-in](#) 309

## **7** Création de plug-ins à l'aide de Maven 311

[Créer un plug-in Orchestrator avec Maven à partir d'un archétype](#) 311

[Archétypes Maven](#) 312

[Recommandations pour le développement de plug-ins basés sur Maven](#) 313

# Développement avec VMware vRealize Orchestrator

*Le développement avec VMware vRealize Orchestrator* fournit des informations et des instructions pour le développement de workflows et d'actions VMware<sup>®</sup> vRealize Orchestrator personnalisés.

En outre, la documentation contient des informations sur les éléments Orchestrator qui nécessitent des scripts et propose des exemples JavaScript. *Le développement avec VMware vRealize Orchestrator* fournit également des instructions sur la création de ressources et de modules.

## Public visé

Ces informations s'adressent aux développeurs qui souhaitent créer des workflows et des actions Orchestrator personnalisés, ainsi que des composants de base personnalisés.

---

**Note** Les procédures décrites dans ce guide sont basées sur l'interface utilisateur de vRealize Orchestrator Legacy Client.

---

# Développement de workflows

Vous pouvez développer les workflows dans l'interface client Orchestrator. Le développement de workflows implique l'utilisation de l'éditeur de workflow, du moteur de scripts intégré JavaScript Mozilla Rhino et des API Orchestrator et vCenter Server.

- **Concepts clés des workflows**

Les workflows se composent d'un schéma, d'attributs et de paramètres. Le schéma de workflow est son composant principal, car il définit tous les éléments de workflow et les connexions logiques entre eux. Les attributs et les paramètres de workflow sont les variables que les workflows utilisent pour transférer des données. Orchestrator enregistre un jeton de workflow à chaque exécution du workflow, qui consigne les détails de cette exécution spécifique du workflow.

- **Phases du processus de développement de workflow**

Le processus de développement d'un workflow passe par toute une série de phases. Vous avez la possibilité de suivre une séquence de phases différente ou sauter une phase, selon le type de workflow que vous développez. Vous pouvez, par exemple, créer un workflow sans écrire de script personnalisé.

- **Meilleures pratiques concernant le développement de workflows**

VMware recommande différentes meilleures pratiques pour le développement de workflows Orchestrator par plusieurs utilisateurs et dans un environnement en cluster.

- **Droits d'accès au client Orchestrator**

Par défaut, seuls les membres du groupe LDAP d'administrateurs Orchestrator peuvent accéder au client Orchestrator.

- **Tester des workflows au cours du développement**

Vous avez la possibilité de tester un workflow à tout moment durant son processus de développement, y compris si vous n'avez pas mené le workflow à son terme ou inclus un élément de fin.

- **Création et modification d'un workflow**

Vous pouvez créer les workflows dans le client Orchestrator et les modifier dans l'éditeur de workflow. L'éditeur de workflow est l'IDE du client Orchestrator pour le développement de workflows.

- **Fourniture des informations générales sur le workflow**

Dans l'onglet **Général** de l'éditeur de workflow, vous indiquerez un nom et une description pour le workflow, définirez des attributs et certains aspects du comportement du workflow, définirez le numéro de version, contrôlerez la signature et définirez les autorisations de l'utilisateur.

## ■ Définition des attributs et des paramètres

Une fois que vous avez créé un workflow, vous devez définir ses attributs globaux, ses paramètres d'entrée et ses paramètres de sortie.

## ■ Schéma de workflow

Un schéma de workflow est une représentation graphique d'un workflow sous la forme d'un diagramme de flux entre des éléments de workflow interconnectés. Le schéma de workflow définit l'enchaînement logique d'un workflow.

## ■ Développement de plug-ins

Orchestrator permet une intégration à des solutions de gestion et d'administration par le biais de son architecture ouverte de plug-ins. Vous utilisez le client Orchestrator pour exécuter et créer des workflows de plug-ins, et accéder à l'API des plug-ins.

## ■ Obtention des paramètres d'entrée auprès des utilisateurs lors du démarrage d'un workflow

Si un workflow nécessite des paramètres d'entrée, il ouvre une boîte de dialogue dans laquelle les utilisateurs saisissent les valeurs des paramètres d'entrée requis lorsqu'il s'exécute. Vous pouvez organiser le contenu et l'agencement, ou présentation, de cette boîte de dialogue dans l'onglet **Présentation** de l'éditeur de workflow.

## ■ (Facultatif) Demander des interactions utilisateurs pendant l'exécution d'un workflow

Au cours de son exécution, un workflow peut parfois demander des paramètres d'entrée supplémentaires provenant d'une source externe. Ces paramètres d'entrée peuvent provenir d'une autre application ou d'un autre workflow. L'utilisateur peut également les fournir directement.

## ■ Appeler des workflows dans des workflows

Les workflows peuvent appeler d'autres workflows au cours de leur exécution. Un workflow peut enclencher un autre workflow soit parce qu'il a besoin de son résultat en tant que paramètre d'entrée pour sa propre exécution, soit pour le démarrer et le laisser s'exécuter en autonomie. Les workflows peuvent également enclencher d'autres workflows à une heure ultérieure définie, ou démarrer plusieurs workflows simultanément.

## ■ Exécuter un workflow sur une sélection d'objets

Vous pouvez automatiser des tâches répétitives en exécutant un workflow sur une sélection d'objets. Par exemple, vous pouvez créer un workflow qui capture un snapshot de l'ensemble des machines virtuelles d'un dossier ou un workflow qui met hors tension l'ensemble des machines virtuelles sur un hôte précis.

## ■ Développement de workflows au long cours

Un workflow en attente consomme des ressources système car il ne cesse de sonder l'objet auquel il demande une réponse. Si vous savez qu'un workflow attendra probablement pendant une longue période avant de recevoir une réponse, vous pouvez lui ajouter des éléments de workflow au long cours.

## ■ Éléments de configuration

Un élément de configuration est une liste d'attributs que vous pouvez utiliser pour configurer les constantes tout au long d'un déploiement d'un serveur Orchestrator.

- [Autorisations de l'utilisateur de workflow](#)

Orchestrator définit les niveaux d'autorisation que vous pouvez appliquer aux groupes pour leur accorder ou leur refuser l'accès aux workflows.

- [Validation des workflows](#)

Orchestrator fournit un outil de validation des workflows. Valider un workflow aide à identifier les erreurs qui peuvent s'y trouver et contrôle que les données s'enchaînent correctement d'un élément au suivant.

- [Débogage de workflows](#)

Orchestrator fournit un outil de débogage des workflows. Vous pouvez déboguer un workflow pour vérifier les paramètres et attributs d'entrée et de sortie au début de n'importe quelle activité, remplacer les valeurs des paramètres ou des attributs au cours d'une exécution de workflow en mode modification et reprendre un workflow à partir de la dernière activité ayant échoué.

- [Workflows en cours d'exécution](#)

Un workflow Orchestrator s'exécute suivant un enchaînement logique d'événements.

- [Reprise d'une exécution de workflow ayant échoué](#)

Si un workflow échoue, Orchestrator propose une option pour reprendre son exécution à partir de la dernière activité ayant échoué.

- [Générer la documentation sur les workflows](#)

Vous pouvez utiliser le format PDF pour exporter à tout moment la documentation sur un workflow ou un dossier de workflows que vous sélectionnez.

- [Utiliser l'historique des versions du workflow](#)

Vous pouvez utiliser l'historique des versions pour restaurer un workflow à une version précédente. Vous pouvez restaurer l'état du workflow à une version antérieure ou ultérieure. Vous pouvez également comparer les différences entre l'état du workflow en cours et une version enregistrée de ce workflow.

- [Restaurer des workflows supprimés](#)

Vous pouvez restaurer des workflows supprimés de la bibliothèque de workflows.

- [Développer un exemple de workflow simple](#)

Le développement d'un exemple de workflow simple permet de démontrer les étapes les plus courantes du processus de développement des workflows.

- [Développer un workflow complexe](#)

Le développement d'un exemple de workflow complexe permet de démontrer les étapes les plus courantes du processus de développement des workflows, ainsi que des scénarios plus avancés tels que la création de décisions et de boucles personnalisées.

## Concepts clés des workflows

Les workflows se composent d'un schéma, d'attributs et de paramètres. Le schéma de workflow est son composant principal, car il définit tous les éléments de workflow et les connexions logiques entre eux. Les attributs et les paramètres de workflow sont les variables que les workflows utilisent pour transférer des données. Orchestrator enregistre un jeton de workflow à chaque exécution du workflow, qui consigne les détails de cette exécution spécifique du workflow.

### Paramètres du workflow

Les workflows reçoivent des paramètres d'entrée et génèrent des paramètres de sortie lorsqu'ils s'exécutent.

#### Paramètres d'entrée

La plupart des workflows exigent un certain ensemble de paramètres d'entrée afin de fonctionner. Un paramètre d'entrée est un argument que le workflow traite quand il démarre. L'utilisateur, une application, ou un autre workflow ou une action transmettent les paramètres d'entrée à un workflow, pour que le workflow le traite lorsqu'il démarre.

Par exemple, si un workflow réinitialise une machine virtuelle, le workflow nécessite comme paramètre d'entrée le nom de la machine virtuelle à réinitialiser.

#### Paramètres de sortie

Les paramètres de sortie du workflow représentent le résultat de l'exécution du workflow. Les paramètres de sortie peuvent changer quand un workflow ou un élément du workflow s'exécute. Pendant leur exécution, les workflows peuvent recevoir les paramètres de sortie d'autres workflows comme paramètres d'entrée.

Par exemple, si un workflow crée un snapshot d'une machine virtuelle, le paramètre de sortie du workflow est le snapshot qui en résulte.

### Attributs de workflow

Les éléments du workflow traitent les données qu'ils reçoivent comme paramètres d'entrée et définissent les données résultantes comme attributs du workflow ou paramètres de sortie.

Les attributs de workflow en lecture seule font office de constantes globales d'un workflow. Les attributs sur lesquels vous disposez des droits d'écriture font office de variables globales d'un workflow.

Vous pouvez utiliser les attributs pour transférer des données entre des éléments d'un workflow. Voici les différentes manières d'obtenir des attributs :

- Définir les attributs à la création du workflow
- Définir le paramètre de sortie d'un élément du workflow en tant qu'un attribut du workflow
- Hériter des attributs d'un élément de configuration

## Schéma de workflow

Un schéma de workflow est une représentation graphique d'un workflow sous la forme d'un diagramme de flux entre des éléments de workflow interconnectés. Le schéma de workflow est l'élément le plus important d'un workflow, car il détermine sa logique.

## Présentation d'un workflow

Lorsqu'un utilisateur exécute un workflow, il indique les valeurs des paramètres d'entrée du workflow dans la présentation de ce dernier. Lorsque vous organisez la présentation du workflow, tenez compte du type et du nombre de paramètres d'entrée.

## Jetons de workflow

Un jeton de workflow représente un workflow en cours d'exécution ou qui s'est exécuté.

Un workflow est une description résumée d'un processus qui définit une séquence générique d'étapes et un ensemble générique de paramètres d'entrée demandés. Lorsque vous exécutez un workflow avec un ensemble de paramètres d'entrée réels, vous recevez une instance de ce workflow résumé qui se comporte selon les paramètres d'entrée spécifiques que vous lui attribuez. Cette instance spécifique de workflow en cours d'exécution ou achevée est appelée « jeton de workflow ».

## Attributs du jeton de workflow

Les attributs de jeton de workflow sont des paramètres spécifiques qui permettent au workflow de s'exécuter. Les attributs de jeton de workflow constituent une agrégation des attributs globaux du workflow et des paramètres d'entrée et de sortie spécifiques avec lesquels vous exécutez le jeton de workflow.

## Phases du processus de développement de workflow

Le processus de développement d'un workflow passe par toute une série de phases. Vous avez la possibilité de suivre une séquence de phases différente ou sauter une phase, selon le type de workflow que vous développez. Vous pouvez, par exemple, créer un workflow sans écrire de script personnalisé.

En général, on développe un workflow suivant les étapes ci-dessous.

- 1 Création d'un nouveau workflow ou création d'un double de workflow existant à partir de la bibliothèque standard.
- 2 Fourniture des informations générales relatives au workflow.
- 3 Définition des paramètres d'entrée du workflow.
- 4 Agencement et création de liens sur le schéma de workflow en vue de définir le déroulement logique du workflow.
- 5 Établissement des liaisons entre les paramètres d'entrée et de sortie de chaque élément du schéma et les attributs du workflow.

- 6 Écriture des scripts nécessaires aux éléments de tâche inscriptible ou aux éléments de décision personnalisés.
- 7 Création de la présentation du workflow pour définir l'agencement de la boîte de dialogue des paramètres d'entrée que les utilisateurs verront lorsqu'ils exécutent le workflow.
- 8 Validez le workflow.

## Meilleures pratiques concernant le développement de workflows

VMware recommande différentes meilleures pratiques pour le développement de workflows Orchestrator par plusieurs utilisateurs et dans un environnement en cluster.

- Chaque développeur dispose d'une instance Orchestrator autonome test dédiée pour créer et développer des workflows.
- Les workflows sont enregistrés comme des projets Maven sur un système de contrôle de code source partagé.
- Pour garantir des performances optimales pour le déploiement en production d'Orchestrator, il est préférable d'importer les workflows dans une fenêtre planifiée.
- Lorsque vous importez des workflows dans un cluster Orchestrator, connectez le client Orchestrator à l'un des nœuds à l'aide de son nom d'hôte local ou adresse IP, au lieu de l'adresse du serveur virtuel d'équilibrage de charge.

---

**Note** Toute modification d'un workflow prend effet avec l'exécution du workflow suivant.

---

## Droits d'accès au client Orchestrator

Par défaut, seuls les membres du groupe LDAP d'administrateurs Orchestrator peuvent accéder au client Orchestrator.

L'administrateur Orchestrator peut octroyer un accès au client Orchestrator à d'autres groupes d'utilisateurs en configurant, au minimum, l'autorisation **Vue**.

Afin de vous permettre d'accéder au client Orchestrator, l'administrateur peut soit vous ajouter au groupe LDAP d'administrateurs d'Orchestrator, soit définir les autorisations **Vue**, **Inspecter**, **Modifier**, **Exécuter** ou **Admin** pour un groupe dont vous êtes membre.

## Tester des workflows au cours du développement

Vous avez la possibilité de tester un workflow à tout moment durant son processus de développement, y compris si vous n'avez pas mené le workflow à son terme ou inclus un élément de fin.



Par défaut, Orchestrator vérifie que le workflow est valide avant que vous puissiez l'exécuter. Vous pouvez désactiver la validation automatique au cours du développement du workflow, pour pouvoir exécuter des workflows partiels à des fins de test.

---

**Note** N'oubliez pas de réactiver la validation automatique à l'issue du développement du workflow.

---

#### Procédure

- 1 Dans le menu du client Orchestrator, cliquez sur **Outils > Préférences utilisateur**.
- 2 Cliquez sur l'onglet **Workflows**.
- 3 Désélectionnez la case **Valider le workflow avant de l'exécuter**.

Vous venez de désactiver la validation automatique du workflow.

## Création et modification d'un workflow

Vous pouvez créer les workflows dans le client Orchestrator et les modifier dans l'éditeur de workflow. L'éditeur de workflow est l'IDE du client Orchestrator pour le développement de workflows.

Vous pouvez ouvrir l'éditeur de workflow en modifiant un workflow existant.

#### ■ [Créer un workflow](#)

Vous pouvez créer des workflows dans la liste hiérarchique des workflows du client Orchestrator.

#### ■ [Modifier un workflow](#)

Vous pouvez modifier un workflow pour apporter des changements à un workflow existant ou pour développer un nouveau workflow vide.

#### ■ [Modifier un workflow à partir de la bibliothèque standard](#)

Orchestrator fournit une bibliothèque standard de workflows que vous pouvez utiliser pour automatiser les opérations dans l'infrastructure virtuelle. Les workflows de la bibliothèque standard sont verrouillés en lecture seule.

#### ■ [Onglets de l'éditeur de workflow](#)

L'éditeur de workflow se compose d'onglets dans lesquels vous pouvez modifier les composants des workflows.

## Créer un workflow

Vous pouvez créer des workflows dans la liste hiérarchique des workflows du client Orchestrator.

#### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Workflows**.
- 3 (Facultatif) Cliquez avec le bouton droit sur la racine de la liste hiérarchique des workflows ou sur un dossier de la liste, puis sélectionnez **Ajouter un dossier** pour créer un nouveau dossier de workflows.

- 4 (Facultatif) Entrez le nom du nouveau dossier.
- 5 Cliquez avec le bouton droit sur le nouveau dossier ou sur un dossier existant, puis sélectionnez **Nouveau workflow**.
- 6 Nommez le nouveau workflow, puis cliquez sur **OK**.

Un nouveau workflow vide a été créé dans le dossier que vous avez sélectionné.

#### Étape suivante

Vous pouvez modifier le workflow.

## Modifier un workflow

Vous pouvez modifier un workflow pour apporter des changements à un workflow existant ou pour développer un nouveau workflow vide.

#### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Workflows**.
- 3 Développez la liste hiérarchique des workflows pour naviguer jusqu'au workflow que vous souhaitez modifier.
- 4 Pour ouvrir le workflow que vous souhaitez modifier, cliquez avec le bouton droit dessus puis sélectionnez **Modifier**.

L'éditeur de workflow ouvre le workflow à modifier.

## Modifier un workflow à partir de la bibliothèque standard

Orchestrator fournit une bibliothèque standard de workflows que vous pouvez utiliser pour automatiser les opérations dans l'infrastructure virtuelle. Les workflows de la bibliothèque standard sont verrouillés en lecture seule.

Pour modifier un workflow de la bibliothèque standard, vous devez créer un doublon de ce workflow. Vous pouvez modifier les doublons de workflows ou les workflows personnalisés.

#### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Workflows**.
- 3 (Facultatif) Cliquez avec le bouton droit sur la racine de la liste hiérarchique des dossiers de workflow, puis sélectionnez **Nouveau dossier** pour créer un dossier comportant le workflow à modifier.
- 4 Développez la liste hiérarchique de la **Bibliothèque** des workflows standard afin de naviguer jusqu'au workflow à modifier.

- 5 Cliquez avec le bouton droit sur le workflow à modifier.

L'option **Modifier** est grisée. Le workflow est en lecture seule.

- 6 Cliquez avec le bouton droit sur le workflow, puis sélectionnez **Dupliquer le workflow**.

- 7 Indiquez un nom pour le doublon de workflow.

Par défaut, Orchestrator nomme le doublon de workflow Copie de *workflow\_name*.

- 8 Cliquez sur la valeur **Dossier de workflow** pour rechercher un dossier dans lequel enregistrer le doublon de workflow.

Sélectionnez le dossier que vous avez créé dans [Étape 3](#). Si vous n'avez pas créé de dossier, sélectionnez un dossier qui ne se trouve pas dans la bibliothèque des workflows standard.

- 9 Cliquez sur **Oui** ou **Non** pour copier l'historique des versions du workflow dans le doublon.

Option	Description
<b>Oui</b>	L'historique des versions du workflow d'origine est copié dans le doublon.
<b>Non</b>	La version du doublon est rétablie à 0.0.0.

- 10 Cliquez sur **Dupliquer** pour dupliquer le workflow.

- 11 Cliquez avec le bouton droit sur le doublon de workflow, puis sélectionnez **Modifier**.

L'éditeur de workflow s'ouvre. Vous pouvez modifier le doublon de workflow.

Vous avez dupliqué un workflow à partir de la bibliothèque standard. Vous pouvez modifier le doublon de workflow.

## Onglets de l'éditeur de workflow

L'éditeur de workflow se compose d'onglets dans lesquels vous pouvez modifier les composants des workflows.

Tableau 1-1. Onglets de l'éditeur de workflow

Onglet	Description
<b>Général</b>	Modifier le nom du workflow, fournir une description de ce qu'il fait, définir le numéro de version, consulter les autorisations de l'utilisateur, définir le comportement du workflow si le serveur Orchestrator redémarre et définir les attributs globaux du workflow.
<b>Entrées</b>	Définir les paramètres que le workflow réclame lorsqu'il s'exécute. Ces paramètres d'entrée sont les données que traite le workflow. Le comportement du workflow varie suivant ces paramètres.
<b>Sorties</b>	Définir les valeurs que génère le workflow au terme de l'exécution. D'autres workflows ou actions peuvent utiliser ces valeurs lorsqu'ils s'exécutent.

Tableau 1-1. Onglets de l'éditeur de workflow (suite)

Onglet	Description
<b>Schéma</b>	Assembler le workflow. Vous assemblez le workflow par glisser-déposer des éléments du schéma de workflow se trouvant sur la palette des workflows du côté gauche de l'onglet <b>Schéma</b> . Cliquez sur un élément du diagramme de schéma vous permet de définir et modifier le comportement de l'élément dans la moitié inférieure de l'onglet <b>Schéma</b> .
<b>Présentation</b>	Définir l'agencement de la boîte de dialogue des saisies utilisateur qui apparaît lorsque les utilisateurs exécutent un workflow. Vous pouvez organiser les paramètres et attributs par étapes et groupes de présentation afin de faciliter l'identification des paramètres dans la boîte de dialogue des paramètres d'entrée. Les contraintes aux paramètres d'entrée que les utilisateurs peuvent renseigner dans la présentation se définissent en spécifiant les propriétés des paramètres.
<b>Références des paramètres</b>	Notez quels éléments du workflow consomment les attributs et paramètres dans l'enchaînement logique du workflow. Cet onglet montre également les contraintes que vous avez fixées à ces paramètres et attributs dans l'onglet <b>Présentation</b> .
<b>Jetons de workflow</b>	Afficher les détails sur chaque exécution de workflow. On peut trouver parmi ces informations le statut du workflow, l'utilisateur qui l'a exécuté, l'état du groupe de l'élément actuel et la date et l'heure de début et de fin du workflow.
<b>Événements</b>	Afficher les informations relatives à chaque événement individuel survenant au cours de l'exécution d'un workflow. On peut trouver parmi ces informations une description de l'événement, son type et son origine et la date et l'heure auxquelles il s'est produit.
<b>Autorisations</b>	Définissez les autorisations en vue des interactions des utilisateurs ou groupes d'utilisateurs avec le workflow.

## Fourniture des informations générales sur le workflow

Dans l'onglet **Général** de l'éditeur de workflow, vous indiquerez un nom et une description pour le workflow, définirez des attributs et certains aspects du comportement du workflow, définirez le numéro de version, contrôlerez la signature et définirez les autorisations de l'utilisateur.

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Cliquez sur l'onglet **Général** de l'éditeur de workflow.
- 2 Cliquez sur les chiffres de la **Version** pour définir un numéro de version pour le workflow.

La boîte de dialogue **Commentaire sur la version** s'ouvre.

- 3 Saisissez un commentaire pour cette version du workflow et cliquez sur **OK**.

Par exemple, saisissez **Création initiale** si vous venez de créer le workflow.

Une nouvelle version du workflow est donc créée. Vous pourrez par la suite restituer l'état du workflow dans cette version.

- 4 Définissez la façon dont le workflow se comportera si le serveur Orchestrator redémarre en définissant la valeur du **Comportement au redémarrage du serveur**.
  - Laissez la valeur par défaut de **Reprendre l'exécution du workflow** pour que le workflow reprenne au point il a été interrompu dans son exécution par l'arrêt du serveur.
  - Cliquez sur **Reprendre l'exécution du workflow** et sélectionnez **Ne pas reprendre l'exécution du workflow (définie comme ÉCHEC)** pour empêcher le workflow de redémarrer si le serveur Orchestrator redémarre.

Empêchez le workflow de redémarrer s'il dépend de l'environnement dans lequel il s'exécute. Par exemple, si un workflow nécessite un vCenter Server précis et que vous reconfigurez Orchestrator pour qu'il se connecte à un autre vCenter Server, le fait de redémarrer le workflow après redémarrage du serveur Orchestrator provoquera l'échec du workflow.

- 5 Dans la zone de texte **Description**, saisissez une description détaillée du workflow.

- 6 Cliquez sur **Enregistrer** au bas de l'éditeur de workflow.

Un message vert dans la partie inférieure gauche de l'éditeur de workflow confirme que vos changements ont été enregistrés.

Vous avez donc défini les aspects du comportement du workflow, défini sa version et défini les opérations que les utilisateurs sont à même de réaliser sur le workflow.

### Étape suivante

Il vous faut maintenant définir les attributs et paramètres du workflow.

## Définition des attributs et des paramètres

Une fois que vous avez créé un workflow, vous devez définir ses attributs globaux, ses paramètres d'entrée et ses paramètres de sortie.

Les attributs de workflow stockent les données que les workflows traitent en interne. Les paramètres d'entrée des workflows sont des données entrées par une source externe, telle qu'un utilisateur ou un autre workflow. Les paramètres de sortie des workflows correspondent aux données que les workflows renvoient à la fin de leur exécution.

- **Définir les paramètres du workflow**  
Vous pouvez utiliser les paramètres d'entrée et de sortie pour transférer les données dans/depuis le workflow.
- **Définir les attributs de workflow**  
Les attributs de workflow correspondent aux données que les workflows traitent.

## ■ Limitations de nommage des attributs et des paramètres

Vous pouvez utiliser les expressions OGNL pour déterminer les paramètres d'entrée de façon dynamique au cours de l'exécution d'un workflow. L'analyseur OGNL d'Orchestrator utilise certains mots-clés au cours du traitement OGNL que vous ne pouvez pas utiliser dans les noms des attributs ou des paramètres.

## Définir les paramètres du workflow

Vous pouvez utiliser les paramètres d'entrée et de sortie pour transférer les données dans/depuis le workflow.

Vous pouvez définir les paramètres d'un workflow dans l'éditeur de workflow. Les paramètres d'entrée sont les premières données dont le workflow a besoin pour s'exécuter. Ce sont les utilisateurs qui entrent les valeurs des paramètres d'entrée lorsqu'ils exécutent le workflow. Les paramètres de sortie correspondent aux données que le workflow renvoie à la fin de son exécution.

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Cliquez sur l'onglet approprié de l'éditeur de workflow.
  - Cliquez sur **Entrées** pour créer des paramètres d'entrée.
  - Cliquez sur **Sorties** pour créer des paramètres de sortie.
- 2 Cliquez avec le bouton droit dans l'onglet des paramètres, puis sélectionnez **Ajouter un paramètre**.
- 3 Cliquez sur le nom du paramètre pour le modifier.
 

Le nom par défaut est `arg_in_X` pour les paramètres d'entrée et `arg_out_X` pour les paramètres de sortie, où *X* est un nombre.
- 4 (Facultatif) Pour modifier la valeur d'un type de paramètre, cliquez sur la valeur puis sélectionnez-en une dans la liste des valeurs disponibles.
 

La valeur par défaut du type de paramètre est Chaîne.
- 5 Ajoutez une description du paramètre dans la zone de texte **Description**.
- 6 (Facultatif) Si vous décidez que le paramètre doit devenir un attribut, cliquez avec le bouton droit sur le paramètre, puis sélectionnez **Transformer en attribut** pour transformer le paramètre en attribut.

Vous venez de définir un paramètre d'entrée ou de sortie pour le workflow.

### Étape suivante

Une fois les paramètres du workflow définis, vous devez créer le schéma de workflow.

## Définir les attributs de workflow

Les attributs de workflow correspondent aux données que les workflows traitent.

**Note** Vous pouvez également définir les attributs de workflow dans les éléments du schéma de workflow lorsque vous créez ce dernier. Il est souvent plus simple de définir un attribut lors de la création de l'élément de schéma qui le traite.

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Cliquez sur l'onglet **Général** de l'éditeur de workflow.

Le volet Attributs apparaît dans l'onglet **Général** situé dans la moitié inférieure de l'écran.

- 2 Cliquez avec le bouton droit dans le volet Attributs, puis sélectionnez **Ajouter un attribut**.

Un nouvel attribut apparaît dans la liste des attributs, avec Chaîne comme type par défaut.

- 3 Cliquez sur le nom de l'attribut pour le modifier.

Le nom par défaut est attX, où X est un nombre.

**Note** Les attributs du workflow ne doivent pas porter le même nom que les paramètres du workflow.

- 4 Cliquez sur le type de l'attribut pour sélectionner un nouveau type dans la liste des valeurs possibles.

Le type d'attribut par défaut est Chaîne.

- 5 Cliquez sur la valeur d'attribut à définir ou sélectionnez une valeur en fonction du type de l'attribut.

- 6 Ajoutez une description de l'attribut dans la zone de texte **Description**.

- 7 Si l'attribut n'est pas une variable mais une constante, cliquez sur la case à cocher à gauche de son nom pour que sa valeur soit en lecture seule.

L'icône en forme de verrou indique la colonne des cases de lecture seule.

- 8 (Facultatif) Si vous décidez que l'attribut doit être un paramètre d'entrée ou de sortie plutôt qu'un attribut, cliquez avec le bouton droit sur l'attribut, puis sélectionnez **Transformer en paramètre d'ENTRÉE/de SORTIE** pour transformer l'attribut en paramètre.

Vous venez de définir un attribut pour le workflow.

### Étape suivante

Vous pouvez maintenant définir les paramètres d'entrée et de sortie du workflow.

## Limitations de nommage des attributs et des paramètres

Vous pouvez utiliser les expressions OGNL pour déterminer les paramètres d'entrée de façon dynamique au cours de l'exécution d'un workflow. L'analyseur OGNL d'Orchestrator utilise certains mots-clés au

cours du traitement OGNL que vous ne pouvez pas utiliser dans les noms des attributs ou des paramètres.

L'utilisation d'un mot-clé OGNL réservé en tant que préfixe d'un nom d'attribut ne perturbe pas le traitement OGNL. Par exemple, vous pouvez nommer un paramètre `trueParameter`. Les mots-clés réservés ne sont pas sensibles à la casse.

Vous ne pouvez pas utiliser les mots-clés suivants dans les noms d'attributs et de paramètres des workflows.

**Tableau 1-2. Mots-clés interdits dans les noms d'attributs et de paramètres**

Mot clé interdit	Mot clé interdit	Mot clé interdit
■ <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>

## Schéma de workflow

Un schéma de workflow est une représentation graphique d'un workflow sous la forme d'un diagramme de flux entre des éléments de workflow interconnectés. Le schéma de workflow définit l'enchaînement logique d'un workflow.

### ■ Visualiser le schéma de workflow

Le schéma d'un workflow peut être visualisé dans l'onglet **Schéma** de ce workflow dans le client Orchestrator.

### ■ Créer un workflow dans le schéma de workflow

Les schémas de workflow se composent d'une séquence d'éléments de schéma. Les éléments du schéma de workflow sont les composants de base du workflow. Il peut s'agir de décisions, de tâches en script, d'actions, de gestionnaires d'exceptions ou bien même d'autres workflows.



## ■ Éléments du schéma

L'éditeur de workflow présente les éléments du schéma de workflow dans les menus de l'onglet **Schéma**. Vous pouvez utiliser les éléments du schéma disponibles dans l'onglet **Schéma** pour constituer un workflow.

## ■ Propriétés de l'élément du schéma

Les éléments du schéma possèdent des propriétés que vous pouvez définir et modifier dans l'onglet **Schéma** de la palette des workflows.

## ■ Liens et liaisons

Les liens entre les éléments déterminent l'enchaînement logique du workflow. Les liaisons renseignent les éléments avec des données issues d'autres éléments par la liaison des paramètres d'entrée et de sortie aux attributs du workflow.

## ■ Décisions

Les workflows peuvent implémenter des fonctions de décision qui définissent des suites d'actions différentes basées sur une instruction booléenne `true` ou `false`.

## ■ Gestion des exceptions

La gestion des exceptions récupère toutes les erreurs qui se produisent au cours de l'exécution d'un élément de schéma. La gestion des exceptions définit le comportement de l'élément de schéma lorsqu'une erreur se produit.

## ■ Utilisation des gestionnaires d'erreurs

Vous pouvez utiliser un gestionnaire d'erreurs standard pour déterminer le comportement à adopter si une erreur se produit dans un élément de schéma de workflow spécifique. Vous pouvez utiliser un gestionnaire d'erreurs global pour déterminer le comportement à adopter si une erreur non détectée par le gestionnaire d'erreurs standard se produit.

## ■ Éléments Foreach et types composites

Vous pouvez insérer un élément Foreach dans le workflow que vous développez afin d'exécuter un workflow secondaire qui couvre les tableaux de paramètres ou d'attributs. Pour améliorer la compréhension et la lecture du workflow, vous pouvez regrouper plusieurs paramètres de workflow de types différents et connectés logiquement dans un type unique appelé « type composite ».

## ■ Ajouter une activité de commutation à un workflow

Vous pouvez ajouter une activité de commutation de base à un schéma de workflow pour définir les cas d'utilisation en fonction des attributs ou paramètres du workflow.

# Visualiser le schéma de workflow

Le schéma d'un workflow peut être visualisé dans l'onglet **Schéma** de ce workflow dans le client Orchestrator.

## Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Localisez un workflow dans la liste hiérarchique des workflows.

### 3 Cliquez dessus.

Les informations relatives à ce workflow s'affichent dans le volet droit.

### 4 Sélectionnez l'onglet **Schéma** dans le volet droit.

Vous voyez alors une représentation graphique du workflow.

## Créer un workflow dans le schéma de workflow

Les schémas de workflow se composent d'une séquence d'éléments de schéma. Les éléments du schéma de workflow sont les composants de base du workflow. Il peut s'agir de décisions, de tâches en script, d'actions, de gestionnaires d'exceptions ou bien même d'autres workflows.

Vous pouvez créer des workflows dans l'éditeur de workflow en faisant glisser des éléments de schéma depuis la palette située à gauche de l'éditeur et en les déposant dans le diagramme du schéma.

## Modifier un schéma de workflow

Pour élaborer un workflow, vous devez créer une séquence d'éléments de schéma qui définissent l'enchaînement logique du futur workflow.

Par défaut, tous les éléments du schéma de workflow sont reliés. Les liens entre les éléments sont représentés par des flèches. Lorsque vous ajoutez un nouvel élément au schéma de workflow, vous devez le faire glisser jusqu'à une flèche ou un élément de workflow existant qui n'est pas relié à un élément suivant. Après l'ajout des éléments au schéma, vous pouvez supprimer les liens existants et créer de nouveaux liens pour définir l'enchaînement logique du workflow.

Vous pouvez copier un élément ou une sélection d'éléments à partir du schéma d'un workflow existant, et le/la coller dans le schéma du workflow que vous modifiez. Reportez-vous à [Copier des éléments du schéma du workflow](#).

Un schéma de workflow doit présenter au moins un élément **Fin de workflow**. Il peut en présenter plusieurs.

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.
- 2 Faites glisser un élément de schéma depuis le menu **Générique** du panneau de gauche vers le schéma de workflow.
- 3 Double-cliquez sur l'élément que vous avez fait glisser dans le schéma de workflow, entrez un nom approprié, puis appuyez sur Entrée.

Les éléments doivent posséder un nom unique dans le contexte du workflow.

Vous ne pouvez pas renommer les éléments **Minuterie d'attente**, **Événement d'attente**, **Fin de workflow** ou **Générer une exception**.

- 4 (Facultatif) Cliquez avec le bouton droit sur un élément du schéma, puis sélectionnez **Copier**.

- 5 (Facultatif) Cliquez avec le bouton droit sur l'emplacement souhaité du schéma, puis sélectionnez **Coller**.

Copier/coller les éléments de schéma existants représente un moyen rapide d'ajouter des éléments identique au schéma. Tous les paramètres de l'élément copié apparaissent dans l'élément collé, à l'exception de l'état du groupe. Ajustez les paramètres de l'élément collé comme il convient.

- 6 Faites glisser les éléments de schéma des menus **De base**, **Journal** ou **Réseau** dans le schéma de workflow.

Vous pouvez modifier les noms des éléments des menus **De base**, **Journal** ou **Réseau**. Vous ne pouvez pas modifier leurs scripts.

- 7 Faites glisser les éléments de schéma du menu **Générique** dans le schéma de workflow.

Lorsque vous faites glisser des actions ou des workflows dans le schéma de workflow, une boîte de dialogue vous permettant de rechercher l'action ou le workflow à insérer apparaît.

- 8 Dans la zone de texte **Filtre**, entrez le nom ou une partie du nom du workflow ou de l'action à insérer dans le workflow.

Les workflows ou les actions correspondant à la recherche apparaissent dans la boîte de dialogue.

- 9 Double-cliquez sur un workflow ou une action à sélectionner.

Vous venez d'insérer le workflow ou l'action dans le schéma de workflow.

- 10 Répétez cette procédure jusqu'à ce que vous ayez ajouté tous les éléments de schéma nécessaires au schéma de workflow.

### Étape suivante

Définissez les propriétés des éléments que vous avez ajoutés au schéma de workflow, puis connectez-les entre eux.

## Copier des éléments du schéma du workflow

Vous pouvez copier un élément ou une sélection d'éléments à partir du schéma d'un workflow existant, et le/la coller dans le schéma du workflow que vous modifiez.

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.
- 2 Dans le volet de gauche, sélectionnez le workflow dans lequel vous souhaitez copier des éléments du schéma.
  - Cliquez sur **Tous les workflows** et sélectionnez le workflow dans la liste hiérarchique des workflows.
  - Saisissez le nom du workflow dans la zone de texte de recherche, puis appuyez sur Entrée.

- 3 Cliquez avec le bouton droit sur le workflow sélectionné, puis cliquez sur **Ouvrir**.

Une fenêtre s'affiche avec les propriétés du workflow.

- 4 Dans la fenêtre du workflow, cliquez sur l'onglet **Schéma**.

- 5 Sélectionnez un ou plusieurs éléments du schéma de workflow, cliquez avec le bouton droit sur cette sélection, puis cliquez sur **Copier**.

- 6 Dans l'onglet **Schéma** du workflow que vous modifiez, cliquez avec le bouton droit, puis sélectionnez **Coller**.

Vous avez copié/collé des éléments de schéma d'un workflow vers un autre.

### Étape suivante

Vous devez relier les éléments de schéma copiés au schéma de workflow existant.

## Promouvoir les paramètres d'entrée et de sortie

Vous pouvez promouvoir les paramètres d'entrée et de sortie d'un élément enfant dans le workflow parent.

Vous pouvez promouvoir un attribut personnalisé que vous avez défini dans l'onglet **Général** de l'éditeur de workflow. Vous pouvez promouvoir des attributs prédéfinis en remplaçant uniquement un paramètre d'entrée par un attribut de type correspondant.

---

**Note** Si vous promouvez un attribut prédéfini et que vous lui affectez une valeur personnalisée, un doublon d'attribut est créé pour éviter le remplacement de la valeur de l'attribut d'origine. Le doublon d'attribut conserve le nom de l'attribut d'origine et ajoute une valeur numérique au bout du nom de l'attribut.

---

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.

- 2 Ajoutez un élément de workflow ou d'action au schéma de workflow.

La notification suivante apparaît dans la partie supérieure du volet du schéma.

Souhaitez-vous ajouter les paramètres de l'activité en tant qu'entrée ou sortie au workflow en cours ?

- 3 Dans la notification, cliquez sur **Configuration**.

Une fenêtre contextuelle s'affiche avec les options disponibles.

#### 4 Sélectionnez le type de mappage pour chaque paramètre d'entrée.

Option	Description
Entrée	L'argument est mappé vers un paramètre de workflow d'entrée.
Ignorer	L'argument est mappé vers une valeur NULL.
Valeur	L'argument est mappé vers un attribut avec une valeur que vous pouvez définir dans la colonne Valeur.

#### 5 Sélectionnez le type de mappage pour chaque paramètre de sortie.

Option	Description
Output	L'argument est mappé vers un paramètre de workflow de sortie.
Ignorer	L'argument est mappé vers une valeur NULL.
Variable locale	L'argument est mappé vers un attribut.

#### 6 Cliquez sur **Promouvoir**.

Vous avez promu des paramètres dans le workflow parent.

## Modifier les résultats de la recherche

La zone de texte **Rechercher** sert à dénicher des éléments tels que workflows ou actions. Si une recherche renvoie un résultat partiel, vous pouvez modifier le nombre de résultats que renverra la recherche.

Lorsque vous utilisez la fonction de recherche sur un élément, une zone de message verte indique que la recherche affiche l'ensemble des résultats. Une zone de message jaune indique que la recherche n'affiche que des résultats partiels.

### Procédure

- (Facultatif) Si vous apportez des modifications à un workflow dans l'éditeur de workflow, cliquez sur **Enregistrer et fermer** pour quitter l'éditeur.
- Dans le menu du client Orchestrator, sélectionnez **Outils > Préférences utilisateur**.
- Cliquez sur l'onglet **Général**.
- Dans la zone de texte **Taille maximum du résultat de recherche**, saisissez le nombre de résultats que la recherche doit renvoyer.
- Dans la boîte de dialogue Préférences utilisateur, cliquez sur **Enregistrer et fermer**.

Vous avez donc modifié le nombre de résultats que la recherche doit renvoyer.

## Éléments du schéma

L'éditeur de workflow présente les éléments du schéma de workflow dans les menus de l'onglet **Schéma**. Vous pouvez utiliser les éléments du schéma disponibles dans l'onglet **Schéma** pour constituer un workflow.

Tableau 1-3. Éléments et icônes du schéma






Nom de l'élément du schéma	Description	Icône	Emplacement dans l'éditeur de workflow
<b>Début de workflow</b>	Point de départ du workflow. Tous les workflows contiennent cet élément. Un workflow ne peut comporter qu'un seul élément de départ. Les éléments de départ comportent une sortie, mais aucune entrée et ne peuvent être escamotés du schéma de workflow.		Toujours présent sur l'onglet <b>Schéma</b>
<b>Tâche inscriptible</b>	Tâches universelles que vous définissez. Les fonctions JavaScript sont à écrire dans cet élément.		Palette des workflows <b>générique</b>
<b>Décision</b>	Fonction booléenne. Les éléments Décision prennent un paramètre en entrée et renvoient soit <code>true</code> , soit <code>false</code> . Le type de décision que prend l'élément dépend du type de paramètre d'entrée. Les éléments Décision permettent au workflow de prendre différents embranchements en fonction du paramètre d'entrée reçu par l'élément Décision. Si le paramètre d'entrée reçu correspond à une valeur attendue, le workflow poursuit le chemin emprunté. Si l'entrée diffère de la valeur attendue, le workflow emprunte un autre chemin.		Palette des workflows <b>générique</b>
<b>Décision personnalisée</b>	Fonction booléenne. Les décisions personnalisées peuvent prendre plusieurs paramètres en entrée et les traiter en fonction de scripts personnalisés. Renvoie soit <code>true</code> , soit <code>false</code> .		Palette des workflows <b>générique</b>
<b>Activité de décision</b>	Fonction booléenne. Une activité de décision exécute un workflow et opère une liaison entre ses paramètres de sortie et un chemin <code>true</code> ou <code>false</code> .		Palette des workflows <b>générique</b>

Tableau 1-3. Éléments et icônes du schéma (suite)





Nom de l'élément du schéma	Description	Icône	Emplacement dans l'éditeur de workflow
<b>Interaction utilisateur</b>	Permet aux utilisateurs de passer de nouveaux paramètres d'entrée au workflow. Vous pouvez concevoir la manière dont l'élément d'interaction utilisateur adresse sa demande de paramètres d'entrée et fixer des contraintes sur les paramètres que les utilisateurs peuvent fournir. Vous pouvez définir des autorisations pour décider des utilisateurs qui pourront fournir les paramètres d'entrée. Lorsqu'un workflow en cours d'exécution arrive sur un élément d'interaction utilisateur, il passe à l'état passif et invite l'utilisateur à une saisie de données. Vous avez la possibilité de définir un délai d'expiration avant lequel les utilisateurs doivent saisir des données. Le workflow reprend son cours en fonction des données que l'utilisateur lui a transmises ou renvoie une exception si le délai imparti a expiré. Pendant qu'il attend la réponse de l'utilisateur, le jeton du workflow est à l'état <code>waiting</code> .		Palette des workflows générique
<b>Minuterie d'attente</b>	Sert aux workflows au long cours. Lorsqu'un workflow en cours d'exécution arrive sur un élément Minuterie d'attente, il passe à l'état passif. À vous de définir une date absolue à laquelle le workflow reprendra son cours. Pendant qu'il attend que la date arrive, le jeton du workflow est à l'état <code>waiting-signal</code> .		Palette des workflows générique
<b>Événement d'attente</b>	Sert aux workflows au long cours. Lorsqu'un workflow en cours d'exécution arrive sur un élément Événement d'attente, il passe à l'état passif. À vous de définir l'événement déclencheur que le workflow attend pour reprendre son cours. Pendant qu'il attend que la date arrive, le jeton du workflow est à l'état <code>waiting-signal</code> .		Palette des workflows générique
<b>Fin de workflow</b>	Point final d'un workflow. Vous pouvez avoir une multiplicité d'éléments de fin dans un schéma pour représenter les différentes issues possibles du workflow. Les éléments de fin doivent posséder une entrée et une sortie. Lorsqu'un workflow atteint un élément Fin de workflow, le jeton du workflow passe à l'état <code>completed</code> .		Palette des workflows générique

Tableau 1-3. Éléments et icônes du schéma (suite)








Nom de l'élément du schéma	Description	Icône	Emplacement dans l'éditeur de workflow
<b>Exception engendrée</b>	Crée une exception et interrompt le workflow. Une multiplicité d'occurrences de cet élément peut être présente dans le schéma du workflow. Les éléments Exception possèdent un paramètre d'entrée, qui ne peut être que du type Chaîne, mais aucun paramètre de sortie. Lorsqu'un workflow atteint un élément Exception, le jeton du workflow passe à l'état failed.		Palette des workflows générique
<b>Remarque sur le workflow</b>	Vous permet d'annoter des sections du workflow. Vous pouvez étirer les remarques pour qu'elles portent sur des sections entières du workflow. Vous avez la possibilité de modifier la couleur de l'arrière-plan des remarques afin de distinguer les différentes zones du workflow. Les remarques sur le workflow ne donnent que des informations visuelles pour vous aider à comprendre le schéma.		Palette des workflows générique
<b>Élément Action</b>	Fait appel à une action provenant des bibliothèques d'actions d'Orchestrator. Lorsqu'un workflow atteint un élément Action, il appelle et exécute cette action.		Palette des workflows générique
<b>Élément Workflow</b>	Démarre un autre workflow de façon synchrone. Lorsqu'un workflow atteint un élément Workflow dans son schéma, il exécute ce workflow au sein de son propre processus. Le workflow d'origine ne se poursuit qu'après que le workflow appelé a achevé de s'exécuter.		Palette des workflows générique
<b>Élément Foreach</b>	Exécute un workflow sur chaque élément d'un tableau. Vous pouvez, par exemple, exécuter le workflow Renommer la machine virtuelle sur la totalité des machines virtuelles d'un dossier.		Palette des workflows générique
<b>Workflow asynchrone</b>	Démarre un workflow de façon asynchrone. Lorsqu'un workflow atteint un élément Workflow asynchrone, il lance ce workflow et poursuit son cours. Le workflow d'origine n'attend pas que le workflow appelé ait fini de s'exécuter.		Palette des workflows générique
<b>Workflow planifié</b>	Crée une tâche pour exécuter le workflow à une heure définie, suite à quoi le workflow poursuit son cours.		Palette des workflows générique



Tableau 1-3. Éléments et icônes du schéma (suite)





Nom de l'élément du schéma	Description	Icône	Emplacement dans l'éditeur de workflow
<b>Workflows intégrés</b>	Lance plusieurs workflows simultanément. Vous pouvez choisir d'imbriquer des workflows locaux et des workflows distants se trouvant dans des serveurs différents d'Orchestrator. Et vous avez également la possibilité d'exécuter des workflows avec différentes informations d'identification. Le workflow attendra que tous les workflows intégrés soient terminés avant de poursuivre son cours.		Palette des workflows générique
<b>Gestionnaire d'erreurs</b>	Traite une erreur dans un élément spécifique du workflow. Le workflow peut traiter l'erreur en créant une exception, en appelant un autre workflow ou en exécutant un script personnalisé.		Palette des workflows générique
<b>Gestionnaire d'erreurs par défaut</b>	Traite les erreurs du workflow qui ne sont pas prises en charge par les traitements d'erreur standard. Vous pouvez vous servir de n'importe quel élément disponible du schéma pour traiter les erreurs.		Palette des workflows générique

Tableau 1-3. Éléments et icônes du schéma (suite)

Nom de l'élément du schéma	Description	Icône	Emplacement dans l'éditeur de workflow
<b>Aiguillage</b>	Aiguille sur des chemins de workflows alternatifs, suivant l'attribut ou le paramètre du workflow.		Palette des workflows <b>générique</b>
<b>Tâche prédéfinie</b>	<p>Éléments en script non modifiables réalisant des tâches standard que les workflows emploient couramment. Les tâches suivantes sont prédéfinies :</p> <p><b>De base</b></p> <ul style="list-style-type: none"> <li>■ Repos</li> <li>■ Modifier les informations d'identification</li> <li>■ Attendre la date</li> <li>■ Attendre l'événement personnalisé</li> <li>■ Envoyer un événement personnalisé</li> <li>■ Augmenter le compteur</li> <li>■ Diminuer le compteur</li> </ul> <p><b>Journal</b></p> <ul style="list-style-type: none"> <li>■ Journal système</li> <li>■ Avertissement système</li> <li>■ Erreur système</li> <li>■ Journal de serveur</li> <li>■ Avertissement serveur</li> <li>■ Erreur serveur</li> <li>■ Journal système+serveur</li> <li>■ Avertissement système+serveur</li> <li>■ Erreur système+serveur</li> </ul> <p><b>Réseau</b></p> <ul style="list-style-type: none"> <li>■ HTTP post</li> <li>■ HTTP get</li> </ul>		Palette des workflows <b>De base, Journal et Réseau</b>

## Propriétés de l'élément du schéma

Les éléments du schéma possèdent des propriétés que vous pouvez définir et modifier dans l'onglet **Schéma** de la palette des workflows.

### Modifier les propriétés globales d'un élément de schéma

Vous pouvez définir les propriétés globales d'un élément de schéma dans l'onglet Infos de l'élément.

#### Conditions préalables

Vérifiez que l'onglet **Schéma** de l'éditeur de workflow contient les éléments.

#### Procédure

- 1 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.

- 2 Sélectionnez un élément à modifier en cliquant sur l'icône **Modifier** (✎).

Une boîte de dialogue répertoriant les propriétés de l'élément apparaît.

- 3 Cliquez sur l'onglet **Infos**.

- 4 Entrez le nom de l'élément de schéma dans la zone de texte **Nom**.

Il s'agit du nom qui apparaît dans l'élément de schéma du diagramme du schéma de workflow.

- 5 Sélectionnez une description dans le menu déroulant **Interaction**.

La propriété **Interaction** vous permet d'effectuer une sélection entre les descriptions standard des interactions de cet élément avec les objets extérieurs au workflow. Cette propriété revêt uniquement un caractère informatif.

- 6 (Facultatif) Entrez une description de l'état du groupe dans la zone de texte **État du groupe**.

La propriété **État du groupe** est une description rapide de la fonction de l'élément. Lorsqu'un workflow s'exécute, le jeton de workflow affiche l'État du groupe des éléments en cours d'exécution. Cette fonction permet de suivre l'état du workflow.

- 7 (Facultatif) Dans la zone de texte **Description**, saisissez la description de l'élément de schéma.

## Onglets des propriétés d'éléments du schéma

Vous accédez aux propriétés d'un élément du schéma en cliquant sur un élément que vous avez fait glisser dans le schéma du workflow. Les propriétés de l'élément apparaîtront dans les onglets situés au bas de l'éditeur de workflow.

Des éléments différents du schéma possèdent des onglets de propriétés différents.

**Tableau 1-4. Onglets des propriétés par élément du schéma**

Onglet de propriétés de l'élément du schéma	Description	S'applique au type d'élément du schéma
<b>Attributs</b>	Attributs que les éléments réclament à une source extérieure, telle que l'utilisateur, un événement ou un temporisateur. Les attributs peuvent être une limite de délai d'expiration, un horodatage, un déclencheur ou les informations d'identification de l'utilisateur.	<ul style="list-style-type: none"> <li>■ Interaction utilisateur</li> <li>■ Événement d'attente</li> <li>■ Minuterie d'attente</li> </ul>
<b>Décision</b>	Définit l'instruction de décision. Soit le paramètre d'entrée que l'élément Décision reçoit concorde avec l'instruction de décision, soit il ne concorde pas, ce qui entraîne deux suites possibles.	Décision

Tableau 1-4. Onglets des propriétés par élément du schéma (suite)

Onglet de propriétés de l'élément du schéma	Description	S'applique au type d'élément du schéma
<b>Fin de workflow</b>	Arrête le workflow, soit parce qu'il a abouti, soit parce qu'il est tombé sur une erreur et a renvoyé une exception.	<ul style="list-style-type: none"> <li>■ Fin</li> <li>■ Exception</li> </ul>
<b>Exception</b>	Manière dont ce schéma se comporte en cas d'exception.	<ul style="list-style-type: none"> <li>■ Action</li> <li>■ Workflow asynchrone</li> <li>■ Exception</li> <li>■ Workflows intégrés</li> <li>■ Tâche prédéfinie</li> <li>■ Workflow planifié</li> <li>■ Tâche inscriptible</li> <li>■ Interaction utilisateur</li> <li>■ Événement d'attente</li> <li>■ Minuterie d'attente</li> <li>■ Workflow</li> </ul>
<b>Saisies externes</b>	Paramètres d'entrée que l'utilisateur doit renseigner à un certain moment de l'exécution du workflow.	Interaction utilisateur
<b>IN</b>	Liaison IN de cet élément. La liaison IN définit la façon dont l'élément du schéma reçoit une entrée de l'élément qui le précède dans le workflow.	<ul style="list-style-type: none"> <li>■ Action</li> <li>■ Workflow asynchrone</li> <li>■ Décision personnalisée</li> <li>■ Tâche prédéfinie</li> <li>■ Workflow planifié</li> <li>■ Tâche inscriptible</li> <li>■ Workflow</li> </ul>
<b>Infos</b>	Propriétés générales et description de l'élément du schéma. Les informations que l'onglet <b>Infos</b> montre dépendent du type d'élément du schéma.	<ul style="list-style-type: none"> <li>■ Action</li> <li>■ Workflow asynchrone</li> <li>■ Décision personnalisée</li> <li>■ Décision</li> <li>■ Workflows intégrés</li> <li>■ Remarque</li> <li>■ Tâche prédéfinie</li> <li>■ Workflow planifié</li> <li>■ Tâche inscriptible</li> <li>■ Interaction utilisateur</li> <li>■ Événement d'attente</li> <li>■ Minuterie d'attente</li> <li>■ Workflow</li> </ul>

Tableau 1-4. Onglets des propriétés par élément du schéma (suite)

Onglet de propriétés de l'élément du schéma	Description	S'applique au type d'élément du schéma
OUT	Liaison OUT de cet élément. La liaison OUT définit la manière dont l'élément du schéma établit une liaison entre les paramètres de sortie et les attributs du workflow ou ses paramètres de sortie.	<ul style="list-style-type: none"> <li>■ Action</li> <li>■ Workflow asynchrone</li> <li>■ Tâche prédéfinie</li> <li>■ Workflow planifié</li> <li>■ Tâche inscriptible</li> <li>■ Workflow</li> </ul>
Présentation	Définit l'agencement de la boîte de dialogue des paramètres d'entrée que l'utilisateur voit dans le cas où le workflow nécessite une saisie par l'utilisateur au cours de son exécution.	Interaction utilisateur
Scripts	Montre la fonction JavaScript qui définit le comportement de cet élément du schéma. Pour les éléments Workflow asynchrone, Workflow planifié et Action, ces scripts sont en lecture seule. Pour les éléments Tâche inscriptible et Décision personnalisée, le JavaScript peut être modifié dans cet onglet.	<ul style="list-style-type: none"> <li>■ Action</li> <li>■ Workflow asynchrone</li> <li>■ Décision personnalisée</li> <li>■ Tâche prédéfinie</li> <li>■ Workflow planifié</li> <li>■ Tâche inscriptible</li> </ul>
Liaison visuelle	Montre une représentation graphique du mode de liaison entre les paramètres et attributs de cet élément du schéma et les paramètres et attributs des éléments précédents et suivants dans le workflow. C'est une autre représentation des liaisons IN et OUT de l'élément.	<ul style="list-style-type: none"> <li>■ Action</li> <li>■ Workflow asynchrone</li> <li>■ Tâche prédéfinie</li> <li>■ Workflow planifié</li> <li>■ Tâche inscriptible</li> <li>■ Workflow</li> </ul>
Workflows	Sélectionne les workflows à intégrer.	Workflows intégrés

## Liens et liaisons

Les liens entre les éléments déterminent l'enchaînement logique du workflow. Les liaisons renseignent les éléments avec des données issues d'autres éléments par la liaison des paramètres d'entrée et de sortie aux attributs du workflow.

Pour comprendre les notions de lien et de liaison, vous devez comprendre la différence entre l'enchaînement logique du workflow et l'enchaînement des données du workflow.

## Enchaînement logique d'un workflow

L'enchaînement logique d'un workflow désigne la progression du workflow d'un élément vers le suivant au sein du schéma au fur et à mesure que le workflow s'exécute. On définit l'enchaînement logique du workflow en liant les éléments au sein du schéma.

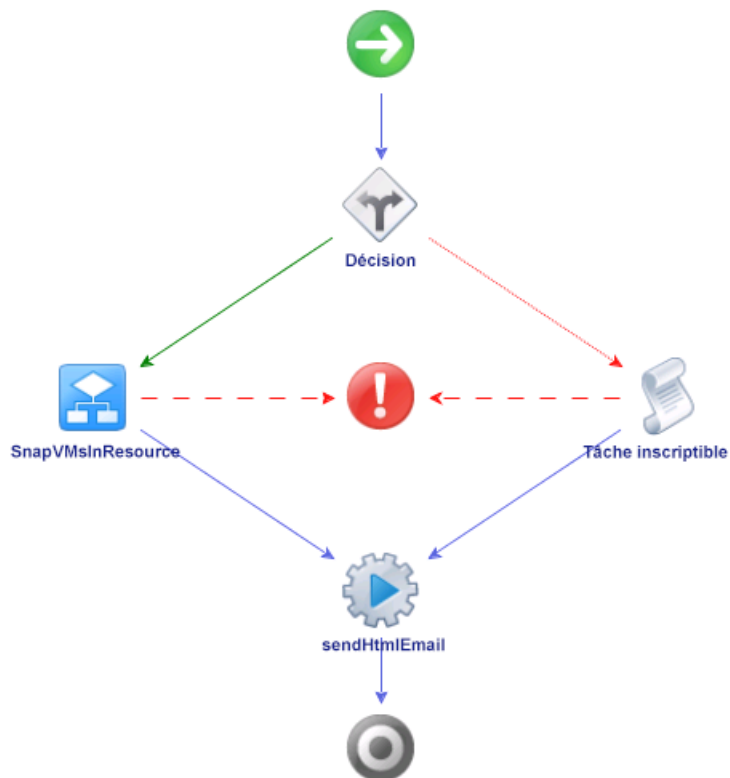
Le parcours standard désigne le parcours que suit le workflow dans l'enchaînement logique si tous les éléments s'exécutent comme prévu. Le parcours en cas d'exception désigne le parcours suivi que le workflow dans l'enchaînement logique si un élément ne s'exécute pas comme prévu.

Des styles de flèche différents dans le schéma signalent les parcours différents que le workflow peut suivre dans l'enchaînement logique.

- Une flèche bleue signale le parcours standard que suit le workflow d'élément en élément.
- Une flèche verte signale le parcours que suit le workflow si un élément de décision booléen renvoie `true`
- Une flèche pointillée rouge signale le parcours que suit le workflow si un élément de décision booléen renvoie `false`
- Une flèche discontinue rouge signale le parcours en cas d'exception que suit le workflow si un de ses éléments ne s'exécute pas correctement.

La figure suivante montre un exemple de schéma qui illustre les différents parcours que les workflows peuvent suivre.

**Figure 1-1. Différents parcours de workflow dans l'enchaînement logique du workflow**



Cet exemple de workflow peut suivre les parcours suivants dans son enchaînement logique.

- Parcours standard, résultat de la décision `true`, sans exception.
  - a L'élément de décision renvoie `true`.
  - b L'exécution du workflow `SnapVMsInResourcePool` est réussie.
  - c L'exécution de l'action `sendHtmlEmail` est réussie.
  - d Le workflow a bien abouti à l'état `completed`.
- Parcours standard, résultat de décision `false`, sans exception.
  - a L'élément de décision renvoie `false`.
  - b L'opération que l'élément de tâche inscriptible définit s'exécute avec succès.
  - c L'exécution de l'action `sendHtmlEmail` est réussie.
  - d Le workflow a bien abouti à l'état `completed`.
- Résultat de décision `true`, avec exception.
  - a L'élément de décision renvoie `true`.
  - b Le workflow `SnapVMsInResourcePool` tombe sur une erreur.
  - c Le workflow renvoie une exception et s'interrompt à l'état `failed`.
- Résultat de décision `false`, avec exception.
  - a L'élément de décision renvoie `false`.
  - b L'opération que l'élément de tâche Inscriptible définit tombe sur une erreur.
  - c Le workflow renvoie une exception et s'interrompt à l'état `failed`.

## Liens des éléments

Les liens connectent les éléments de schéma et définissent la progression du workflow d'un élément vers le suivant.

Généralement, les éléments ne peuvent présenter qu'un seul lien sortant vers un autre élément du workflow et un lien d'exception vers un élément qui définit le comportement en cas d'exception. Le lien sortant définit le parcours standard du workflow. Le lien d'exception définit le parcours du workflow en cas d'exception. Dans la plupart des cas, un élément de schéma peut recevoir des liens de parcours standard en provenance de plusieurs éléments.

Les éléments suivants font figure d'exceptions.

- L'élément Début de workflow ne peut recevoir aucun lien entrant et ne présente aucun lien d'exception.
- Les éléments d'exception peuvent recevoir plusieurs liens d'exception entrants et ne présentent aucun lien sortant ou d'exception.

- Les éléments Décision présentent deux liens sortants qui définissent le parcours que le workflow suivra selon si le résultat de la décision est `true` ou `false`. Les décisions ne présentent aucun lien d'exception.
- Les éléments Fin de workflow ne peuvent présenter aucun lien sortant ou d'exception.

## Créer des liens de parcours standard

Les liens de parcours standard déterminent le mode normal d'exécution du workflow.

Lorsque vous reliez un élément à un autre, vous les reliez toujours selon leur ordre d'exécution dans le workflow. Vous commencez toujours par l'élément qui s'exécute en premier pour créer un lien entre deux éléments.

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Vérifiez que l'onglet **Schéma** de l'éditeur de workflow contient les éléments.

### Procédure

- 1 Positionnez le curseur sur l'élément que vous souhaitez connecter à un autre élément.  
Un flèche bleue et une flèche rouge apparaissent à droite de l'élément.
- 2 Positionnez le curseur sur la flèche bleue.  
La flèche bleue s'agrandit.
- 3 Cliquez avec le bouton gauche sur la flèche bleue, maintenez le bouton gauche de la souris enfoncé, puis déplacez le curseur jusqu'à l'élément cible.  
Une flèche bleue apparaît entre les deux éléments et un rectangle vert apparaît autour de l'élément cible.
- 4 Relâchez le bouton gauche de la souris.  
La flèche bleue reste entre les deux éléments.

Un parcours standard relie désormais les éléments.

### Étape suivante

Les éléments sont reliés mais vous n'avez pas défini l'enchaînement de données. Vous devez définir les liaisons Entrée et Sortie pour relier les données entrantes et sortantes aux attributs du workflow.

## Enchaînement de données d'un workflow

L'enchaînement de données d'un workflow correspond à la manière dont les paramètres d'entrée et de sortie d'un élément de workflow se lient à des attributs au cours de l'exécution de ce workflow. Vous pouvez définir l'enchaînement de données d'un workflow à l'aide des liaisons d'éléments de schémas.

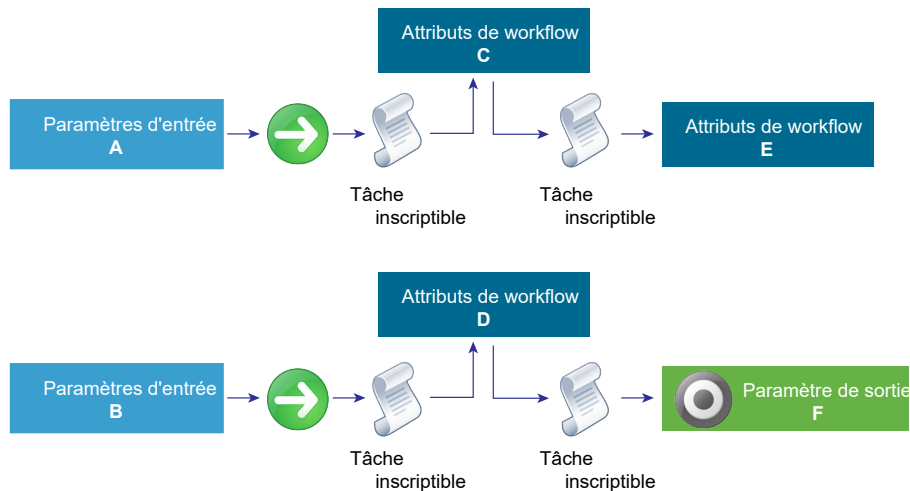


Lorsqu'un élément du schéma de workflow s'exécute, il doit disposer de données sous forme de paramètres d'entrée. Il récupère les données pour ses paramètres d'entrée en se connectant à un attribut de workflow que vous définissez à la création du workflow ou en se connectant à un attribut défini par un élément précédent du workflow lors de son exécution.

L'élément traite les données, éventuellement en les transformant, et génère les résultats de son exécution sous forme de paramètres de sortie. L'élément relie ses paramètres de sortie à de nouveaux attributs de workflow qu'il crée. Les autres éléments du schéma peuvent se connecter à ces nouveaux attributs de workflow pour en faire des paramètres d'entrée. Le workflow peut générer les attributs en tant que paramètres de sortie à la fin de son exécution.

La figure suivante illustre un workflow très simple. Les flèches bleues représentent les liens entre les éléments et l'enchaînement logique du workflow. Les lignes rouges illustrent l'enchaînement de données du workflow.

**Figure 1-2. Exemple d'enchaînement de données de workflow**



Les données s'enchaînent dans le workflow de la manière suivante.

- 1 Le workflow démarre avec les paramètres d'entrée a et b.
- 2 Le premier élément traite le paramètre a et relie le résultat de ce traitement à l'attribut de workflow c.
- 3 Le premier élément traite le paramètre b et relie le résultat de ce traitement à l'attribut de workflow d.
- 4 Le deuxième élément récupère l'attribut de workflow c en tant que paramètre d'entrée, le traite, et relie le paramètre de sortie qui en découle à l'attribut de workflow e.
- 5 Le deuxième élément récupère l'attribut de workflow d en tant que paramètre d'entrée, le traite, et génère le paramètre de sortie f.
- 6 Le workflow se termine et génère l'attribut de workflow f en tant que paramètre de sortie et résultat de son exécution.

## Liaisons des éléments

Vous devez relier tous les paramètres d'entrée et de sortie des éléments de workflow à des attributs de workflow. Les liaisons permettent de définir les données des éléments, ainsi que leur comportement face aux exceptions et lors de la sortie. Les liens définissent l'enchaînement logique du workflow tandis que les liaisons définissent l'enchaînement de données.

Veillez à définir la liaison de l'élément pour déterminer les données dans un élément, générer les paramètres de sortie de l'élément après son traitement et gérer toutes les erreurs potentielles lors de l'exécution de cet élément.

<b>Liaisons IN</b>	Définissez les données entrantes d'un élément de schéma. Vous devez relier les paramètres d'entrée locaux de l'élément aux attributs source du workflow. L'onglet <b>Entrée</b> répertorie les paramètres d'entrée de l'élément dans la colonne Paramètre local. L'onglet <b>Entrée</b> répertorie les attributs de workflow auquel le paramètre local se relie dans la colonne Paramètre source. L'onglet affiche également le type de paramètre et une description de ce paramètre.
<b>Liaisons OUT</b>	Modifiez les attributs du workflow et générez des paramètres de sortie lorsqu'un élément a terminé son exécution. L'onglet <b>Sortie</b> répertorie les paramètres de sortie de l'élément dans la colonne Paramètre local. L'onglet <b>Sortie</b> répertorie les attributs de workflow auquel le paramètre local se relie dans la colonne Paramètre source. L'onglet affiche également le type de paramètre et une description de ce paramètre.
<b>Liaisons d'exception</b>	Créez un lien vers les gestionnaires d'exceptions si l'élément rencontre une exception au cours de son exécution.

Les liaisons **IN** lisent les valeurs dans le paramètre source relié. Les liaisons **OUT** rédigent les valeurs dans le paramètre source relié.

Vous devez utiliser les liaisons **IN** pour relier chaque attribut ou paramètre d'entrée de votre élément de schéma à un attribut de workflow. Si l'élément modifie les valeurs des paramètres d'entrée qu'il reçoit lors de son exécution, vous devez les relier à un attribut de workflow à l'aide d'une liaison **OUT**. La liaison des paramètres de sortie de l'élément à des éléments de workflow permet aux autres éléments qui suivent dans le schéma de récupérer ces paramètres de sortie en tant que paramètres d'entrée.

Lors de la création de workflows, une erreur fréquente est de ne pas relier les valeurs des paramètres de sortie pour répercuter les modifications créées par l'élément dans les attributs du workflow.

---

**Important** Lorsque vous ajoutez un élément qui demande des paramètres d'entrée et de sortie d'un type déjà défini dans le workflow, Orchestrator établit des liaisons vers ces paramètres. Vous devez vérifier que les paramètres reliés par Orchestrator sont corrects, au cas où le workflow aurait défini des paramètres ne correspondant pas au type auquel l'élément peut se relier.

---

## Définir les liaisons des éléments

Une fois que vous avez relié les éléments pour créer l'enchaînement logique du workflow, vous devez définir les liaisons des éléments pour déterminer comment chaque élément traite les données qu'il reçoit et génère.

### Conditions préalables

Vérifiez que vous disposez d'un schéma de workflow dans l'onglet **Schéma** de l'éditeur de workflow, et que vous avez créé des liens entre les éléments.

### Procédure

- 1 Cliquez sur l'icône **Modifier** (✎) de l'élément sur lequel les liaisons doivent être définies.

Une boîte de dialogue répertoriant les propriétés de l'élément apparaît.

- 2 Cliquez sur l'onglet **Entrée**.

Le contenu de l'onglet **Entrée** varie en fonction du type d'élément que vous avez sélectionné.

- Si vous avez sélectionné une tâche, un workflow ou un élément d'action prédéfini, l'onglet **Entrée** répertorie les paramètres d'entrée locaux possibles pour ce type d'élément sans définir la liaison.
- Si vous avez sélectionné un autre type d'élément, vous pouvez effectuer votre sélection dans une liste de paramètres d'entrée et d'attributs que vous avez déjà définis pour ce workflow en cliquant-droit dans l'onglet **Entrée**, puis en sélectionnant **Relier à un paramètre/attribut de workflow**.
- Si l'attribut nécessaire n'existe pas encore, vous pouvez le créer en cliquant-droit dans l'onglet **Entrée** et en sélectionnant **Relier à un paramètre/attribut de workflow > Créer un paramètre/attribut de workflow**.

- 3 Si un paramètre approprié existe déjà, sélectionnez un paramètre d'entrée à relier, puis cliquez sur le bouton **Non défini** de la zone de texte **Paramètres source**.

Une liste des paramètres source et attributs possibles pour la liaison apparaît.

- 4 Dans la liste proposée, sélectionnez un paramètre source à relier au paramètre d'entrée local.
- 5 (Facultatif) Si vous n'avez pas défini le paramètre source de liaison, vous pouvez le créer en cliquant sur le lien **Créer un paramètre/attribut de workflow** de la boîte de dialogue de sélection des paramètres.

- 6 Cliquez sur l'onglet **Sortie**.

Le contenu de l'onglet **Sortie** varie en fonction du type d'élément que vous avez sélectionné.

- Si vous avez sélectionné une tâche, un workflow ou un élément d'action prédéfini, l'onglet **Sortie** répertorie les paramètres de sortie locaux possibles pour ce type d'élément sans définir la liaison.
- Si vous avez sélectionné un autre type d'élément, vous pouvez effectuer votre sélection dans une liste de paramètres de sortie et d'attributs que vous avez déjà définis pour ce workflow en cliquant-droit dans l'onglet **Entrée**, puis en sélectionnant **Relier à un paramètre/attribut de workflow**.

- Si l'attribut nécessaire n'existe pas, vous pouvez le créer en cliquant-droit dans l'onglet **Entrée** et en sélectionnant **Relier à un paramètre/attribut de workflow > Créer un paramètre/attribut de workflow**.

- 7 Sélectionnez un paramètre à relier.
- 8 Cliquez sur le bouton **Paramètre source > Non défini**.
- 9 Sélectionnez un paramètre source à relier au paramètre d'entrée.
- 10 (Facultatif) Si vous n'avez pas défini le paramètre source de liaison, vous pouvez le créer en cliquant sur le lien **Créer un paramètre/attribut de workflow** de la boîte de dialogue de sélection des paramètres.

Vous avez défini les paramètres que l'élément reçoit et les paramètres de sortie qu'il génère, puis vous les avez reliés aux attributs et paramètres du workflow.

### Étape suivante

Vous pouvez créer des embranchements sur le parcours du workflow en définissant des décisions.

## Décisions

Les workflows peuvent implémenter des fonctions de décision qui définissent des suites d'actions différentes basées sur une instruction booléenne `true` ou `false`.

Les décisions constituent des embranchements dans le workflow. Les décisions de workflows s'appuient sur les entrées que vous avez saisies ou qui proviennent d'autres workflows, d'applications ou de l'environnement dans lequel le workflow s'exécute. La valeur du paramètre d'entrée que l'élément Décision reçoit détermine la branche qui sera empruntée par le workflow. Par exemple, une décision de workflow peut recevoir l'état de mise sous tension d'une machine virtuelle donnée en tant qu'entrée. Si la machine virtuelle est sous tension, le workflow emprunte un chemin déterminé de son enchaînement logique. Si la machine virtuelle est hors tension, le workflow emprunte un autre parcours.

Les décisions sont toujours des fonctions booléennes. Les seuls résultats possibles de chaque décision sont `true` ou `false`.

### Décisions personnalisées

Les décisions personnalisées diffèrent des décisions standard car vous devez définir leur instruction dans un script. Les décisions personnalisées renvoient `true` ou `false` en fonction de l'instruction que vous avez définie, comme illustré par l'exemple suivant.

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

## Créer des liens d'éléments Décision

Les éléments Décision diffèrent des autres éléments d'un workflow. Leurs seuls paramètres d'entrée sont `true` ou `false`. Les éléments Décision ne présentent aucun lien vers des exceptions.

### Conditions préalables

Vérifiez que l'onglet **Schéma** de l'éditeur de workflow contient des éléments, notamment au moins un élément Décision qui n'est pas lié à d'autres éléments.

### Procédure

- 1 Positionnez le curseur de la souris sur un élément Décision pour le lier à deux autres éléments définissant deux parcours possibles du workflow.  
Un flèche bleue ou rouge apparaît à droite de l'élément.  
  
2 Positionnez le curseur sur la flèche bleue et, tout en maintenant le bouton gauche de la souris enfoncé, déplacez le curseur jusqu'à l'élément cible.  
Une flèche verte apparaît entre les deux éléments et l'élément cible devient vert. La flèche verte représente le parcours `true` emprunté par le workflow si le paramètre d'entrée ou l'attribut provenant de l'élément Décision correspond à l'instruction de décision.  
  
3 Relâchez le bouton gauche de la souris.  
La flèche verte reste entre les deux éléments. Vous avez défini le parcours que le workflow empruntera lorsque l'élément Décision recevra la valeur attendue.  
  
4 Positionnez le curseur sur l'élément Décision, maintenez le bouton gauche de la souris enfoncé, puis déplacez le curseur jusqu'à l'élément cible.  
Une flèche en pointillés rouges apparaît entre les deux éléments et l'élément cible devient vert. La flèche rouge représente le parcours `false` emprunté par le workflow si le paramètre d'entrée ou l'attribut provenant de l'élément Décision ne correspond pas à l'instruction de décision.  
  
5 Relâchez le bouton gauche de la souris.  
La flèche en pointillés rouges reste entre les deux éléments. Vous avez défini le parcours que le workflow empruntera si l'élément Décision ne reçoit pas la valeur attendue.

Vous avez défini les parcours `true` ou `false` que le workflow empruntera selon le paramètre d'entrée ou l'attribut reçu par l'élément Décision.

### Étape suivante

Définissez maintenant l'instruction de décision. Reportez-vous à [Créer des branches de workflow à l'aide des décisions](#).

## Supprimer un élément de décision lié

Lorsque vous supprimez un élément de décision lié à partir d'un schéma de workflow, vous devez indiquer les chemins d'accès de workflow à supprimer.

## Conditions préalables

Vérifiez que l'onglet **Schéma** de l'éditeur de workflow contient les éléments, notamment au moins un élément de décision avec les chemins d'accès corrects et incorrects.

## Procédure

- 1 Sélectionnez l'élément de décision, puis appuyez sur Supprimer.

Une boîte de dialogue s'affiche avec les options disponibles.

- 2 Sélectionnez la branche de décision à supprimer.

Option	Description
<b>Branche de réussite</b>	L'élément de décision et l'ensemble des éléments qui suivent le chemin d'accès de décision correct sont supprimés du schéma de workflow.
<b>Branche d'échec</b>	L'élément de décision et l'ensemble des éléments qui suivent le chemin d'accès de décision incorrect sont supprimés du schéma de workflow.
<b>Les deux branches</b>	L'élément de décision et l'ensemble des éléments qui suivent les deux chemins d'accès de décision sont supprimés du schéma de workflow.
<b>Aucun</b>	Seul l'élément de décision et ses liens sont supprimés du schéma de workflow. Tous les éléments qui suivent les deux chemins d'accès de décision sont conservés dans le schéma de workflow.

- 3 Cliquez sur **OK**.

## Créer des branches de workflow à l'aide des décisions

Les éléments Décision sont des fonctions booléennes simples que vous utilisez pour créer des branches dans les workflows. Les éléments Décision déterminent si l'entrée reçue correspond à l'instruction de décision que vous avez déterminée. Selon la décision, le workflow continue sa progression sur l'un des deux parcours possibles.

## Conditions préalables

Avant de définir la décision, vérifiez qu'un élément Décision est bien relié à deux autres éléments du schéma dans l'éditeur de workflow.

## Procédure

- 1 Cliquez sur l'icône **Modifier** (✎) de l'élément Décision.

Une boîte de dialogue répertoriant les propriétés de l'élément Décision apparaît.

- 2 Cliquez sur l'onglet **Décision** dans la boîte de dialogue.

- 3 Cliquez sur le lien **Non défini (NULL)** pour sélectionner le paramètre d'entrée source pour cette décision.

Une boîte de dialogue répertoriant tous les attributs et paramètres d'entrée définis dans ce workflow apparaît.

- 4 Sélectionnez un paramètre d'entrée dans la liste en double-cliquant dessus.

5 Si vous n'avez pas défini le paramètre source de liaison, créez-le en cliquant sur le lien **Créer un attribut/paramètre dans le workflow** dans la boîte de dialogue de sélection des paramètres.

6 Sélectionnez une instruction de décision dans le menu déroulant.

Les instructions proposées par le menu sont contextuelles et diffèrent selon le type de paramètre d'entrée sélectionné.

7 Ajoutez une valeur à laquelle l'instruction de décision doit correspondre.

Selon le type d'entrée et l'instruction que vous sélectionnez, vous pouvez constater la présence d'un lien **Non défini (NULL)** dans la zone de texte de la valeur. Si vous cliquez sur ce lien, vous obtenez une sélection prédéfinie de valeurs. Sinon, par exemple pour les chaînes, il s'agit d'une zone de texte dans laquelle vous entrez une valeur.

Vous avez défini une instruction pour l'élément Décision. Lorsque l'élément Décision reçoit le paramètre d'entrée, il compare la valeur de ce paramètre d'entrée à la valeur de l'instruction et détermine si l'instruction est true ou false.

### Étape suivante

Vous devez maintenant définir la manière dont le workflow gère les exceptions.

## Gestion des exceptions

La gestion des exceptions récupère toutes les erreurs qui se produisent au cours de l'exécution d'un élément de schéma. La gestion des exceptions définit le comportement de l'élément de schéma lorsqu'une erreur se produit.

Hormis les éléments Décision de début et de fin, tous les éléments des workflows incluent un type de paramètre de sortie dont le seul rôle est de gérer les exceptions. Si un élément rencontre une erreur au cours de son exécution, il envoie un signal d'erreur au gestionnaire d'exceptions. Les gestionnaires d'exceptions récupèrent l'erreur et réagissent en fonction. Si les gestionnaires d'exceptions que vous avez définis ne peuvent pas gérer une erreur donnée, vous pouvez relier le paramètre de sortie d'exception de l'élément à un élément Exception qui achève l'exécution du workflow avec l'état échec.

Les exceptions agissent en tant que séquence `try` et `catch` au sein d'un élément de workflow. Si vous n'avez pas besoin de gérer une exception dans un élément, il ne vous est pas nécessaire de relier le paramètre de sortie d'exception de l'élément.

Le type de paramètre de sortie des exceptions est toujours un objet `errorCode`.




### Créer des liaisons d'exception

Les éléments peuvent définir des liaisons qui déterminent le comportement du workflow lorsqu'il rencontre une erreur dans cet élément.

#### Conditions préalables

Vérifiez que l'onglet **Schéma** de l'éditeur de workflow contient les éléments.

## Procédure

- 1 Positionnez le curseur sur l'élément pour lequel vous souhaitez définir une liaison d'exception.  
Un flèche rouge apparaît à droite de l'élément.
- 2 Positionnez le curseur sur la flèche rouge jusqu'à ce qu'elle s'agrandisse, maintenez le bouton gauche de la souris enfoncé, puis faites glisser la flèche rouge jusqu'à l'élément cible.  
Une flèche discontinue rouge relie les deux éléments. L'élément cible définit le comportement du workflow si l'élément auquel il est relié rencontre une erreur.
- 3 Cliquez sur l'icône **Modifier** () de l'élément assurant la liaison avec l'élément de gestion des exceptions.
- 4 Cliquez sur l'onglet **Exception** dans les onglets des propriétés d'éléments du schéma.
- 5 Pour définir la valeur **Liaison des exceptions de sortie**, cliquez sur **Non définie**.
  - Sélectionnez un paramètre à relier au paramètre de sortie des exceptions dans la boîte de dialogue de liaison des attributs d'exceptions, puis cliquez sur **Sélectionner**.
  - Cliquez sur **Créer un paramètre/attribut dans le workflow** afin de créer un paramètre de sortie des exceptions.
- 6 Cliquez sur l'élément cible qui définit le comportement de gestion des exceptions.
- 7 Cliquez sur l'onglet **Entrée** dans les onglets de propriétés des éléments du schéma.
- 8 Cliquez sur l'icône **Relier à un paramètre/à un attribut du workflow** ().  
La boîte de dialogue de sélection du paramètre d'entrée apparaît.
- 9 Sélectionnez le paramètre de sortie des exceptions, puis cliquez sur **Sélectionner**.
- 10 Cliquez sur l'onglet **Sortie** de l'élément de gestion des exceptions dans les onglets de propriétés des éléments du schéma.
- 11 Définissez le comportement de l'élément de gestion des exceptions.
  - Cliquez sur l'icône **Relier à un paramètre/à un attribut du workflow** () pour sélectionner un paramètre de sortie que l'élément de gestion des exceptions doit générer.
  - Cliquez sur l'onglet **Script** et utilisez JavaScript pour définir le comportement de l'élément de gestion des exceptions.

Vous avez défini la manière dont l'élément gère les exceptions.

## Étape suivante

Vous devez définir le mode d'obtention des paramètres d'entrée auprès des utilisateurs qui exécutent le workflow.



## Utilisation des gestionnaires d'erreurs

Vous pouvez utiliser un gestionnaire d'erreurs standard pour déterminer le comportement à adopter si une erreur se produit dans un élément de schéma de workflow spécifique. Vous pouvez utiliser un gestionnaire d'erreurs global pour déterminer le comportement à adopter si une erreur non détectée par le gestionnaire d'erreurs standard se produit.

### Ajouter un gestionnaire d'erreurs à un workflow

Vous pouvez définir la façon dont les erreurs d'un élément de workflow spécifique seront gérées au cours de l'exécution du workflow grâce à l'ajout d'un gestionnaire d'erreurs à cet élément. Seuls les éléments de workflow ne disposant pas d'un chemin d'accès aux erreurs déterminé peuvent se voir ajouter un gestionnaire d'erreurs.

---

**Important** Les workflows qui comportent un élément de **Gestion des erreurs** ne sont pas compatibles avec Orchestrator 5.5.x ou les versions antérieures.

---

#### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.
- Ajoutez des éléments au schéma de workflow.

#### Procédure

- 1 Faites glisser un élément de **Gestion des erreurs** vers l'élément approprié dans le schéma du workflow.

Une boîte de dialogue apparaît.

- 2 Dans le menu déroulant de la boîte de dialogue, sélectionnez le mode de gestion des erreurs.

Option	Description
Générer une exception	Lorsqu'une erreur se produit, une exception est générée. Vous pouvez modifier cette exception.
Appeler un workflow	Lorsqu'une erreur se produit, le workflow sélectionné s'exécute.
Script personnalisé	Lorsqu'une erreur se produit, un script personnalisé s'exécute.

- 3 Cliquez sur **Sélectionner**.

Vous avez ajouté un gestionnaire d'erreurs à un workflow. Lorsque le workflow atteint cet élément, il exécute l'action sélectionnée avant la fin de son exécution.

## Ajouter un gestionnaire d'erreurs global à un workflow

Vous pouvez définir la façon dont les erreurs non prises en charge par les gestionnaires d'erreur standard seront gérées au cours de l'exécution d'un workflow grâce à l'ajout d'un gestionnaire d'erreurs global au schéma du workflow. Vous pouvez ajouter un gestionnaire d'erreurs global à un schéma de workflow.

---

**Important** Les workflows qui comportent un élément **Gestionnaire d'erreurs par défaut** ne sont pas compatibles avec Orchestrator 5.5 ou les versions antérieures.

---

### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.
- Ajoutez des éléments au schéma de workflow.

### Procédure

- 1 Faites glisser un élément **Gestionnaire d'erreurs par défaut** dans le schéma du workflow.
- 2 (Facultatif) Ajoutez des éléments du schéma entre l'élément **Gestionnaire d'erreurs par défaut** et l'élément **Générer une exception** pour indiquer les modalités de gestion des erreurs globales du workflow.

Vous avez ajouté un gestionnaire d'erreurs global à un workflow. Lorsqu'une erreur non gérée par les gestionnaires d'erreurs standard du workflow se produit, le gestionnaire d'erreurs global exécute les actions indiquées avant de mettre fin à l'exécution du workflow.

## Éléments Foreach et types composites

Vous pouvez insérer un élément Foreach dans le workflow que vous développez afin d'exécuter un workflow secondaire qui couvre les tableaux de paramètres ou d'attributs. Pour améliorer la compréhension et la lecture du workflow, vous pouvez regrouper plusieurs paramètres de workflow de types différents et connectés logiquement dans un type unique appelé « type composite ».

### Utilisation des éléments Foreach

L'élément Foreach exécute un workflow secondaire de manière itérative sur un tableau de paramètres d'entrée ou d'attributs. Vous pouvez sélectionner les tableaux sur lesquels le workflow secondaire s'exécute et transmettre les valeurs des éléments de ces tableaux lors de l'exécution du workflow. Le workflow secondaire s'exécute autant de fois que le nombre d'éléments définis dans le tableau.

Si vous disposez d'un élément de configuration comportant un tableau d'attributs, vous pouvez exécuter un workflow qui couvre ces attributs dans un élément Foreach.

Par exemple, imaginez que vous disposez d'un dossier avec 10 machines virtuelles à renommer. Pour procéder à cette opération, vous devez insérer un élément Foreach dans un workflow, puis définir le workflow Renommer la machine virtuelle en tant que workflow secondaire dans l'élément. Le workflow Renommer la machine virtuelle implique deux paramètres d'entrée, une machine virtuelle et son nouveau nom. Vous pouvez promouvoir ces paramètres en tant qu'entrée du workflow en cours pour qu'ils

prennent la forme de tableaux couverts par le workflow Renommer la machine virtuelle. Lorsque vous exécutez votre workflow, vous pouvez indiquer 10 machines virtuelles dans le dossier avec leur nouveau nom. À chacune de ses exécutions, le workflow prend un élément du tableau des machines virtuelles et un élément du tableau des nouveaux noms des machines virtuelles.

## Utilisation de types composites

Un type composite est un groupe de plusieurs paramètres d'entrée ou attributs connectés logiquement, mais appartenant à des types différents. Vous pouvez utiliser un élément Foreach pour relier un groupe de paramètres sous une valeur composite. Ainsi, l'élément Foreach prend toutes les valeurs des paramètres regroupés en une seule fois dans chaque exécution ultérieure du workflow.

Par exemple, imaginez que vous êtes sur le point de renommer une machine virtuelle. Vous avez besoin de l'objet et du nouveau nom de la machine virtuelle. Si vous devez renommer plusieurs machines virtuelles, vous devez disposer de deux tableaux : un pour les machines virtuelles et un pour leurs noms. Ces deux tableaux ne sont pas connectés de manière explicite. Un type composite vous permet de disposer d'un seul tableau dans lequel chaque élément comporte la machine virtuelle et son nouveau nom. De cette manière, même en cas de valeurs multiples, la connexion entre ces deux paramètres est explicite au lieu d'être créée de manière implicite par le schéma de workflow.

---

**Note** Vous ne pouvez pas exécuter un workflow qui comporte des types composites du vSphere Web Client.

---

## Définir un élément Foreach

Si vous souhaitez exécuter un workflow secondaire plusieurs fois en modifiant les valeurs de ses paramètres ou attributs à chaque fois, vous pouvez insérer un élément Foreach dans le workflow parent.

Lorsque vous insérez un élément Foreach, vous devez sélectionner au moins un tableau sur lequel l'élément Foreach agit. Un élément de tableau peut présenter différentes valeurs pour chaque nouvelle exécution du workflow.

Si le workflow présente des paramètres de sortie, vous devez sélectionner les paramètres de sortie de l'élément Foreach dans lequel il est prévu d'accumuler les sorties de workflow afin que le workflow secondaire puisse les couvrir.

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Dans l'éditeur de workflows, sélectionnez l'onglet **Schéma**.
- 2 Dans le menu **Générique**, faites glisser un élément Foreach dans le schéma de workflow.
- 3 Sélectionnez un workflow dans la boîte de dialogue Sélectionneur.

La notification suivante apparaît dans la partie supérieure du volet du schéma.

Souhaitez-vous ajouter les paramètres de l'activité en tant qu'entrée ou sortie au workflow en cours ?

- 4 Dans la notification, cliquez sur **Configuration**.

Une fenêtre contextuelle s'affiche avec les options disponibles.

- 5 Sélectionnez le type de mappage pour chaque paramètre d'entrée.

Option	Description
Entrée	L'argument est mappé vers un paramètre de workflow d'entrée.
Ignorer	L'argument est mappé vers une valeur NULL.
Valeur	L'argument est mappé vers un attribut avec une valeur que vous pouvez définir dans la colonne Valeur.

- 6 Sélectionnez le type de mappage pour chaque paramètre de sortie.

Option	Description
Output	L'argument est mappé vers un paramètre de workflow de sortie.
Ignorer	L'argument est mappé vers une valeur NULL.
Variable locale	L'argument est mappé vers un attribut.

- 7 Cliquez sur **Promouvoir**.

- 8 Cliquez avec le bouton droit sur l'élément Foreach et sélectionnez **Synchroniser > Synchroniser la présentation**.

Une boîte de dialogue de confirmation apparaît.

- 9 Cliquez sur **OK** pour répercuter la présentation de l'élément Foreach sur le workflow en cours.

Une boîte de dialogue affiche des informations sur le résultat de l'opération.

- 10 Dans l'onglet **Entrées**, vérifiez que les paramètres du workflow secondaire sont ajoutés en tant qu'éléments du tableau des types.

- 11 Dans l'onglet **Sorties**, vérifiez que les paramètres du workflow secondaire sont ajoutés en tant qu'éléments du tableau des types.

Vous avez défini un élément Foreach dans votre workflow. L'élément Foreach exécute un workflow qui considère comme un paramètre chaque élément du tableau des paramètres ou attributs que vous avez défini.

Pour ce qui concerne les paramètres ou attributs qui ne sont pas définis dans des tableaux, le workflow reprend la même valeur à chaque exécution.

## Exemple : Renommer les machines virtuelles à l'aide d'un élément Foreach

Vous pouvez utiliser un élément Foreach pour renommer plusieurs machines virtuelles en une seule fois. Vous devez insérer un élément Foreach dans un workflow et positionner les paramètres `vm` et `newName` en tant qu'entrées du workflow en cours. Ainsi, lorsque vous exécutez le workflow, vous indiquez les machines virtuelles à renommer et les nouveaux noms des machines virtuelles. Les machines virtuelles sont intégrées en tant qu'éléments dans le tableau que vous avez créé pour le paramètre `vm`. Les nouveaux noms des machines virtuelles sont intégrés au tableau que vous avez créé pour le paramètre `newName`.

## Définir un type composite dans un élément Foreach

Vous pouvez regrouper plusieurs paramètres de workflow connectés logiquement dans un nouveau type appelé « type composite ». Vous pouvez utiliser un élément Foreach pour regrouper des paramètres sous une valeur composite afin de relier plusieurs tableaux de paramètres dans un seul tableau.

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Vérifiez que vous disposez d'un élément Foreach dans votre workflow.

### Procédure

- 1 Sélectionnez l'onglet **ENTRÉE** ou **SORTIE** de l'élément Foreach.
- 2 Sélectionnez un paramètre local que vous souhaitez regrouper avec les autres paramètres locaux dans un type composite.
- 3 Cliquez sur **Relier un groupe de paramètres sous une valeur composite** dans la partie supérieure de l'onglet **ENTRÉE** ou **SORTIE**.
- 4 Dans le volet Connexions, sélectionnez les paramètres que vous souhaitez regrouper sous un type composite.
- 5 Sélectionnez **Lier en tant qu'itérateur**.

Vous avez défini l'élément Foreach à itérer pour un tableau de type composite.

- 6 Cliquez sur **Accepter**.

Vous avez défini un type composite et vous vous êtes assuré que le workflow couvrirait le tableau de ce type composite. Les paramètres regroupés sous le type composite sont appelés `composite_type_name.parameter_name`. Par exemple, si vous créez un type composite `snapshots`, les paramètres regroupés dans ce type peuvent être `snapshots.vm[in-parameter]` ou `snapshots.name[in-parameter]`. Tous les éléments du tableau du type composite contiennent une instance unique de chaque paramètre regroupé dans ce type composite.

## Exemple : Renommer les machines virtuelles

Imaginez que vous souhaitez renommer 10 machines virtuelles en une seule fois. Pour ce faire, vous insérez un élément Foreach dans un workflow, puis vous sélectionnez le workflow Renommer la machine virtuelle dans l'élément. Vous créez un type composite afin de connecter les paramètres vm et newName de façon explicite. Vous connectez le type composite en tant qu'itérateur, et vous créez ainsi un tableau unique contenant le paramètre vm et le paramètre newName.

## Ajouter une activité de commutation à un workflow

Vous pouvez ajouter une activité de commutation de base à un schéma de workflow pour définir les cas d'utilisation en fonction des attributs ou paramètres du workflow.

Chaque activité de commutation peut présenter plusieurs cas de commutation. Chaque cas de commutation est défini par une condition relative à un attribut ou à un paramètre. Si la condition est remplie, l'exécution du workflow passe à l'élément de workflow correspondant que vous indiquez. Si aucune des conditions définies n'est remplie, le workflow exécute des commutations sur un élément par défaut que vous indiquez.

---

**Important** Les workflows qui comportent un élément de **Commutation** ne sont pas compatibles avec Orchestrator 5.5 ou les versions antérieures.

---

### Conditions préalables

Vérifiez que l'onglet **Schéma** de l'éditeur de workflow contient les éléments.

### Procédure

- 1 Faites glisser un élément de **Commutation** vers l'élément approprié dans le schéma du workflow.
- 2 Cliquez sur l'icône **Modifier** (✎) de l'élément de **Commutation**.
- 3 Dans l'onglet **Cas**, ajoutez ou supprimez des cas de commutation.  
Vous pouvez modifier la priorité des cas de commutation.
- 4 Définissez la condition pour chaque cas de commutation.
- 5 Sélectionnez l'élément de workflow correspondant à chaque cas de commutation.
- 6 Sélectionnez l'élément de workflow par défaut pour la commutation.
- 7 Cliquez sur **Fermer**.
- 8 Cliquez sur **Enregistrer**.

Vous avez défini les conditions de cas de commutation et les chemins de workflow.

## Développement de plug-ins

Orchestrator permet une intégration à des solutions de gestion et d'administration par le biais de son architecture ouverte de plug-ins. Vous utilisez le client Orchestrator pour exécuter et créer des workflows de plug-ins, et accéder à l'API des plug-ins.

## Présentation des plug-ins

Les plug-ins Orchestrator doivent inclure un jeu standard de composants et se conformer à une architecture standard. Le respect de ces pratiques vous aidera à créer des plug-ins pour le plus large éventail possible de technologies externes.

- **Structure d'un plug-in Orchestrator**

Les plug-ins Orchestrator possèdent une structure commune composée de différents types de couches servant à implémenter des fonctionnalités spécifiques.

- **Exposer une API externe à Orchestrator**

Vous exposez une API provenant d'un produit externe à la plateforme Orchestrator en créant un plug-in Orchestrator. Il vous est possible de créer un plug-in pour toute technologie qui expose une API pouvant être mappée dans les objets JavaScript utilisables par Orchestrator.

- **Composants d'un plug-in**

Les plug-ins sont composés d'un jeu standard de composants qui exposent les objets au sein de la technologie en plug-in à la plate-forme Orchestrator

- **Rôle du fichier vso.xml**

Vous utilisez le fichier `vso.xml` pour mapper les objets, les classes, les méthodes et les attributs de la technologie en plug-in vers les objets d'inventaire, les types de scripts, les classes de scripts, les méthodes de scripts et les attributs. Le fichier `vso.xml` définit également la configuration et le comportement de départ du plug-in.

- **Rôles de l'adaptateur de plug-in**

L'adaptateur de plug-in constitue le point d'entrée du plug-in dans le serveur Orchestrator. L'adaptateur de plug-in sert de banque de données à la technologie en plug-in du serveur Orchestrator, crée la fabrique de plug-in et gère les événements qui surviennent au sein de la technologie en plug-in.

- **Rôles de la fabrique de plug-in**

La fabrique de plug-in définit la façon dont Orchestrator recherche les objets au sein de la technologie en plug-in et réalise des opérations sur les objets.

- **Rôle des objets de l'outil de recherche**

Les objets de l'outil de recherche identifient et localisent des instances précises des types d'objets gérés de la technologie en plug-in. Orchestrator peut modifier les objets et interagir avec eux dès qu'il les trouve dans la technologie en plug-in. Pour ce faire, il exécute des workflows dans les objets de l'outil de recherche.

- **Rôle des objets de scripts**

Les objets de scripts sont des représentations JavaScript des objets provenant de la technologie en plug-in. Les objets de script des plug-ins apparaissent dans l'API JavaScript d'Orchestrator et vous pouvez les utiliser dans les éléments en script des workflows et des actions.

## ■ Rôle des gestionnaires d'événements

Les événements sont des modifications apportées aux états ou aux attributs des objets qu'Orchestrator trouve dans la technologie en plug-in. Orchestrator surveille les événements en implémentant des gestionnaires d'événement.

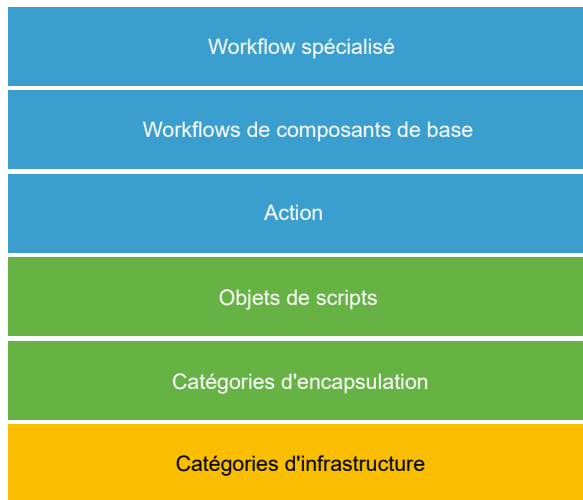
## Structure d'un plug-in Orchestrator

Les plug-ins Orchestrator possèdent une structure commune composée de différents types de couches servant à implémenter des fonctionnalités spécifiques.

Les trois couches inférieures d'un plug-in Orchestrator sont les classes d'infrastructure, les catégories d'encapsulation et les objets de script. Elles connectent la technologie en plug-in à Orchestrator.

Dans un plug-in Orchestrator, les parties visibles par l'utilisateur sont les trois couches supérieures, soit les actions, les composants de base et les workflows de spécialisés.

**Figure 1-3. Structure d'un plug-in Orchestrator**



### Classes d'infrastructure

Ensemble de classes qui connecte la technologie en plug-in à Orchestrator. Les classes d'infrastructure incluent les classes à implémenter en fonction de la définition du plug-in ; par exemple, la fabrique du plug-in, l'adaptateur du plug-in, etc. Les classes d'infrastructure incluent également les classes qui fournissent des fonctionnalités pour les tâches et les objets courants tels que les systèmes d'aide, la mise en cache, l'inventaire, etc.

### Classes d'encapsulation

Ensemble de classes qui adaptent le modèle d'objet de la technologie en plug-in au modèle d'objet que vous souhaitez exposer dans Orchestrator.

### Objets de scripts

Types d'objets JavaScript conférant un accès aux classes d'encapsulation, aux méthodes et aux attributs de la technologie en plug-in. Le fichier `vso.xml` vous permet de définir les classes d'encapsulation, les attributs et les méthodes de la technologie en plug-in qui seront exposés à Orchestrator.



<b>Actions</b>	Ensemble de fonctions JavaScript que vous pouvez utiliser directement dans les workflows et les tâches d'écriture de scripts. Les actions peuvent présenter plusieurs paramètres d'entrée et une seule valeur de retour.
<b>Workflows de composants de base</b>	Ensemble de workflows qui couvrent toutes les fonctionnalités génériques que vous souhaitez transmettre avec le plug-in. Généralement, un workflow de composants de base représente une opération dans l'interface utilisateur de la technologie orchestrée. Les workflows de composants de base peuvent être utilisés directement ou intégrés aux workflows spécialisés.
<b>Workflows spécialisés</b>	Ensemble de workflows qui couvrent une fonctionnalité spécifique du plug-in. Vous pouvez fournir des workflows spécialisés pour répondre à des exigences précises ou pour illustrer des exemples complexes d'utilisation des plug-ins.

## Exposer une API externe à Orchestrator

Vous exposez une API provenant d'un produit externe à la plateforme Orchestrator en créant un plug-in Orchestrator. Il vous est possible de créer un plug-in pour toute technologie qui expose une API pouvant être mappée dans les objets JavaScript utilisables par Orchestrator.

Les plug-ins mappent les objets et méthodes Java vers les objets JavaScript qu'ils ajoutent à l'API de script d'Orchestrator. Si une technologie externe expose une API Java, vous pouvez mapper l'API directement dans JavaScript pour qu'Orchestrator l'utilise dans les workflows et les actions.

Vous pouvez créer des plug-ins pour les applications qui exposent une API dans une langue autre que Java à l'aide de WSDL (Web Service Definition Language), REST (Representational State Transfer) ou un service de messagerie afin d'intégrer l'API exposée aux objets Java. Vous mappez ensuite les objets Java intégrés dans JavaScript pour qu'Orchestrator puisse les utiliser.

La technologie en plug-in est indépendante d'Orchestrator. Vous pouvez créer des plug-ins Orchestrator pour les produits externes même si vous ne pouvez accéder qu'au code binaire et non au code source : par exemple dans les archives Java (fichiers JAR).

## Composants d'un plug-in

Les plug-ins sont composés d'un jeu standard de composants qui exposent les objets au sein de la technologie en plug-in à la plate-forme Orchestrator

Les principaux composants d'un plug-in sont l'adaptateur de plug-in, la fabrique et les implémentations d'événement. Vous mappez les objets et opérations définis dans l'adaptateur, la fabrique et les implémentations d'événement avec les objets Orchestrator au sein d'un fichier de définition XML nommé `vso.xml`. Le fichier `vso.xml` mappe les objets et les fonctions de la technologie en plug-in avec les objets de script JavaScript qui apparaissent dans l'API JavaScript d'Orchestrator. Le fichier `vso.xml` mappe également les types d'objet issus de la technologie en plug-in avec les fonctions de recherche, qui apparaissent dans l'onglet **Inventaire**

Les plug-ins sont constitués des composants suivants.

<b>Le module plug-in</b>	C'est le plug-in en lui-même, tel que défini par un jeu de classes Java, un fichier <code>vso.xml</code> et des modules de workflows et d'actions qui interagissent avec les objets auxquels vous accédez au travers du plug-in. Le module plug-in est obligatoire.
<b>L'adaptateur de plug-in</b>	Il définit l'interface entre la technologie en plug-in et le serveur Orchestrator. L'adaptateur constitue le point d'entrée du plug-in dans la plate-forme Orchestrator. L'adaptateur crée la fabrique de plug-in, gère le chargement et le déchargement du plug-in et gère les événements qui arrivent aux objets de la technologie en plug-in. L'adaptateur de plug-in est obligatoire.
<b>La fabrique de plug-in</b>	Elle définit la façon dont Orchestrator trouve les objets de la technologie en plug-in et effectue des opérations sur ces derniers. L'adaptateur crée une fabrique pour la session client qui s'ouvre entre Orchestrator et une technologie en plug-in. La fabrique vous permet soit de partager une session entre toutes les connexions client, soit d'ouvrir une session par connexion client. La fabrique de plug-in est obligatoire.
<b>Configuration</b>	Orchestrator ne définit pas de moyen standard pour le plug-in de stocker sa configuration. Vous pouvez stocker les informations de configuration à l'aide de registres Windows, de fichiers de configuration statique, en conservant les informations dans une base de données ou dans des fichiers XML. Les plug-ins Orchestrator peuvent être configurés par l'exécution de workflows de configuration au sein du client Orchestrator.
<b>Fonctions de recherche</b>	Les règles d'interaction définissent la façon dont Orchestrator localise et représente les objets de la technologie en plug-in. Les fonctions de recherche retrouvent les objets dans la série d'objets que la technologie en plug-in expose à Orchestrator. Dans le fichier <code>vso.xml</code> , vous définissez les relations entre les objets pour pouvoir naviguer dans le réseau d'objets. Orchestrator représente l'objet modèle de la technologie en plug-in dans l'onglet <b>Inventaire</b> . Les fonctions de recherche sont obligatoires si vous désirez exposer les objets de la technologie en plug-in à Orchestrator.
<b>Objets de scripts</b>	Types d'objet JavaScript donnant accès aux objets, attributs et opérations de la technologie en plug-in. Les objets de script définissent la façon dont Orchestrator accède à l'objet modèle de la technologie en plug-in grâce au JavaScript. Vous mappez les classes et les méthodes de la technologie en plug-in aux objets JavaScript dans le fichier <code>vso.xml</code> . Vous pouvez accéder aux objets JavaScript dans l'API des scripts Orchestrator et les intégrer aux tâches, actions et workflows en script d'Orchestrator. Les objets de scripts sont obligatoires si vous désirez ajouter des types, classes et méthodes de script à l'API JavaScript d'Orchestrator.

**Inventaire**

Les instances d'objets de la technologie en plug-in qu'Orchestrator localise à l'aide des fonctions de recherche apparaissent dans la vue **Inventaire** du client Orchestrator. Vous pouvez réaliser des opérations sur les objets de l'inventaire en exécutant des workflows sur eux. L'inventaire est facultatif. Vous avez la possibilité de créer un plug-in qui ne fait qu'ajouter des types et classes de scripts à l'API JavaScript d'Orchestrator sans exposer la moindre instance d'objet de l'inventaire.

**Événements**

Changements dans l'état d'un objet de la technologie en plug-in. Orchestrator est en mesure de guetter de façon passive la survenue d'événements au sein de la technologie en plug-in. Il peut aussi déclencher de façon active des événements au sein de la technologie en plug-in. Les événements sont facultatifs.

**Rôle du fichier vso.xml**

Vous utilisez le fichier `vso.xml` pour mapper les objets, les classes, les méthodes et les attributs de la technologie en plug-in vers les objets d'inventaire, les types de scripts, les classes de scripts, les méthodes de scripts et les attributs. Le fichier `vso.xml` définit également la configuration et le comportement de départ du plug-in.

Le fichier `vso.xml` joue les rôles clés suivants.

**Comportement de départ et de configuration**

Définit la manière selon laquelle le plug-in démarre et localise les implémentations de configuration qu'il doit définir. Charge l'adaptateur de plug-ins.

**Objets d'inventaire**

Définit les types d'objets auxquels le plug-in accède dans la technologie en plug-in. Les méthodes de recherche de la fabrique de plug-ins localisent les instances de ces objets et les affichent dans l'inventaire Orchestrator.

**Types de scripts**

Ajoute des types de scripts à l'API JavaScript d'Orchestrator afin de permettre la représentation des différents types d'objets dans l'inventaire. Vous pouvez utiliser ces types de scripts en tant que paramètres d'entrée des workflows.

**Classes de scripts**

Ajoute des classes à l'API JavaScript d'Orchestrator que vous pouvez utiliser dans les éléments en scripts des workflows, actions, stratégies, etc.

**Méthodes de scripts**

Ajoute des méthodes à l'API JavaScript d'Orchestrator que vous pouvez utiliser dans les éléments en scripts des workflows, actions, stratégies, etc.

**Attributs de scripts**

Ajoute les attributs des objets de la technologie en plug-in à l'API JavaScript d'Orchestrator. Vous pouvez utiliser ces attributs dans les éléments en scripts des workflows, actions, stratégies, etc.

## Rôles de l'adaptateur de plug-in

L'adaptateur de plug-in constitue le point d'entrée du plug-in dans le serveur Orchestrator. L'adaptateur de plug-in sert de banque de données à la technologie en plug-in du serveur Orchestrator, crée la fabrique de plug-in et gère les événements qui surviennent au sein de la technologie en plug-in.

Pour créer un adaptateur de plug-in, commencez par créer une classe Java qui met en œuvre l'interface `IPluginAdaptor`.

La classe Adaptateur de plug-in que vous créez gère la fabrique, les événements et les déclencheurs du plug-in au sein de la technologie en plug-in. L'interface `IPluginAdaptor` fournit les méthodes dont vous vous servez pour réaliser ces tâches.

L'adaptateur de plug-in remplit les rôles principaux ci-dessous.

### Il crée une fabrique

Le rôle le plus important de l'adaptateur de plug-in est de charger et décharger une instance de la fabrique de plug-in pour chaque connexion d'Orchestrator à la technologie en plug-in. La classe Adaptateur de plug-in appelle la méthode `IPluginAdaptor.createPluginFactory()` pour créer une instance d'une classe qui met en œuvre l'interface `IPluginFactory`.

### Il gère les événements

L'adaptateur de plug-in constitue l'interface entre le serveur Orchestrator et la technologie en plug-in. Il gère les événements qu'Orchestrator suscite ou guette sur les objets se trouvant dans la technologie en plug-in.

L'adaptateur gère les événements au travers des modules de publication d'événement. Les modules de publication d'événement sont des instances de l'interface `IPluginEventPublisher` que crée l'adaptateur en appelant la méthode `IPluginAdaptor.registerEventPublisher()`. Les modules de publication d'événement définissent des déclencheurs et des indicateurs de stratégie sur les objets se trouvant dans la technologie en plug-in pour permettre à Orchestrator de lancer des actions définies en cas de survenue de certains événements dans l'objet ou si les valeurs de l'objet franchissent certains seuils. De la même façon, vous pouvez définir des instances `PluginTrigger` et `PluginWatcher` qui définissent les événements que les éléments de workflows au long cours En attente d'événement attendent.

### Il définit le nom du plug-in

Indiquez un nom pour le plug-in dans le fichier `vso.xml`. L'adaptateur de plug-in obtient ce nom auprès du fichier `vso.xml` et le publie sur la vue **Inventaire** du client Orchestrator.

### Il installe les licences

Vous pouvez appeler des méthodes pour installer tout fichier de licence requis par la technologie en plug-in pour la mise en œuvre de l'adaptateur.

Pour plus de précisions sur l'interface `IPluginAdaptor`, l'ensemble de ses méthodes et l'ensemble des autres classes de l'API du plug-in, consultez [Guide de référence de l'API de plug-in Orchestrator](#).

## Rôles de la fabrique de plug-in

La fabrique de plug-in définit la façon dont Orchestrator recherche les objets au sein de la technologie en plug-in et réalise des opérations sur les objets.

Pour créer la fabrique de plug-in, il vous faut mettre en œuvre et étendre l'interface `IPluginFactory` depuis l'API de plug-in Orchestrator. La classe Fabrique de plug-in que vous créez définit les fonctions de recherche qu'Orchestrator utilise pour accéder aux objets au sein de la technologie en plug-in. La fabrique permet au serveur Orchestrator de rechercher les objets par leur ID, leur relation à d'autres objets ou de les rechercher par l'intermédiaire d'une chaîne de requête.

La fabrique de plug-in accomplit les tâches principales ci-dessous.

**Elle recherche les objets**

Vous pouvez créer des fonctions qui recherchent les objets en fonction de leur nom et de leur type. Vous rechercherez les objets par nom et par type à l'aide de la méthode `IPluginFactory.find()`.

**Elle recherche les objets apparentés à d'autres objets**

Vous pouvez créer des fonctions qui recherchent les objets qui sont apparentés à un objet donné par le biais d'un type de relation donné. Vous définirez ces relations dans le fichier `vso.xml`. Vous pouvez également créer des fonctions de recherche pour les objets enfants dépendants qui sont apparentés à tous les parents par le biais d'un type de relation donné. Pour rechercher tout objet apparenté à un objet parent donné par le biais d'un type de relation donné, mettez en œuvre la méthode `IPluginFactory.findRelation()`. Pour découvrir si au moins un objet enfant existe pour une instance parent, mettez en œuvre la méthode `IPluginFactory.hasChildrenInRelation()`.

**Définir des requêtes pour rechercher des objets selon vos propres critères**

Vous pouvez créer des fonctions de recherche d'objets qui mettent en œuvre les règles de requête que vous définissez. Mettez en œuvre la méthode `IPluginFactory.findAll()` pour rechercher tous les objets qui satisfont aux règles de requête que vous définissez lorsque la fabrique appelle cette méthode. Vous obtiendrez les résultats d'une méthode `findAll()` dans un objet de `QueryResult` qui contient une liste de l'ensemble des objets répondant aux règles de requête que vous définissez.

Pour plus d'informations, sur l'interface `IPluginFactory`, l'ensemble de ses méthodes et l'ensemble des autres classes de l'API de plug-in, consultez [Guide de référence de l'API de plug-in Orchestrator](#).

## Rôle des objets de l'outil de recherche

Les objets de l'outil de recherche identifient et localisent des instances précises des types d'objets gérés de la technologie en plug-in. Orchestrator peut modifier les objets et interagir avec eux dès qu'il les trouve dans la technologie en plug-in. Pour ce faire, il exécute des workflows dans les objets de l'outil de recherche.

Chaque instance d'un type d'objet géré donné de la technologie en plug-in doit posséder un identifiant unique afin que les objets de l'outil de recherche Orchestrator puisse la retrouver. La technologie en plug-in qui fournit des identifiants uniques sous forme de chaînes aux instances des objets. Lorsqu'un workflow s'exécute, Orchestrator définit les identifiants uniques des objets qu'il trouve en tant que valeurs d'attribut du workflow. Les workflows nécessitant un objet d'un type précis en tant que paramètre d'entrée s'exécutent sur une instance spécifique de ce type d'objet.

Les objets de l'outil de recherche que les plug-ins ajoutent à l'API JavaScript d'Orchestrator se voient affecter le nom du plug-in en tant que préfixe. Par exemple, le type d'objet géré `VirtualMachine` de l'API vCenter Server apparaît dans Orchestrator sous le type JavaScript `VC:VirtualMachine`.

Par exemple, Orchestrator accède à une instance `VC:VirtualMachine` précise via le plug-in vCenter Server en implémentant un objet de l'outil de recherche qui utilise l'attribut `id` de la machine virtuelle en tant qu'identifiant unique. Vous pouvez transmettre cette instance d'objet aux éléments de workflow sous forme de valeurs d'attribut.

Un plug-in Orchestrator mappe les objets de la technologie en plug-in vers des objets équivalents de l'outil de recherche Orchestrator dans les éléments `<finder>` du fichier `vso.xml`. Les éléments `<finder>` identifient la méthode ou la fonction de la technologie en plug-in qui obtient l'identifiant unique d'une instance ou d'un objet spécifique. Les éléments `<finder>` déterminent également les relations entre les objets, afin de permettre de retrouver des objets en fonction de la manière dont ils sont liés aux autres objets.

Les objets de l'outil de recherche apparaissent dans l'onglet **Inventaire** d'Orchestrator, sous leur plug-in.

## Rôle des objets de scripts

Les objets de scripts sont des représentations JavaScript des objets provenant de la technologie en plug-in. Les objets de script des plug-ins apparaissent dans l'API JavaScript d'Orchestrator et vous pouvez les utiliser dans les éléments en script des workflows et des actions.

Les objets de script des plug-ins apparaissent dans l'API JavaScript d'Orchestrator sous forme de modules, de types et de catégories JavaScript. La plupart des objets de l'outil de recherche sont représentés par un objet de script. Les classes JavaScript peuvent ajouter des méthodes et des attributs à l'API JavaScript d'Orchestrator ; ces méthodes représentent les méthodes et attributs des objets de l'API de la technologie en plug-in. La technologie en plug-in fournit les implémentations des objets, des types, des classes et des méthodes indépendamment d'Orchestrator. Par exemple, le plug-in vCenter Server représente tous les objets de l'API vCenter Server sous forme d'objets JavaScript dans l'API JavaScript d'Orchestrator, avec les représentations de l'ensemble des classes, des méthodes et des attributs définis par l'API vCenter Server. Vous pouvez utiliser les classes de script vCenter Server, ainsi que les méthodes et attributs qu'ils définissent dans les fonctions en script d'Orchestrator.

Par exemple, le type d'objet géré `VirtualMachine` de l'API vCenter Server est retrouvé par l'outil de recherche `VC:VirtualMachine` et apparaît dans l'API JavaScript d'Orchestrator en tant que classe JavaScript `VcVirtualMachine`. La classe JavaScript `VcVirtualMachine` de l'API JavaScript d'Orchestrator définit des méthodes et attributs identiques aux objets gérés `VirtualMachine` de l'API vCenter Server.

Un plug-in Orchestrator mappe les objets, les types, les classes, les attributs et les méthodes de la technologie en plug-in vers des objets, des types, des classes, des attributs et des méthodes JavaScript Orchestrator équivalents dans l'élément `<scripting-objects>` du fichier `vso.xml`.

## Rôle des gestionnaires d'événements

Les événements sont des modifications apportées aux états ou aux attributs des objets qu'Orchestrator trouve dans la technologie en plug-in. Orchestrator surveille les événements en implémentant des gestionnaires d'événement.

Avec les plug-ins Orchestrator, vous pouvez surveiller les événements d'une technologie en plug-in de plusieurs manières. L'API des plug-ins Orchestrator vous permet de créer les types de gestionnaires d'événements suivants pour surveiller les événements d'une technologie en plug-in.

### **Écouteurs**

Surveillent de manière passive les modifications apportées à l'état des objets de la technologie en plug-in. La technologie en plug-in ou l'implémentation des plug-ins détermine les événements surveillés par les écouteurs. Les écouteurs ne lancent pas d'événements, mais ils avertissent Orchestrator lorsqu'un événement se produit. Les écouteurs détectent les événements soit en interrogeant la technologie en plug-in, soit en recevant des notifications de la technologie en plug-in. Lorsqu'un événement se produit, les stratégies ou workflows Orchestrator qui attendent l'événement peuvent réagir en lançant des opérations dans le serveur Orchestrator. Les composants des écouteurs sont facultatifs.

### **Stratégies**

Surveillent certains événements dans la technologie en plug-in et lancent des opérations dans le serveur Orchestrator si un événement se produit. Les stratégies peuvent surveiller les déclencheurs de stratégies et les indicateurs de stratégies. Les déclencheurs de stratégie définissent un événement de la technologie en plug-in qui, lorsqu'il se produit, provoque le lancement d'une opération dans le serveur Orchestrator par une stratégie en cours d'exécution. Il peut par exemple s'agir de lancer un workflow. Les indicateurs de stratégies définissent des gammes de valeurs pour les attributs d'un objet de la technologie en plug-in. Lorsque ces valeurs sont dépassées, Orchestrator lance une opération. Les stratégies sont facultatives.

### **Déclencheurs de workflow**

Si un workflow en cours d'exécution comporte un élément En attente d'événement et qu'il atteint cet élément, il interrompt son exécution et attend que l'événement se produise dans la technologie en plug-in. Les déclencheurs de workflows définissent les événements de la technologie en plug-in que les éléments En attente d'événement des workflows doivent attendre. Les déclencheurs de workflows s'enregistrent avec les observateurs. Les déclencheurs de workflows sont facultatifs.

### **Observateurs**

Observent les déclencheurs de workflows sur certains événements de la technologie en plug-in pour le compte de l'élément En attente d'événement d'un workflow. Lorsque l'événement se produit, les observateurs notifient tous les workflows qui attendaient cet événement. Les observateurs sont facultatifs.

## **Contenu et structure d'un plug-in**

Les plug-ins Orchestrator doivent contenir un ensemble normalisé de composants et respecter une structure de fichier normalisée. Pour que le plug-in soit conforme à la structure de fichier normalisée, il doit inclure des dossiers et des fichiers spécifiques.

Pour créer un plug-in Orchestrator, vous devez définir la façon dont Orchestrator accède aux objets et interagit avec eux dans la technologie en plug-in. De même, vous devez mapper tous les objets et toutes les fonctions de la technologie en plug-in vers les objets Orchestrator correspondants dans le fichier `vso.xml`.

Le fichier `vso.xml` doit inclure une référence à chaque type d'objet ou d'opération à exposer à Orchestrator. Chaque objet détecté par le plug-in dans la technologie en plug-in doit disposer d'un identifiant unique que vous indiquez. Vous définissez les noms d'objet dans les éléments `finder` et dans les éléments d'objet du fichier `vso.xml`.

Un plug-in peut prendre la forme d'un fichier d'archive Java standard (JAR) ou d'un fichier ZIP mais, dans tous les cas, le fichier doit être renommé avec une extension `.dar`.

---

**Note** Vous pouvez utiliser le Centre de contrôle Orchestrator pour importer un fichier DAR sur le serveur Orchestrator.

---

- **Définition du mappage des applications dans le fichier `vso.xml`**

Les objets que vous insérez dans le fichier `vso.xml` apparaissent en tant qu'objets de scripts dans l'API de script Orchestrator, ou en tant qu'objets de la fonction de recherche dans l'onglet **Inventaire** d'Orchestrator.

- **Format du fichier de définition du plug-in `vso.xml`**

Le fichier `vso.xml` définit la façon dont le serveur Orchestrator interagit avec la technologie en plug-in. Le fichier `vso.xml` doit inclure une référence à chaque type d'objet ou d'opération à exposer à Orchestrator.

- **Attribution de noms aux objets de plug-in**

Vous devez fournir un identifiant unique pour chaque objet que le plug-in trouve dans la technologie en plug-in. Les noms des objets sont à définir dans les éléments `<finder>` et dans les éléments `<object>` du fichier `vso.xml`.

- **Conventions de nommage des objets des plug-ins**

Vous devez respecter les conventions de nommage des classes Java lorsque vous nommez les objets des plug-ins.

- **Structure de fichiers du plug-in**

Un plug-in doit respecter une structure de fichiers standard et comprendre certains dossiers et fichiers précis. Le plug-in est livré sous la forme d'un fichier archive Java (JAR) ou ZIP standard que vous renommerez avec l'extension `.dar`.

## Définition du mappage des applications dans le fichier `vso.xml`

Les objets que vous insérez dans le fichier `vso.xml` apparaissent en tant qu'objets de scripts dans l'API de script Orchestrator, ou en tant qu'objets de la fonction de recherche dans l'onglet **Inventaire** d'Orchestrator.

Le fichier `vso.xml` fournit les informations suivantes au serveur Orchestrator :

- Version, nom et description du plug-in :



- Se réfère aux catégories de la technologie en plug-in et à l'adaptateur de plug-in associé
- Initialise le plug-in au démarrage du serveur Orchestrator
- Types de scripts afin de permettre la représentation des types d'objets dans la technologie en plug-in
- Les relations entre les types d'objets afin de définir le mode d'affichage des objets dans l'inventaire Orchestrator
- Les catégories de scripts qui mappent les objets et les opérations de la technologie en plug-in vers des types d'objets et de fonctions de l'API JavaScript d'Orchestrator.
- Des énumérations pour définir une liste de valeurs des constantes qui s'appliquent à tous les objets d'un certain type
- Les événements qu'Orchestrator surveille dans la technologie en plug-in

Le fichier `vso.xml` doit être conforme à la définition des schémas XML des plug-ins Orchestrator. Vous pouvez accéder à la définition des schémas sur le site de support de VMware.

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

Pour consulter les descriptions de tous les éléments du fichier `vso.xml`, reportez-vous à [Éléments du fichier de définition du plug-in vso.xml](#).

## Format du fichier de définition du plug-in vso.xml

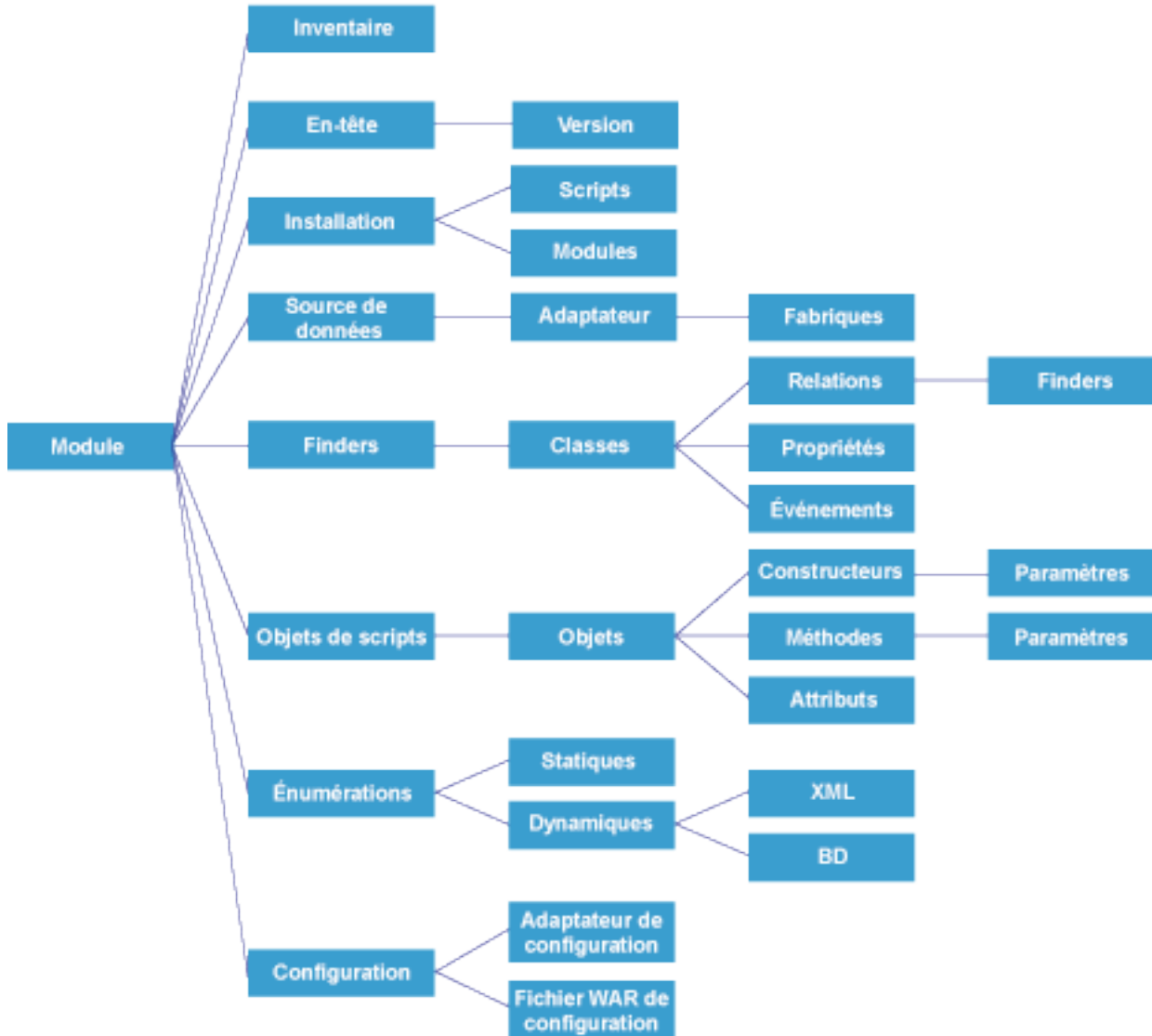
Le fichier `vso.xml` définit la façon dont le serveur Orchestrator interagit avec la technologie en plug-in. Le fichier `vso.xml` doit inclure une référence à chaque type d'objet ou d'opération à exposer à Orchestrator.

Les objets que vous insérez dans le fichier `vso.xml` apparaissent en tant qu'objets de scripts dans l'API de script Orchestrator, ou en tant qu'objets de la fonction de recherche dans l'onglet **Inventaire** d'Orchestrator.

S'inscrivant dans l'architecture ouverte et l'implémentation normalisée des plug-ins, le fichier `vso.xml` doit se conformer à un format standard.

Le diagramme suivant illustre le format du fichier de définition du plug-in `vso.xml` et la manière dont les éléments s'imbriquent les uns avec les autres.

Figure 1-4. Format du fichier de définition du plug-in vso.xml



### Attribution de noms aux objets de plug-in

Vous devez fournir un identifiant unique pour chaque objet que le plug-in trouve dans la technologie en plug-in. Les noms des objets sont à définir dans les éléments `<finder>` et dans les éléments `<object>` du fichier `vso.xml`.

Les opérations de la fonction de recherche que vous définissez dans l'implémentation de la fabrique recherchent les objets dans la technologie en plug-in. Lorsque le plug-in trouve des objets, vous avez la possibilité de les utiliser dans les workflows d'Orchestrator et de les passer d'un élément du workflow à un autre. Les identifiants uniques que vous fournissez pour les objets permettent de passer ces derniers d'un élément à un autre du workflow.

Le serveur Orchestrator ne stocke que le type et l'identifiant de chaque objet qu'il traite et ne stocke aucune information sur l'origine de l'objet ou la façon dont Orchestrator l'a obtenu. Vous devez nommer les objets de façon cohérente dans l'implémentation du plug-in de sorte que vous puissiez suivre à la trace les objets provenant des plug-ins.

Si le serveur Orchestrator s'arrête durant l'exécution des workflows, lorsque vous redémarrerez le serveur, les workflows reprendront à l'élément qui s'exécutait au moment de l'arrêt du serveur. Le workflow utilise les identifiants pour récupérer les objets que l'élément traitait lorsque le serveur s'est arrêté.

## Conventions de nommage des objets des plug-ins

Vous devez respecter les conventions de nommage des classes Java lorsque vous nommez les objets des plug-ins.

---

**Important** Compte tenu de la façon dont le moteur de workflows effectue la sérialisation des données, n'utilisez pas les séquences des chaînes suivantes dans les noms d'objets. L'utilisation de ces séquences de caractères dans les identifiants d'objets ne permet pas au moteur d'analyser correctement les workflows, ce qui peut entraîner un comportement imprévu lors de l'exécution de ces derniers.

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

---

Suivez les directives suivantes pour nommer les objets et le plug-ins.

- Chaque mot du nom doit commencer par une majuscule.
- N'utilisez pas d'espaces pour séparer les mots.
- Pour les lettres, utilisez uniquement des caractères standard de A à Z et de a à z.
- N'utilisez pas de caractères spéciaux tels que des accents.
- Le premier caractère d'un nom ne doit pas être un chiffre.
- Si possible, ne dépassez pas 10 caractères.

[Tableau 1-5. Règles de nommage des objets des plug-ins](#) indique les règles qui s'appliquent à chaque type d'objets.

**Tableau 1-5. Règles de nommage des objets des plug-ins**

Type d'objet	Règles de nommage
Plug-in	<ul style="list-style-type: none"> <li>■ Définies dans l'élément <code>&lt;module&gt;</code> du fichier <code>vso.xml</code>.</li> <li>■ Doit suivre les conventions de nommage des classes Java.</li> <li>■ Doivent être uniques. Vous ne pouvez pas exécuter deux plug-ins portant le même nom dans un serveur Orchestrator.</li> </ul>
Objet de l'outil de recherche	<ul style="list-style-type: none"> <li>■ Défini dans les éléments <code>&lt;finder&gt;</code> du fichier <code>vso.xml</code>.</li> <li>■ Doit suivre les conventions de nommage des classes Java.</li> <li>■ Doit être unique dans le plug-in.</li> </ul> <p>Orchestrator ajoute le nom du plug-in avec deux points aux noms des objets de l'outil de recherche dans les types d'objets de l'outil de recherche de l'API de script Orchestrator. Par exemple, le type d'objet <code>VirtualMachine</code> du plug-in <code>vCenter Server</code> apparaît dans l'API de script Orchestrator en tant que <code>VC:VirtualMachine</code>.</p>
Objet de script	<ul style="list-style-type: none"> <li>■ Défini dans les éléments <code>&lt;scripting-object&gt;</code> du fichier <code>vso.xml</code>.</li> <li>■ Doit suivre les conventions de nommage des classes Java.</li> <li>■ Doit être unique dans le serveur Orchestrator.</li> <li>■ Pour éviter toute confusion entre les objets de script, les objets de l'outil de recherche ou les objets de script d'autres plug-ins portant le même nom, n'oubliez jamais de placer un préfixe avec le nom du plug-in devant le nom de l'objet de script sans ajouter les deux points. Par exemple, la classe <code>VirtualMachine</code> du plug-in <code>vCenter Server</code> apparaît dans l'API de script Orchestrator en tant que classe <code>VcVirtualMachine</code>.</li> </ul>

## Structure de fichiers du plug-in

Un plug-in doit respecter une structure de fichiers standard et comprendre certains dossiers et fichiers précis. Le plug-in est livré sous la forme d'un fichier archive Java (JAR) ou ZIP standard que vous renommez avec l'extension `.dar`.

Le contenu du fichier archive DAR doit suivre la structure de dossiers et les conventions de nommage suivantes.

**Tableau 1-6. Structure du fichier archive DAR**

Dossiers	Description
<i>nom_du_plug-in</i> \VS0-INF\	Contient le fichier <code>vso.xml</code> qui définit le mappage des objets présents dans la technologie en plug-in avec les objets Orchestrator.  Le dossier VS0-INF et le fichier <code>vso.xml</code> sont obligatoires.
<i>nom_du_plug-in</i> \lib\	Contient les fichiers JAR qui renferment les binaires de la technologie en plug-in. Contient aussi les fichiers JAR qui renferment les implémentations de l'adaptateur, de la fabrique, des gestionnaires de notifications et d'autres interfaces au sein du plug-in.  Le dossier lib et les fichiers JAR sont obligatoires.
<i>nom_du_plug-in</i> \resources\	Contient les fichiers de ressources que nécessite le plug-in. Le dossier resources peut comprendre les types d'élément suivants : <ul style="list-style-type: none"> <li>■ Des fichiers image, pour représenter les objets du plug-in dans l'onglet <b>Inventaire</b> d'Orchestrator.</li> <li>■ Des scripts, pour définir le comportement à l'initialisation lorsque le plug-in se lance.</li> <li>■ Des modules Orchestrator, qui peuvent contenir des workflows personnalisés et d'autres ressources pour interagir avec les objets auxquels vous accédez à l'aide du plug-in.</li> </ul> <p>Vous pouvez organiser les ressources dans des sous-dossiers. Par exemple, <code>resources\images\</code>, <code>resources\scripts\</code> ou <code>resources\packages\</code>.</p> <p>Le dossier resources est facultatif.</p>

Vous pouvez utiliser le Centre de contrôle Orchestrator pour importer un fichier DAR sur le serveur Orchestrator.

## Guide de référence de l'API de plug-in Orchestrator

L'API de plug-in Orchestrator définit les interfaces et classes à mettre en œuvre et à étendre lorsque vous développez les implémentations de `IPluginAdaptor` et de `IPluginFactory` pour créer un plug-in.

Sauf indication contraire, la totalité des classes est contenue dans le module `ch.dunes.vso.sdk.api`.

### Interface IAop

L'interface `IAop` propose des méthodes pour obtenir et définir des propriétés sur les objets de la technologie en plug-in.

```
public interface IAop
```

L'interface `IAop` définit les méthodes suivantes :

Méthode	Renvois	Description
<code>get(java.lang.String propertyName, java.lang.Object object, java.lang.Object sdkObject)</code>	<code>java.lang.Object</code>	Obtient une propriété à partir d'un objet donné dans le plug-in.
<code>set(java.lang.String propertyName, java.lang.String propertyValue, java.lang.Object object)</code>	<code>Void</code>	Définit une propriété sur un objet donné dans le plug-in.

## Interface IDynamicFinder

L'interface `IDynamicFinder` renvoie l'ID et les propriétés d'une fonction de recherche par la programmation au lieu de définir l'ID et les propriétés dans le fichier `vso.xml`.

L'interface `IDynamicFinder` définit les méthodes suivantes.

Méthode	Renvois	Description
<code>getIdAccessor(java.lang.String type)</code>	<code>java.lang.String</code>	Fournit une expression OGNL pour obtenir l'ID d'un objet par programmation.
<code>getProperties(java.lang.String type)</code>	<code>java.util.List&lt;SDKFinderProperty&gt;</code>	Fournit une liste des propriétés des objets par programmation.

## Interface IPluginAdaptor

L'interface `IPluginAdaptor` est mise en œuvre pour gérer les fabriques, événements et observateurs de plug-in. L'interface `IPluginAdaptor` définit un adaptateur entre un plug-in et le serveur Orchestrator.

Les instances d'`IPluginAdaptor` sont en charge de la gestion des sessions. L'interface `IPluginAdaptor` définit les méthodes suivantes.

Méthode	Renvois	Description
<code>addWatcher(PluginWatcher watcher)</code>	<code>Void</code>	Ajoute un observateur pour surveiller la survenue d'un événement précis.
<code>createPluginFactory(java.lang.String sessionId, java.lang.String username, java.lang.String password, IPluginNotificationHandler notificationHandler)</code>	<code>IPluginFactory</code>	<p>Crée une instance d'<code>IPluginFactory</code>. Le serveur Orchestrator se sert de la fabrique pour obtenir les objets auprès de la technologie en plug-in par leur ID, leur relation avec d'autres objets et ainsi de suite.</p> <p>L'ID de session vous permet d'identifier une session en cours d'exécution. Par exemple, un utilisateur peut très bien se connecter à deux clients Orchestrator différents et exécuter deux sessions simultanément.</p> <p>De la même façon, le démarrage d'un workflow crée une session indépendante du client dans lequel le workflow a démarré. Un workflow continue de s'exécuter même si vous refermez le client Orchestrator.</p>

Méthode	Renvois	Description
<code>installLicenses(PluginLicense[] licenses)</code>	Void	Installe les informations de licence des plug-ins standard fournis par VMware
<code>registerEventPublisher(java.lang.String type, java.lang.String id, IPluginEventPublisher publisher)</code>	Void	Définit des déclencheurs et des indicateurs de stratégie sur un élément de l'inventaire
<code>removeWatcher(java.lang.String watcherId)</code>	Void	Supprime un observateur
<code>setPluginName(java.lang.String pluginName)</code>	Void	Récupère le nom du plug-in dans le fichier <code>vso.xml</code>
<code>setPluginPublisher(IPluginPublisher pluginPublisher)</code>	Void	Définit l'éditeur du plug-in
<code>uninstallPluginFactory(IPluginFactory plugin)</code>	Void	Désinstalle une fabrique de plug-ins.
<code>unregisterEventPublisher(java.lang.String type, java.lang.String id, IPluginEventPublisher publisher)</code>	Void	Supprime les déclencheurs et les indicateurs de stratégie d'un élément de l'inventaire

## Interface IPluginEventPublisher

L'interface `IPluginEventPublisher` publie des déclencheurs et des indicateurs de stratégie sur un bus de notification d'événements pour une surveillance par les stratégies d'Orchestrator.

Vous avez la possibilité de créer des instances d'`IPluginEventPublisher` directement dans l'implémentation de l'adaptateur du plug-in ou les créer dans des classes de générateur d'événement distinctes.

Vous pouvez implémenter l'interface `IPluginEventPublisher` pour publier des événements dans la technologie en plug-in pour le moteur de stratégies Orchestrator. Vous créez des méthodes pour définir des déclencheurs et des indicateurs de stratégies sur des objets dans la technologie en plug-in et des écouteurs d'événements pour écouter les événements de ces objets.

Les stratégies permettent d'implémenter des indicateurs ou des déclencheurs pour surveiller les objets de la technologie en plug-in. Les indicateurs de stratégie surveillent les attributs des objets et insèrent un événement dans le serveur Orchestrator si les valeurs des objets dépassent certaines limites. Les déclencheurs de stratégie surveillent les objets et insèrent un événement dans le serveur Orchestrator si un événement prédéfini se produit sur l'objet. Vous enregistrez les indicateurs et déclencheurs de stratégies avec les instances de `IPluginEventPublisher` afin que les stratégies Orchestrator puissent les surveiller.

L'interface `IPluginEventPublisher` définit les méthodes suivantes.

Type	Renvoie	Description
<code>pushGauge(java.lang.String type, java.lang.String id, java.lang.String gaugeName, java.lang.String deviceName, java.lang.Double gaugeValue)</code>	Void	Publier un indicateur de stratégie pour une surveillance par les stratégies. Adopte les paramètres suivants : <ul style="list-style-type: none"> <li>■ <code>type</code> : type de l'objet à surveiller.</li> <li>■ <code>id</code> : identifiant de l'objet à surveiller.</li> <li>■ <code>gaugeName</code> : nom de cet indicateur de stratégie.</li> <li>■ <code>deviceName</code> : nom du type d'attribut que l'indicateur de stratégie surveille.</li> <li>■ <code>gaugeValue</code> : valeur en attente de laquelle l'indicateur de stratégie surveille l'objet.</li> </ul>
<code>pushTrigger(java.lang.String type, java.lang.String id, java.lang.String triggerName, java.util.Properties additionalProperties)</code>	Void	Publier un déclencheur pour une surveillance par les stratégies. Adopte les paramètres suivants : <ul style="list-style-type: none"> <li>■ <code>type</code> : type de l'objet à surveiller.</li> <li>■ <code>id</code> : identifiant de l'objet à surveiller.</li> <li>■ <code>triggerName</code> : nom de ce déclencheur.</li> <li>■ <code>additionalProperties</code> : toute propriété complémentaire pour la surveillance par le déclencheur.</li> </ul>

## Interface IPluginFactory

L'API `IPluginAdaptor` renvoie des instances de `IPluginFactory`. Les instances de `IPluginFactory` exécutent des commandes dans l'application en plug-in et trouve les objets sur lesquels réaliser les opérations d'Orchestrator.

L'interface `IPluginFactory` définit le champ suivant :

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

L'interface `IPluginFactory` définit les méthodes suivantes.

Méthode	Renvoie	Description
<code>executePluginCommand(java.lang.String cmd)</code>	Void	Utiliser le plug-in pour exécuter une commande. VMware recommande de ne pas utiliser cette méthode.
<code>find(java.lang.String type, java.lang.String id)</code>	<code>java.lang.Object</code>	Utiliser le plug-in pour rechercher un objet. Identifier l'objet par son ID et son type.
<code>findAll(java.lang.String type, java.lang.String query)</code>	<code>QueryResult</code>	Utiliser le plug-in pour rechercher des objets d'un certain type et répondant à une chaîne de requête. La syntaxe de la requête se définit dans l'implémentation d' <code>IPluginFactory</code> du plug-in. Si vous ne définissez pas la syntaxe de la requête, <code>findAll()</code> renverra tous les objets du type indiqué.



Méthode	Renvoie	Description
<code>findRelation(java.lang.String parentType, java.lang.String parentId, java.lang.String relationName)</code>	<code>java.util.List</code>	Détermine si un objet possède des enfants.
<code>hasChildrenInRelation(java.lang.String parentType, java.lang.String parentId, java.lang.String relationName)</code>	<code>HasChildrenResult</code>	Recherche tous les enfants d'un parent donné au travers d'une certaine relation.
<code>invalidate(java.lang.String type, java.lang.String id)</code>	<code>Void</code>	Invalider les objets par type et ID.
<code>void invalidateAll()</code>	<code>Void</code>	Invalider tous les objets se trouvant dans le cache.

## Interface IPluginNotificationHandler

L'API `IPluginNotificationHandler` définit des méthodes pour signaler à Orchestrator la survenue de différents types d'événements sur les objets auxquels il accède grâce au plug-in.

L'interface `IPluginNotificationHandler` définit les méthodes suivantes.

Méthode	Renvoie	Description
<code>getSessionID()</code>	<code>java.lang.String</code>	Renvoie l'ID de la session en cours
<code>notifyElementDeleted(java.lang.String type, java.lang.String id)</code>	<code>Void</code>	Signale au système qu'un objet du type et de l'ID donnés a été supprimé
<code>notifyElementInvalidate(java.lang.String type, java.lang.String id)</code>	<code>Void</code>	Signale au système que les relations d'un objet ont changé. Vous pouvez utiliser la méthode <code>notifyElementInvalidate()</code> pour signaler à Orchestrator tous les changements intervenus dans les relations entre objets et pas seulement ceux qui invalident un objet. Par exemple, l'ajout d'un objet enfant à un parent représente un changement dans la relation entre les deux objets.
<code>notifyElementUpdated(java.lang.String type, java.lang.String id)</code>	<code>Void</code>	Signale au système que les attributs d'un objet ont été modifiés
<code>notifyMessage(ch.dunes.vso.sdk.api.ErrorLevel severity, java.lang.String type, java.lang.String id, java.lang.String message)</code>	<code>Void</code>	Publie un message d'erreur relatif au module actuel

## Interface IPluginPublisher

L'interface `IPluginPublisher` publie un événement observateur sur un bus de notification d'événements pour la surveillance des éléments de workflow au long cours En attente d'événement.

Lorsqu'un déclencheur de workflows lance un événement dans la technologie en plug-in, l'observateur de plug-ins surveillant ce déclencheur et enregistré auprès d'une instance de `IPluginPublisher` notifie tous les workflows en attente que l'événement a eu lieu.

L'interface `IPluginPublisher` définit la méthode suivante.

Type	Valeur	Description
<code>pushWatcherEvent(java.lang.String id, java.util.Properties properties)</code>	<code>Void</code>	Publier un événement observateur sur un bus de notification d'événements

## Interface `WebConfigurationAdaptor`

L'interface `WebConfigurationAdaptor` implémente `IConfigurationAdaptor` et définit des méthodes pour localiser et installer une application Web dans l'onglet de configuration d'un plug-in.

**Note** L'interface `WebConfigurationAdaptor` est obsolète depuis Orchestrator 4.1. Pour ajouter une application Web à la configuration, implémentez `IConfigurationAdaptor` et utilisez l'attribut `configuration-war` du fichier `vso.xml` pour identifier l'application Web.

L'interface `WebConfigurationAdaptor` définit les méthodes suivantes.

Méthode	Renvois	Description
<code>getWebAppContext()</code>	<code>String</code>	Localise le fichier WAR de l'application Web pour l'onglet de configuration. Fournit le nom et le chemin d'accès au fichier WAR sous forme de chaîne à partir du répertoire <code>/webapps</code> du fichier DAR.
<code>setWebConfiguration(boolean webConfiguration)</code>	<code>Booléen</code>	Détermine si le contenu de l'onglet de configuration est défini par une application Web.

## Classe `PluginTrigger`

La classe `PluginTrigger` crée un module déclencheur qui obtient les informations relatives aux objets et événements à surveiller au sein de la technologie en plug-in pour le compte de l'élément de workflow En attente d'événement.

La classe `PluginTrigger` définit des méthodes pour obtenir ou définir le type et le nom de l'objet à surveiller, la nature de l'événement et le délai d'expiration.

Vous créez des implémentations de la classe `PluginTrigger` qui seront exclusivement utilisées par les éléments Attendre événement des workflows. Vous définissez les déclencheurs des stratégies Orchestrator dans les classes qui définissent les événements et implémentent la méthode `IPluginEventPublisher.pushTrigger()`.

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

La classe `PluginTrigger` définit les méthodes suivantes :

Méthode	Renvoie	Description
getModuleName()	java.lang.String	Obtient le nom du module déclencheur.
getProperties()	java.util.Properties	Obtient la liste des propriétés pour le déclencheur.
getSdkId()	java.lang.String	Obtient l'ID de l'objet à surveiller dans la technologie en plug-in.
getSdkType()	java.lang.String	Obtient le type de l'objet à surveiller dans la technologie en plug-in.
getTimeout()	Long	Obtient le délai d'expiration du déclencheur.
setModuleName(java.lang.String moduleName)	Void	Définit le nom du module déclencheur.
setProperties(java.util.Properties properties)	Void	Définit la liste des propriétés pour le déclencheur.
setSdkId(java.lang.String sdkId)	Void	Définit l'ID de l'objet à surveiller dans la technologie en plug-in.
setSdkType(java.lang.String sdkType)	Void	Définit le type de l'objet à surveiller dans la technologie en plug-in.
setTimeout(long timeout)	Void	Définit un délai d'expiration en secondes. Une valeur négative désactivera le délai d'expiration.

## Constructeurs

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

## Classe PluginWatcher

La classe PluginWatcher guette un événement défini de la technologie en plug-in dans le module déclencheur pour le compte de l'élément de workflow au long cours En attente d'événement.

La classe PluginWatcher définit un outil de création que vous pouvez utiliser pour créer des instances d'observation des plug-ins. La classe PluginWatcher définit des méthodes pour obtenir ou définir le nom du déclencheur de workflows à observer et un délai d'expiration.

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

La classe PluginWatcher définit les méthodes suivantes :

Méthode	Renvoie	Description
getId()	java.lang.String	Obtient l'ID du déclencheur
getModuleName()	java.lang.String	Obtient le nom du module déclencheur

Méthode	Renvoie	Description
getTimeoutDate()	Long	Obtient la date d'expiration du déclencheur
getTrigger()	Void	Obtient un déclencheur
setId(java.lang.String id)	Void	Définit l'ID du déclencheur
setTimeoutDate()	Void	Définit la date d'expiration du déclencheur

## Constructeur

PluginWatcher(PluginTrigger trigger)

## Classe QueryResult

La classe QueryResult contient les résultats d'une requête find effectuée sur les objets auxquels Orchestrator accède grâce aux plug-ins.

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

La valeur de totalCount peut être supérieure au nombre d'éléments que le QueryResult renvoie si le nombre total de résultats trouvés dépasse le nombre de résultats que la requête renvoie. Le nombre de résultats que la requête renvoie est défini dans la syntaxe de requête présente dans le fichier vso.xml.

La classe QueryResult définit les méthodes suivantes :

Méthode	Renvoie	Description
addElement(java.lang.Object element)	Void	Ajoute un élément au QueryResult
addElements(java.util.List elements)	Void	Ajoute une liste d'éléments au QueryResult
getElements()	java.util.List	Obtient les éléments auprès de l'application en plug-in
getTotalCount()	Long	Obtient un comptage de tous les éléments disponibles au sein de la technologie en plug-in
isPartialResult()	Booléen	Détermine si le résultat obtenu est final
removeElement(java.lang.Object element)	Void	Supprime un élément de la technologie en plug-in
setElements(java.util.List elements)	Void	Définit les éléments dans la technologie en plug-in
setTotalCount(long totalCount)	Void	Définit le nombre total d'éléments disponibles dans la technologie en plug-in

## Constructeurs

- QueryResult()

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

## Classe SDKFinderProperty

La classe `SDKFinderProperty` définit des méthodes pour obtenir et définir les propriétés dans un objet trouvé dans la technologie en plug-in par les objets fonction de recherche d'Orchestrator. La méthode `IDynamicFinder.getProperties` renvoie des objets `SDKFinderProperty`.

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

La classe `SDKFinderProperty` définit les méthodes suivantes :

Méthode	Renvoie	Description
<code>getAttributeName()</code>	<code>java.lang.String</code>	Obtient un nom d'attribut pour l'objet
<code>getBeanProperty()</code>	<code>java.lang.String</code>	Obtient les propriétés auprès d'un Java bean
<code>getDescription()</code>	<code>java.lang.String</code>	Obtient une description pour l'objet
<code>getDisplayName()</code>	<code>java.lang.String</code>	Obtient un nom complet pour l'objet
<code>getPossibleResultType()</code>	<code>java.lang.String</code>	Obtient les types possibles de résultat que la fonction de recherche renvoie
<code>getPropertyAccessor()</code>	<code>java.lang.String</code>	Obtient un accesseur de propriété pour l'objet
<code>getPropertyAccessorTree()</code>	<code>java.lang.Object</code>	Obtient une arborescence d'accesseurs de propriétés pour l'objet
<code>isHidden()</code>	Booléen	Affiche ou masque l'objet
<code>isShowInColumn()</code>	Booléen	Affiche ou masque l'objet dans la colonne Base de données
<code>isShowInDescription()</code>	Booléen	Affiche ou masque la description de l'objet
<code>setAttributeName(java.lang.String attributeName)</code>	Void	Définit un nom d'attribut pour l'objet
<code>setBeanProperty(java.lang.String beanProperty)</code>	Void	Définit les propriétés auprès d'un Java bean
<code>setDescription(java.lang.String description)</code>	Void	Définit une description pour l'objet
<code>setDisplayName(java.lang.String displayName)</code>	Void	Définit un nom complet pour l'objet
<code>setHidden(boolean hidden)</code>	Void	Affichent ou masquent l'objet
<code>setPossibleResultType(java.lang.String possibleResultType)</code>	Void	Définit les types possibles de résultat que la fonction de recherche renvoie
<code>setPropertyAccessor(java.lang.String propertyAccessor)</code>	Void	Définit un accesseur de propriété pour l'objet

Méthode	Renvoie	Description
setPropertyAccessorTree(java.lang.Object propertyAccessorTree)	Void	Définit une arborescence d'accesseurs de propriétés pour l'objet
setShowInColumn(boolean showInTable)	Void	Affichent ou masquent l'objet dans la colonne Base de données
setShowInDescription(boolean showInDescription)	Void	Affichent ou masquent la description de l'objet

## Constructeur

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

## Classe PluginExecutionException

La classe PluginExecutionException renvoie un message d'erreur si le plug-in tombe sur une exception lorsqu'il exécute une opération.

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

La classe PluginExecutionException hérite des méthodes suivantes de class java.lang.Throwable :

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

## Constructeur

PluginExecutionException(java.lang.String message)

## Classe PluginOperationException

La classe PluginOperationException traite les erreurs rencontrées au cours du fonctionnement du plug-in.

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

La classe PluginOperationException hérite des méthodes suivantes de class java.lang.Throwable :

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

## Constructeur

PluginOperationException(java.lang.String message)

## Énumération HasChildrenResult

L'énumération `HasChildrenResult` déclare si un parent donné possède des enfants. La méthode `IPluginFactory.hasChildrenInRelation` renvoie des objets `HasChildrenResult`.

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

L'énumération `HasChildrenResult` définit les constantes suivantes :

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

L'énumération `HasChildrenResult` définit les méthodes suivantes :

Méthode	Renvois	Description
<code>getValue()</code>	<code>int</code>	Renvoie l'une des valeurs suivantes : <div> <div><b>1</b></div> <div>Le parent possède des enfants</div> </div> <div> <div><b>-1</b></div> <div>Le parent ne possède pas d'enfant</div> </div> <div> <div><b>0</b></div> <div>Paramètre inconnu ou invalide</div> </div>
<code>valueOf(java.lang.String name)</code>	<code>static HasChildrenResult</code>	Renvoie une constante d'énumération de ce type avec le nom indiqué. La Chaîne doit correspondre exactement à l'identifiant utilisé pour déclarer une énumération de ce type. Ne pas utiliser d'espaces dans le nom de l'énumération.
<code>values()</code>	<code>static HasChildrenResult[]</code>	Renvoie un tableau comprenant les constantes de ce type d'énumération, dans l'ordre déclaré. Cette méthode peut couvrir les constantes de la manière suivante : <div> <pre>for (HasChildrenResult c : HasChildrenResult.values()) System.out.println(c);</pre> </div>

L'énumération `HasChildrenResult` hérite des méthodes suivantes de class `java.lang.Enum` :

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

## Type d'annotation ScriptingAttribute

Le type d'annotation `ScriptingAttribute` annote un attribut issu d'un objet de la technologie en plug-in pour l'utiliser comme propriété dans l'écriture de scripts.

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

Le type d'annotation `ScriptingAttribute` possède la valeur suivante :

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

## Type d'annotation ScriptingFunction

Le type d'annotation `ScriptingFunction` annote une méthode pour l'utiliser comme propriété dans l'écriture de scripts.

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

Le type d'annotation `ScriptingFunction` possède la valeur suivante :

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

## Type d'annotation ScriptingParameter

Le type d'annotation `ScriptingParameter` annote un paramètre pour l'utiliser comme propriété dans l'écriture de scripts.

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

Le type d'annotation `ScriptingParameter` possède la valeur suivante :

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

## Éléments du fichier de définition du plug-in vso.xml

Le fichier `vso.xml` comporte un ensemble de workflows standard. Certains éléments sont obligatoires tandis que d'autres sont facultatifs. Chaque élément présente des attributs qui définissent les valeurs des objets et des opérations que vous mappez vers des objets et opérations Orchestrator.

En outre, les éléments peuvent présenter des éléments enfants ou en être dépourvus. Un élément enfant permet de mieux définir l'élément parent. Le même élément enfant peut apparaître dans plusieurs éléments parents. Par exemple, l'élément `description` est dépourvu d'éléments enfants mais apparaît lui-même en tant qu'élément enfant dans de nombreux éléments parents : `module`, `example`, `trigger`, `gauge`, `finder`, `constructor`, `method`, `object` et `enumeration`.



Chaque définition d'élément qui suit répertorie ses attributs, ses parents et ses enfants.

## Élément module

Un module désigne un jeu d'objets de plug-in à mettre à disposition d'Orchestrator.

Le module contient des informations sur la façon dont les données issues de la technologie en plug-in sont mappées avec les classes Java, sur le contrôle de version, sur la façon de déployer le module et sur la façon dont le plug-in apparaîtra dans l'inventaire d'Orchestrator.

L'élément `<module>` est facultatif. L'élément `<module>` possède les attributs suivants :

Attributs	Valeur	Description
name	Chaîne	Définit le type de tous les éléments <code>&lt;finder&gt;</code> au sein du plug-in. Attribut obligatoire.
version	Numéro	Numéro de version du plug-in, à utiliser au moment de recharger les modules dans une nouvelle version du plug-in. Attribut obligatoire.
build-number	Numéro	Numéro de build du plug-in, à utiliser au moment de recharger les modules dans une nouvelle version du plug-in. Attribut obligatoire.
image	Fichier image	Icône à afficher dans l'inventaire d'Orchestrator. Attribut obligatoire.
display-name	Chaîne	Nom apparaissant dans l'inventaire d'Orchestrator. Attribut facultatif.
interface-mapping-allowed	true ou false	VMware décourage fortement de mapper les interfaces. Attribut facultatif.

**Tableau 1-7. Hiérarchie des éléments**

Élément parent	Éléments enfants
Aucun	<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>

## Élément obsolète

Les éléments `<description>` fournissent des descriptions des éléments de plug-in apparaissant dans la documentation de l'explorateur de l'API.

Vous ajoutez le texte qui apparaît dans la documentation de l'explorateur de l'API entre les balises `<description>` et `</description>`.

L'élément `<description>` est facultatif. L'élément `<description>` ne dispose d'aucun attribut.

**Tableau 1-8. Hiérarchie des éléments**

Éléments parents	Éléments enfants
■ <code>&lt;module&gt;</code>	Aucun
■ <code>&lt;example&gt;</code>	
■ <code>&lt;trigger&gt;</code>	
■ <code>&lt;gauge&gt;</code>	
■ <code>&lt;finder&gt;</code>	
■ <code>&lt;constructor&gt;</code>	
■ <code>&lt;method&gt;</code>	
■ <code>&lt;object&gt;</code>	
■ <code>&lt;enumeration&gt;</code>	

## Élément obsolète

L'élément `<deprecated>` indique les objets et les méthodes obsolètes dans la documentation de l'explorateur de l'API.

Vous ajoutez le texte qui apparaît dans la documentation de l'explorateur de l'API entre les balises `<deprecated>` et `</deprecated>`.

L'élément `<deprecated>` est facultatif. L'élément `<deprecated>` ne dispose d'aucun attribut.

**Tableau 1-9. Hiérarchie des éléments**

Éléments parents	Éléments enfants
■ <code>&lt;method&gt;</code>	Aucun
■ <code>&lt;object&gt;</code>	

## Élément url

L'élément `<url>` fournit une URL qui pointe sur la documentation externe d'un objet ou d'une énumération.

Vous indiquerez l'URL entre les balises `<url>` et `</url>`.

L'élément `<url>` est facultatif. L'élément `<url>` ne possède aucun attribut.

**Tableau 1-10. Hiérarchie des éléments**

Éléments parents	Éléments enfants
■ <code>&lt;enumeration&gt;</code>	Aucun
■ <code>&lt;object&gt;</code>	

## Élément installation

L'élément `<installation>` vous permet d'installer un module ou d'exécuter un script au démarrage du serveur.

L'élément `<installation>` est facultatif. L'élément `<installation>` possède les attributs suivants :

Attributs	Valeur	Description
mode	always, never ou version	La définition de la valeur mode occasionne le comportement suivant lorsque le serveur Orchestrator se lance : <ul style="list-style-type: none"> <li>■ L'action s'exécute always</li> <li>■ L'action ne s'exécute never</li> <li>■ L'action s'exécute lorsque le serveur détecte une toute nouvelle version du plug-in</li> </ul> Attribut obligatoire.

Tableau 1-11. Hiérarchie des éléments

Élément parent	Élément enfant
<module>	<action>

## Élément d'action

L'élément <action> indique l'action qui s'exécute au démarrage du serveur Orchestrator.

Les attributs de l'élément <action> fournissent un chemin d'accès au module ou au script Orchestrator qui définit le comportement du plug-in à son démarrage.

L'élément <action> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <action>.

L'élément <action> présente les attributs suivants.

Attributs	Valeur	Description
resource	Chaîne	Chemin d'accès au module ou au script Java à partir de la racine du fichier dar. Attribut obligatoire.
type	install-package ou execute-script	Installe le module Orchestrator indiqué sur le serveur Orchestrator ou exécute le script indiqué. Attribut obligatoire.

Tableau 1-12. Hiérarchie des éléments

Élément parent	Éléments enfants
<installation>	Aucun

## Élément finder-datasource

L'élément <finder-datasources> constitue le conteneur des éléments <finder-datasource>.

L'élément <finder-datasources> est facultatif. L'élément <finder-datasources> ne dispose d'aucun attribut.

Tableau 1-13. Hiérarchie des éléments

Élément parent	Éléments enfants
<module>	<finder-datasource>

## Élément finder-datasource

L'élément `<finder-datasource>` redirige vers le fichier de classe Java de l'implémentation `IPluginAdaptor` que vous créez pour le plug-in.

Vous définissez comment Orchestrator accède aux objets de la technologie en plug-in dans l'élément `<finder-datasource>`. L'élément `<finder-datasource>` identifie la classe Java de l'adaptateur de plug-ins que vous créez. La classe de l'adaptateur de plug-ins instancie la fabrique de plug-ins que vous créez. La fabrique de plug-ins définit les méthodes qui recherchent les objets dans la technologie en plug-in. Vous pouvez définir des délais d'expiration dans l'élément `<finder-datasource>` pour les appels à la méthode de recherche réalisés par la fabrique. Les délais d'expiration varient selon les méthodes de recherche dans l'interface `IPluginFactory`.

L'élément `<finder-datasource>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<finder-datasources>`. L'élément `<finder-datasource>` présente les attributs suivants.

Attributs	Valeur	Description
<code>name</code>	Chaîne	Identifie la source de données dans les attributs <code>datasource</code> de l'élément <code>&lt;finder&gt;</code> . Identique à un id XML. Attribut obligatoire.
<code>adaptor-class</code>	Classe Java	Redirige vers l'implémentation <code>IPluginAdaptor</code> que vous définissez pour créer l'adaptateur de plug-ins, par exemple <code>com.vmware.plugins.sample.Adaptor</code> . Attribut obligatoire.
<code>concurrent-call</code>	<code>true</code> (par défaut) ou <code>false</code>	Permet à plusieurs utilisateurs d'accéder simultanément à l'adaptateur. Vous devez définir <code>concurrent-call</code> sur <code>false</code> si le plug-in ne prend pas en charge les appels simultanés. Attribut facultatif.
<code>invoker-mode</code>	<code>direct</code> (par défaut) ou <code>timeout</code>	Définit un délai d'expiration pour la fonction de recherche. Si elle est définie sur <code>direct</code> , les appels vers la fonction de recherche n'expirent jamais. Si elle est définie sur <code>timeout</code> , le serveur Orchestrator applique le délai d'expiration qui correspond à la méthode de la fonction de recherche. Attribut facultatif.
<code>anonymous-login-mode</code>	<code>never</code> (par défaut) ou <code>always</code>	Transmet ou ne transmet pas le nom d'utilisateur et le mot de passe de l'utilisateur au plug-in. Attribut facultatif.
<code>timeout-fetch-relation</code>	Numéro ; 30 secondes par défaut	S'applique aux appels de <code>findRelation()</code> . Attribut facultatif.
<code>timeout-find-all</code>	Numéro ; 60 secondes par défaut	S'applique aux appels de <code>findAll()</code> . Attribut facultatif.
<code>timeout-find</code>	Numéro ; 60 secondes par défaut	S'applique aux appels de <code>find()</code> . Attribut facultatif.

Attributs	Valeur	Description
timeout-has-children-in-relation	Numéro ; 2 secondes par défaut	S'applique aux appels de <code>findChildrenInRelation()</code> . Attribut facultatif.
timeout-execute-plugin-command	Numéro ; 30 secondes par défaut	S'applique aux appels de <code>executePluginCommand()</code> . Attribut facultatif.

Tableau 1-14. Hiérarchie des éléments

Élément parent	Élément enfant
<finder-datasources>	Aucun

## Élément inventaire

L'élément <inventory> définit la racine de la liste hiérarchique pour le plug-in qui apparaît sur la vue **Inventaire** et les boîtes de dialogue de sélection d'objets du client Orchestrator.

L'élément <inventory> ne représente pas un objet au sein de l'application en plug-in, mais plutôt le plug-in lui-même en tant qu'objet de l'API de scripts d'Orchestrator.

L'élément <inventory> est facultatif. L'élément <inventory> possède l'attribut suivant.

Attributs	Valeur	Description
type	Un type d'objet Orchestrator	Type de l'élément <finder> qui représente la racine de la hiérarchie des objets. Attribut obligatoire.

Tableau 1-15. Hiérarchie des éléments

Élément parent	Éléments enfants
<module>	Aucun

## Élément Fonctions de recherche

L'élément <finders> constitue le conteneur des éléments <finder>.

L'élément <finders> est facultatif. L'élément <finders> ne dispose d'aucun attribut.

Tableau 1-16. Hiérarchie des éléments

Élément parent	Élément enfant
<module>	<finder>

## Élément Fonction de recherche

Dans le client Orchestrator, l'élément <finder> représente un type d'objet trouvé dans le plug-in.

L'élément `<finder>` identifie la catégorie Java qui définit les représentations de l'objet et de la fonction de recherche de l'objet. L'élément `<finder>` définit l'apparence de l'objet dans l'interface du client Orchestrator. Il identifie également l'objet de script que l'API de script Orchestrator définit pour représenter cet objet.

Les fonctions de recherche servent d'interface entre les formats d'objets utilisés par les différents types de technologies en plug-in.

L'élément `<finder>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<finder>`.

L'élément `<finder>` définit les attributs suivant :

Attributs	Valeur	Description
type	Un type d'objet Orchestrator	Type d'objet représenté par la fonction de recherche. Attribut obligatoire.
datasource	Attribut <code>&lt;finder-datasource name&gt;</code>	Identifie la catégorie Java qui définit l'objet en s'appuyant sur la source de données refid. Attribut obligatoire.
dynamic-finder	Méthode Java	Détermine une méthode personnalisée que vous pouvez implémenter dans une instance <code>IDynamicFinder</code> pour renvoyer l'ID et les propriétés d'une fonction de recherche par la programmation au lieu de la définir dans le fichier <code>vso.xml</code> . Attribut facultatif.
hidden	true ou false (par défaut)	Si true, masque la fonction de recherche dans le client Orchestrator. Attribut facultatif.
image	Chemin d'accès à un fichier graphique.	Icône en 16 x 16 pour représenter la fonction de recherche dans les listes hiérarchiques du client Orchestrator. Attribut facultatif.
java-class	Nom de la catégorie Java.	Catégorie Java qui définit l'objet que la fonction de recherche a trouvé et mappé vers un objet de script. Attribut facultatif.
script-object	Attribut <code>&lt;scripting-object type&gt;</code>	Le type <code>&lt;scripting-object&gt;</code> , le cas échéant, vers lequel cette fonction de recherche doit être mappée. Attribut facultatif.

Tableau 1-17. Hiérarchie des éléments

Élément parent	Éléments enfants
<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>

## Élément propriétés

L'élément <properties> est le conteneur pour les éléments <finder><property>.

L'élément <properties> est facultatif. L'élément <properties> ne dispose d'aucun attribut.

Tableau 1-18. Hiérarchie des éléments

Élément parent	Élément enfant
<finder>	<property>

## Élément propriété

L'élément <property> mappe les propriétés de l'objet trouvé avec des propriétés Java ou des appels de méthode.

Vous pouvez faire appel aux méthodes de la classe SDKFinderProperty lorsque vous mettez en œuvre la fabrique de plug-in pour obtenir les propriétés de l'implémentation de fabrique de plug-in à traiter.

Vous pouvez afficher ou masquer les propriétés de l'objet dans les vues du client Orchestrator. Vous pouvez aussi utiliser des énumérations pour définir les propriétés de l'objet.

L'élément <property> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <property>. L'élément <property> présente les attributs suivants.

Attributs	Valeur	Description
name	Nom de la fonction de recherche	Nom que FinderResult emploie pour stocker l'élément. Attribut obligatoire.
display-name	Nom de la fonction de recherche	Nom complet de la propriété. Attribut facultatif.

Attributs	Valeur	Description
bean-property	Nom de la propriété	Vous vous servirez de l'attribut bean-property pour identifier une propriété à obtenir au moyen des opérations get et set. Si vous identifiez une propriété nommée MyProperty, le plug-in définira les opérations getMyProperty et setMyProperty.  Vous définirez bean-property ou property-accessor, mais pas les deux. Attribut facultatif.
property-accessor	Méthode qui obtient une valeur de propriété auprès d'un objet	L'attribut property-accessor vous permet de définir une expression OGNL pour valider les propriétés d'un objet.  Vous définirez bean-property ou property-accessor, mais pas les deux. Attribut facultatif.
show-in-column	true (par défaut) ou false	Si la valeur est true, cette propriété apparaîtra dans le tableau des résultats du client Orchestrator. Attribut facultatif.
show-in-description	true (par défaut) ou false	Si la valeur est true, cette propriété apparaîtra dans la description de l'objet. Attribut facultatif.
hidden	true ou false (par défaut)	Si la valeur est true, cette propriété sera masquée dans tous les cas. Attribut facultatif.
linked-enumeration	Nom de l'énumération	Lie une propriété fonction de recherche à une énumération. Attribut facultatif.

**Tableau 1-19. Hiérarchie des éléments**

Élément parent	Éléments enfants
<properties>	Éléments enfants

## Élément relations

L'élément <relations> constitue le conteneur des éléments <finder><relation>.

L'élément <relations> est facultatif. L'élément <relations> ne dispose d'aucun attribut.

**Tableau 1-20. Hiérarchie des éléments**

Élément parent	Élément enfant
<finder>	<relation>

## Élément relation

L'élément <relation> définit le mode de relation des objets entre eux

C'est dans l'élément <relation> que vous définissez le nom de la relation.



L'élément `<relation>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<relation>`. L'élément `<relation>` présente les attributs suivants.

Attributs	Valeur	Description
name	Nom de la relation	Nom de cette relation. Attribut obligatoire.
type	Type d'objet Orchestrator	Type d'objet apparenté à un autre objet du fait de la relation. Attribut obligatoire.
cardinality	to-one ou to-many	Définit la relation entre les objets comme une relation de un à un ou de un à plusieurs. Attribut facultatif.

**Tableau 1-21. Hiérarchie des éléments**

Élément parent	Éléments enfants
<code>&lt;relations&gt;</code>	Aucun

## Élément id

L'élément `<id>` définit une méthode d'obtention d'un ID unique pour l'objet que la fonction de recherche identifie.

L'élément `<id>` est facultatif. L'élément `<id>` possède les attributs suivants.

Attributs	Valeur	Description
accessor	Nom de la méthode	L'attribut accessor vous permet de définir une expression OGNL pour valider les propriétés d'un objet. Attribut obligatoire.

**Tableau 1-22. Hiérarchie des éléments**

Élément parent	Éléments enfants
<code>&lt;finder&gt;</code>	Aucun

## Élément inventaire-enfants

L'élément `<inventory-children>` définit la hiérarchie des listes qui figurent les objets sur la vue **Inventaire** et les boîtes de dialogue de sélection d'objets du client Orchestrator.

L'élément `<inventory-children>` est facultatif. L'élément `<inventory-children>` ne possède aucun attribut.

**Tableau 1-23. Hiérarchie des éléments**

Élément parent	Élément enfant
<code>&lt;finder&gt;</code>	<code>&lt;relation-link&gt;</code>

## Élément lien-relation

L'élément `<relation-link>` définit les hiérarchies entre objets parent et enfant dans l'onglet **Inventaire**.

L'élément `<relation-link>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<relation-link>`. L'élément `<relation-link>` présente l'attribut suivant.

Type	Valeur	Description
name	Nom de la relation	refid à un nom de relation. Attribut obligatoire.

**Tableau 1-24. Hiérarchie des éléments**

Élément parent	Éléments enfants
<code>&lt;inventory-children&gt;</code>	Aucun

## Élément Événements

L'élément `<events>` constitue le conteneur des éléments `<trigger>` et `<gauge>`.

L'élément `<events>` peut inclure un nombre illimité de déclencheurs ou d'indicateurs.

L'élément `<events>` est facultatif. L'élément `<events>` ne dispose d'aucun attribut.

**Tableau 1-25. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Élément trigger

L'élément `<trigger>` déclare les déclencheurs que vous pouvez utiliser pour cet outil de recherche. Vous devez implémenter les méthodes `registerEventPublisher()` et `unregisterEventPublisher()` de `IPluginAdaptor` pour définir les déclencheurs.

L'élément `<trigger>` est facultatif. L'élément `<trigger>` présente l'attribut suivant.

Type	Valeur	Description
name	Nom du déclencheur	Nom pour ce déclencheur. Attribut obligatoire.

**Tableau 1-26. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Élément trigger-properties

L'élément `<trigger-properties>` constitue le conteneur des éléments `<trigger-property>`.

L'élément `<trigger-properties>` est facultatif. L'élément `<trigger-properties>` ne possède aucun attribut.

Tableau 1-27. Hiérarchie des éléments

Élément parent	Élément enfant
<trigger>	<trigger-property>

## Élément trigger-property

L'élément <trigger-property> définit les propriétés qui identifient un objet déclencheur.

L'élément <trigger-property> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <trigger-property>. L'élément <trigger-property> présente les attributs suivants.

Type	Valeur	Description
name	Nom du déclencheur	Nom attribué au déclencheur. Attribut facultatif.
display-name	Nom du déclencheur	Nom qui s'affiche dans le client Orchestrator. Attribut facultatif.
type	Type de déclencheur	Type d'objet définissant le déclencheur. Attribut obligatoire.

Tableau 1-28. Hiérarchie des éléments

Élément parent	Éléments enfants
<trigger-properties>	Aucun

## Élément Indicateur

L'élément <gauge> définit les indicateurs que vous pouvez utiliser pour cette fonction de recherche. Vous devez implémenter les méthodes `registerEventPublisher()` et `unregisterEventPublisher()` de `IPluginAdaptor` pour définir les indicateurs.

L'élément <gauge> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <gauge>.

L'élément <gauge> présente les attributs suivants.

Type	Valeur	Description
name	Nom de l'indicateur	Un nom pour l'indicateur. Attribut obligatoire.
min-value	Nombre	Seuil minimal. Attribut facultatif.
max-value	Nombre	Seuil maximal. Attribut facultatif.
unit	Type d'objet	Type d'objet définissant l'indicateur. Attribut obligatoire.
format	Chaîne	Format de la valeur surveillée. Attribut facultatif.

Tableau 1-29. Hiérarchie des éléments

Élément parent	Élément enfant
<events>	<description>

## Élément scripting-objects

L'élément `<scripting-objects>` constitue le conteneur des éléments `<object>`.

L'élément `<scripting-objects>` est facultatif. L'élément `<scripting-objects>` ne possède aucun attribut.

**Tableau 1-30. Hiérarchie des éléments**

Élément parent	Élément enfant
<code>&lt;module&gt;</code>	<code>&lt;object&gt;</code>

## Élément objet

L'élément `<object>` mappe les constructeurs, attributs et méthodes de la technologie en plug-in aux types d'objet JavaScript que l'API de scripts d'Orchestrator expose.

Pour les conventions de nommage des objets, consultez [Attribution de noms aux objets de plug-in](#).

L'élément `<object>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<object>`.

L'élément `<object>` présente les attributs suivants.

Type	Valeur	Description
<code>script-name</code>	Nom JavaScript	Nom de la classe de scripts. Doivent être uniques au niveau global. Attribut obligatoire.
<code>java-class</code>	Classe Java	Classe Java encapsulée par cette classe JavaScript. Attribut obligatoire.
<code>create</code>	true (par défaut) ou false	Si true, vous pouvez créer une nouvelle instance de cette classe. Attribut facultatif.
<code>strict</code>	true ou false (par défaut)	Si true, vous ne pouvez appeler que les méthodes que vous annotez ou déclarez dans le fichier <code>vso.xml</code> . Attribut facultatif.
<code>is-deprecated</code>	true ou false (par défaut)	Si true, l'objet est mappé avec une classe obsolète. Attribut facultatif.
<code>since-version</code>	String	Version depuis l'obsolescence de la classe Java. Attribut facultatif.

**Tableau 1-31. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Élément constructeurs

L'élément `<constructors>` constitue le conteneur des éléments `<object><constructor>`.

L'élément `<constructors>` est facultatif. L'élément `<constructors>` ne dispose d'aucun attribut.

**Tableau 1-32. Hiérarchie des éléments**

Élément parent	Élément enfant
<code>&lt;object&gt;</code>	<code>&lt;constructor&gt;</code>

## Élément constructeur

L'élément `<constructor>` définit une méthode constructeur. La méthode `<constructor>` produit de la documentation dans l'explorateur d'API.

L'élément `<constructor>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<constructor>`. L'élément `<constructor>` ne dispose d'aucun attribut.

**Tableau 1-33. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Élément Paramètres du constructeur

L'élément `<parameters>` constitue le conteneur des éléments `<constructor><parameter>`.

L'élément `<parameters>` est facultatif. L'élément `<parameters>` ne dispose d'aucun attribut.

**Tableau 1-34. Hiérarchie des éléments**

Élément parent	Élément enfant
<code>&lt;constructor&gt;</code>	<code>&lt;parameter&gt;</code>

## Élément Paramètre du constructeur

L'élément `<parameter>` définit les paramètres du constructeur.

L'élément `<parameter>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<parameter>`. L'élément `<parameter>` présente les attributs suivants.

Type	Valeur	Description
name	String	Nom du paramètre à utiliser dans la documentation sur l'API. Attribut obligatoire.
type	Type de paramètre Orchestrator	Type de paramètre à utiliser dans la documentation sur l'API. Attribut obligatoire.

Type	Valeur	Description
is-optional	true ou false	Si true, la valeur peut être Null. Attribut facultatif.
since-version	String	Version de la méthode. Attribut facultatif.

Tableau 1-35. Hiérarchie des éléments

Élément parent	Éléments enfants
<parameters>	Aucun

## Élément attributs

L'élément <attributes> constitue le conteneur des éléments <object><attribute>.

L'élément <attributes> est facultatif. L'élément <attributes> ne dispose d'aucun attribut.

Tableau 1-36. Hiérarchie des éléments

Élément parent	Élément enfant
<object>	<attribute>

## Élément attribut

L'élément <attribute> mappe les attributs d'une catégorie Java issue de la technologie en plug-in aux attributs JavaScript que le moteur JavaScript d'Orchestrator met à disposition.

L'élément <attribute> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <attribute>. L'élément <attribute> présente les attributs suivants.

Type	Valeur	Description
java-name	Attribut Java	Nom de l'attribut Java. Attribut obligatoire.
script-name	Objet JavaScript	Nom de l'objet JavaScript correspondant. Attribut obligatoire.
return-type	String	Type d'objet que cet attribut renvoie. Indiqué dans la documentation de l'explorateur d'API. Attribut facultatif.  <b>Note</b> Si le type de retour JavaScript est Properties, les implémentations Java sous-jacentes prises en charge seront java.util.HashMap et java.util.Hashtable.
read-only	true ou false	Si true, vous ne pouvez pas modifier cet attribut. Attribut facultatif.
is-optional	true ou false	Si true, ce champ peut être nul. Attribut facultatif.
show-in-api	true ou false	Si false, cet attribut ne figure pas dans la documentation de l'API. Attribut facultatif.

Type	Valeur	Description
is-deprecated	true ou false	Si true, l'objet mappe un attribut obsolète. Attribut facultatif.
since-version	Numéro	Version à laquelle l'attribut est devenu obsolète. Attribut facultatif.

**Tableau 1-37. Hiérarchie des éléments**

Élément parent	Éléments enfants
<attributes>	Aucun

## Élément méthodes

L'élément <methods> constitue le conteneur des éléments <object><method>.

L'élément <methods> est facultatif. L'élément <methods> ne dispose d'aucun attribut.

**Tableau 1-38. Hiérarchie des éléments**

Élément parent	Élément enfant
<object>	<method>

## Élément méthode

L'élément <method> mappe une méthode Java issue de la technologie en plug-in à une méthode JavaScript que le moteur JavaScript d'Orchestrator expose.

L'élément <method> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <method>.

L'élément <method> présente les attributs suivants.

Type	Valeur	Description
java-name	méthode Java	Nom de la signature de méthode Java avec les types d'argument entre parenthèses, par exemple : <code>getVms(DataStore)</code> . Attribut obligatoire.
script-name	méthode JavaScript	Nom de la méthode JavaScript correspondante. Attribut obligatoire.
return-type	Type d'objet en Java	Le type que cette méthode obtient. Attribut facultatif.  <b>Note</b> Si le type de retour JavaScript est <code>Properties</code> , les implémentations Java sous-jacentes prises en charge seront <code>java.util.HashMap</code> et <code>java.util.Hashtable</code> .
static	true ou false	Si true, cette méthode est statique. Attribut facultatif.
show-in-api	true ou false	Si false, cette méthode ne figure pas dans la documentation des API. Attribut facultatif.

Type	Valeur	Description
is-deprecated	true ou false	Si true, l'objet mappe une méthode obsolète. Attribut facultatif.
since-version	Numéro	Version à laquelle la méthode est devenue obsolète. Attribut facultatif.

**Tableau 1-39. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Exemple d'élément

L'élément <example> vous permet d'ajouter des exemples de codes aux méthodes JavaScript qui apparaissent dans la documentation de l'explorateur de l'API.

L'élément <example> est facultatif. L'élément <example> ne dispose d'aucun attribut.

**Tableau 1-40. Hiérarchie des éléments**

Élément parent	Éléments enfants
<method>	<ul style="list-style-type: none"> <li>■ &lt;code&gt;</li> <li>■ &lt;description&gt;</li> </ul>

## Élément de code

L'élément <code> fournit un exemple de code qui apparaît dans la documentation de l'explorateur de l'API.

Vous indiquez l'exemple de code entre les balises <code> et </code>. L'élément <code> est facultatif. L'élément <code> ne dispose d'aucun attribut.

**Tableau 1-41. Hiérarchie des éléments**

Élément parent	Éléments enfants
<example>	Aucun

## Élément paramètres de la méthode

L'élément <parameters> constitue le conteneur des éléments <method><parameter>.

L'élément <parameters> est facultatif. L'élément <parameters> ne dispose d'aucun attribut.

**Tableau 1-42.**

Élément parent	Élément enfant
<method>	<parameter>



## Élément paramètre de la méthode

L'élément `<parameter>` définit les paramètres d'entrée de la méthode.

L'élément `<parameter>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<parameter>`. L'élément `<parameter>` présente les attributs suivants.

Type	Valeur	Description
name	String	Nom du paramètre. Attribut obligatoire.
type	Type de paramètre Orchestrator	Type de paramètre. Attribut obligatoire.
is-optional	true ou false	Si true, la valeur peut être Null. Attribut facultatif.
since-version	String	Version de la méthode. Attribut facultatif.

Tableau 1-43. Hiérarchie des éléments

Élément parent	Élément enfant
<code>&lt;parameters&gt;</code>	Aucun

## Élément singleton

L'élément `<singleton>` crée un objet de script JavaScript en tant qu'instance singleton.

Les objets singleton se comportent de la même façon que les catégories Java statiques. Les objets singleton définissent les objets génériques que le plug-in doit utiliser plutôt que des instances spécifiques d'objets auxquels Orchestrator accède dans la technologie en plug-in. Par exemple, vous pouvez utiliser un objet singleton pour établir la connexion à la technologie en plug-in.

L'élément `<singleton>` est facultatif. L'élément `<singleton>` présente les attributs suivants.

Type	Valeur	Description
script-name	Objet JavaScript	Nom de l'objet JavaScript correspondant. Attribut obligatoire.
datasource	Objet Java	Objet Java source pour cet objet JavaScript. Attribut obligatoire.

Tableau 1-44. Hiérarchie des éléments

Élément parent	Élément enfant
<code>&lt;object&gt;</code>	Aucun

## Élément Énumérations

L'élément `<enumerations>` constitue le conteneur des éléments `<enumeration>`.

L'élément `<enumerations>` est facultatif. L'élément `<enumerations>` ne dispose d'aucun attribut.

Tableau 1-45. Hiérarchie des éléments

Élément parent	Élément enfant
<module>	<enumeration>

## Élément Énumération

L'élément <enumeration> définit les valeurs communes qui s'appliquent à tous les objets d'un certain type.

Si tous les objets d'un certain type ont besoin d'un certain attribut et que la plage de valeur de cet attribut est limitée, vous pouvez définir les valeurs en tant qu'entrées d'énumération. Par exemple, si un type d'objet nécessite un attribut `color` et que les seules couleurs disponibles sont le rouge, le bleu et le vert, vous pouvez définir trois entrées d'énumération pour déterminer ces trois valeurs de couleur. Vous devez définir ces entrées en tant qu'éléments enfants de l'élément Énumération.

L'élément <enumeration> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <enumeration>. L'élément <enumeration> présente l'attribut suivant.

Type	Valeur	Description
type	Type d'objet Orchestrator	Type d'énumération. Attribut obligatoire.

Tableau 1-46. Hiérarchie des éléments

Élément parent	Éléments enfants
<enumerations>	<ul style="list-style-type: none"> <li>■ &lt;url&gt;</li> <li>■ &lt;description&gt;</li> <li>■ &lt;entries&gt;</li> </ul>

## Élément Entrées

L'élément <entries> constitue le conteneur des éléments <enumeration><entry>.

L'élément <entries> est facultatif. L'élément <entries> ne dispose d'aucun attribut.

Tableau 1-47. Hiérarchie des éléments

Élément parent	Élément enfant
<enumeration>	<entry>

## Élément Entrée

L'élément <entry> fournit les valeurs des attributs d'énumération.

L'élément <entry> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <entry>.

L'élément <entry> présente les attributs suivants.

Type	Valeur	Description
id	Texte	Identifiant utilisé par les objets pour définir l'entrée d'énumération en tant qu'Attribut. Attribut obligatoire.
name	Texte	Nom de l'entrée. Attribut obligatoire.

**Tableau 1-48. Hiérarchie des éléments**

Élément parent	Éléments enfants
<entries>	Aucun

## Recommandations pour le développement de plug-ins Orchestrator

Vous pouvez améliorer certains aspects des plug-ins Orchestrator que vous développez si vous comprenez la structure et le contenu des plug-ins, et que vous savez comment éviter des problèmes spécifiques.

### ■ Méthodes de création de plug-ins Orchestrator

Vous pouvez vous appuyer sur différentes méthodes pour créer vos plug-ins Orchestrator. Vous pouvez commencer par créer un plug-in couche par couche ou vous pouvez créer toutes les couches du plug-in en même temps.

### ■ Types des plug-ins Orchestrator

L'utilisation de plug-ins vous permet d'intégrer à Orchestrator des bibliothèques généralistes, des utilitaires tels que XML ou SSH, ou des systèmes complets comme vCloud Director. Selon la technologie que vous intégrez à Orchestrator, les plug-ins peuvent être classés en plug-ins pour services, plug-ins généralistes et plug-ins pour systèmes.

### ■ Implémentation des plug-ins

Vous pouvez utiliser certaines pratiques et techniques utiles pour structurer vos plug-ins, implémenter les catégories Java et les objets JavaScript demandés, développer les workflows et les actions des plug-ins et rédiger la présentation d'un workflow.

### ■ Recommandations pour le développement de plug-ins Orchestrator

Lors du développement des différents composants de vos plug-ins Orchestrator, le respect de certaines pratiques vous permettra d'améliorer la qualité de ces plug-ins.

### ■ Documentation des chaînes et des API de l'interface utilisateur du plug-in

Veillez à suivre les règles validées de style et de format lorsque vous écrivez des chaînes d'interface utilisateur pour les plug-ins Orchestrator et la documentation de l'API associée.

## Méthodes de création de plug-ins Orchestrator

Vous pouvez vous appuyer sur différentes méthodes pour créer vos plug-ins Orchestrator. Vous pouvez commencer par créer un plug-in couche par couche ou vous pouvez créer toutes les couches du plug-in en même temps.

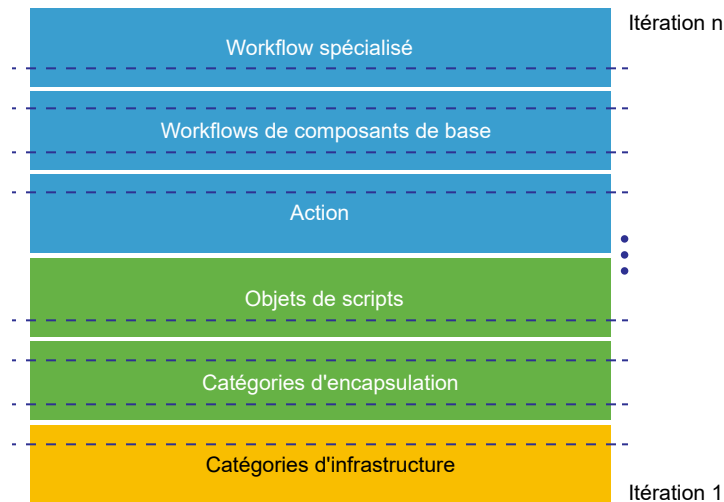
Pour des informations détaillées sur les couches de plug-ins, reportez-vous à [Structure d'un plug-in Orchestrator](#).

## Développement de plug-ins ascendant

Un plug-in peut être développé couche par couche dans le cadre d'une approche ascendante.

L'approche de développement ascendant permet de créer le plug-in couche par couche en commençant par les couches inférieures et en poursuivant par les couches supérieures. Lorsque cette approche est combinée à une approche interactive et itérative, une partie ou l'ensemble de chaque couche est fournie pour chaque itération. Au bout de N itérations, le plug-in est complètement terminé.

**Figure 1-5. Développement de plug-ins ascendant**



L'avantage du développement de plug-ins ascendant est qu'il est concentré sur chaque couche, l'une après l'autre.

Veuillez tenir compte des inconvénients suivants de l'approche de développement ascendant des plug-ins.

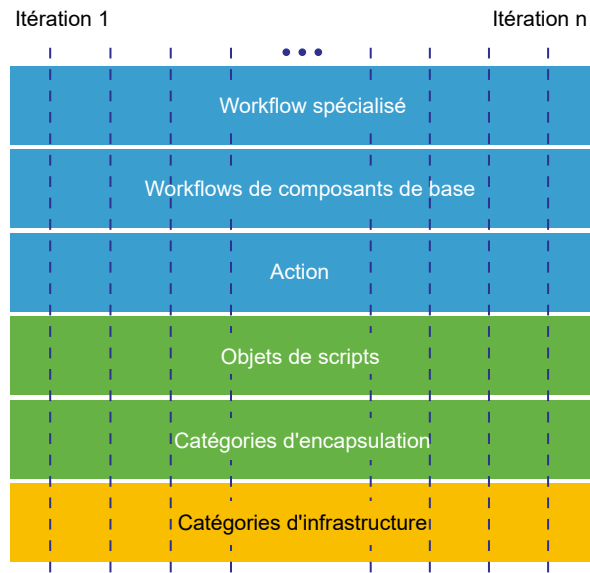
- La progression du développement de plug-ins est difficile à évaluer jusqu'à ce que certaines insertions soient terminées.
- Il n'est pas parfaitement compatible avec les pratiques de développement Agile.

Le processus de développement ascendant est considéré comme satisfaisant pour les petits plug-ins, avec un ensemble inexistant ou limité de catégories d'encapsulation, d'objets de script, d'actions ou de workflows.

## Développement de plug-ins de haut en bas

Un plug-in peut être créé par un fractionnement de haut en bas de fonctionnalités, dans une approche de développement de haut en bas.

Lorsque l'approche de haut en bas est associée à un processus de développement Agile, les nouvelles fonctionnalités sont incluses dans chaque itération. Ainsi, au bout de l'itération N, le plug-in est complètement implémenté.

**Figure 1-6. Développement de plug-ins de haut en bas**

L'approche de développement de plug-ins de haut en bas présente les avantages suivants.

- La progression du développement du plug-in est facile à observer dès la première itération, car les nouvelles fonctionnalités sont implémentées pour chaque itération et le plug-in peut être lancé et utilisé après chaque itération.
- Cette approche par fractions verticales des fonctionnalités permet d'observer clairement le degré de réussite et la définition de ce qui a été accompli, et facilite la communication entre les développeurs, le service de gestion des produits et les ingénieurs de l'assurance qualité (QA).
- Permet aux ingénieurs QA de lancer les tests et l'automatisation dès le début du processus de développement. Une telle approche permet la formulation de commentaires pertinents et réduit le temps de mise à exécution du projet.

L'inconvénient de l'approche du développement de plug-ins de haut en bas est que le développement a lieu sur plusieurs couches différentes en même temps.

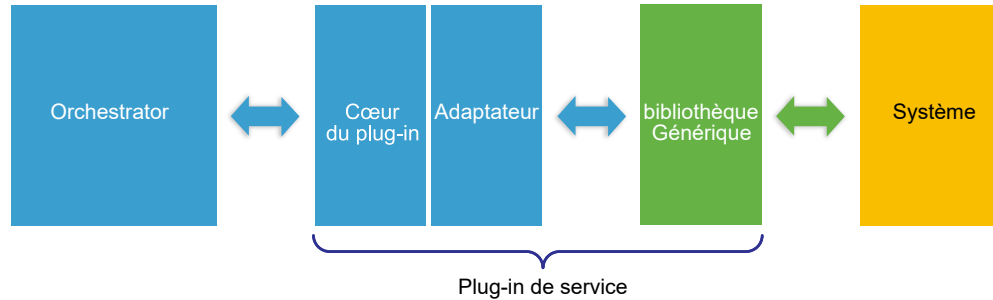
Il est conseillé d'appliquer le processus de développement de plug-ins de haut en bas pour la plupart des plug-ins. Ce processus convient aux plug-ins présentant des exigences de dynamique.

## Types des plug-ins Orchestrator

L'utilisation de plug-ins vous permet d'intégrer à Orchestrator des bibliothèques généralistes, des utilitaires tels que XML ou SSH, ou des systèmes complets comme vCloud Director. Selon la technologie que vous intégrez à Orchestrator, les plug-ins peuvent être classés en plug-ins pour services, plug-ins généralistes et plug-ins pour systèmes.

### Plug-ins pour services

Les plug-ins pour services, encore appelés plug-ins généralistes, confèrent des fonctionnalités qui peuvent être considérées comme des services dans Orchestrator.

**Figure 1-7. Architecture des plug-ins pour services**

Les plug-ins pour services exposent des bibliothèques ou utilitaires génériques à Orchestrator comme XML, SSH ou SOAP. Par exemple, les plug-ins suivants disponibles dans Orchestrator sont des plug-ins pour services.

<b>Plug-in JDBC</b>	Vous permet d'utiliser n'importe quelle base de données d'un workflow.
<b>Plug-in Mail</b>	Vous permet d'envoyer des e-mails dans un workflow.
<b>Plug-in SSH</b>	Vous permet d'ouvrir les connexions SSH et d'exécuter les commandes dans un workflow.
<b>Plug-in XML</b>	Vous permet de gérer les documents XML dans un workflow.

Les plug-ins pour services présentent les caractéristiques suivantes.

<b>Complexité</b>	Les plug-ins pour services présentent des niveaux de complexité de faibles à intermédiaires. Les plug-ins pour services exposent une bibliothèque spécifique ou une partie de bibliothèque dans Orchestrator afin de mettre à disposition une fonctionnalité précise. Par exemple, le plug-in XML ajoute l'implémentation d'un analyseur XML de modèle d'objet de documents (DOM) à l'API JavaScript d'Orchestrator.
<b>Taille</b>	Les plug-ins pour services sont relativement petits. Ils nécessitent le même ensemble de classes de base que les autres plug-ins, ainsi que d'autres classes qui proposent de nouveaux objets de script permettant d'ajouter de nouvelles fonctionnalités.
<b>Inventaire</b>	Les plug-ins pour services peuvent fonctionner avec un inventaire restreint d'objets ou sans inventaire du tout. Les plug-ins pour services présentent un modèle d'objet générique et peu volumineux ; ils ne sont donc pas obligés de faire apparaître ce modèle dans l'inventaire Orchestrator.

### Plug-ins pour systèmes

Les plug-ins pour systèmes connectent le moteur de workflows d'Orchestrator à un système externe afin que vous puissiez orchestrer ce système externe.

Exemples de plug-ins pour systèmes.

<b>Plug-in vCenter Server</b>	Vous permet de gérer les instances vCenter Server à l'aide de workflows.
<b>Plug-in vCloud Director</b>	Vous permet d'interagir avec l'installation vCloud Director au sein d'un workflow.
<b>Plug-in Cisco UCSM</b>	Vous permet d'interagir avec les entités Cisco au sein d'un workflow.

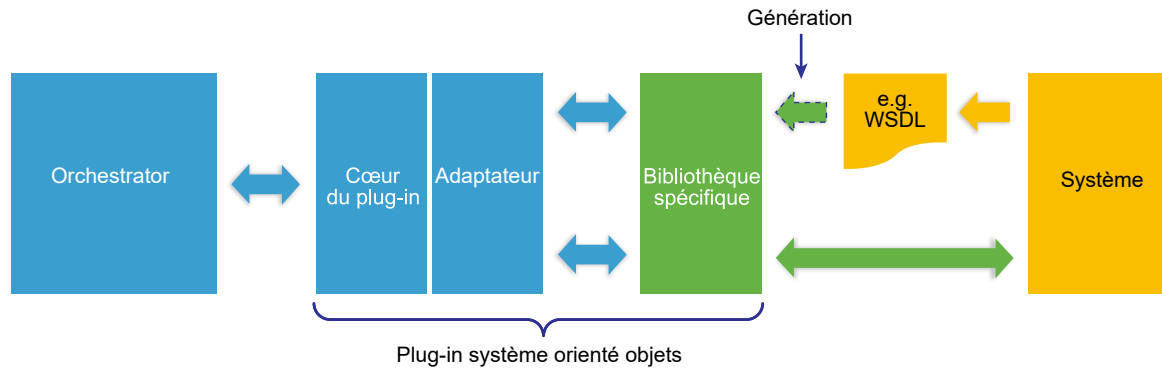
Principales caractéristiques des plug-ins pour systèmes.

<b>Complexité</b>	Les plug-ins pour systèmes présentent un degré de complexité supérieur à celui des plug-ins généralistes, car les technologies qu'ils exposent sont relativement complexes. Les plug-ins pour systèmes doivent représenter tous les éléments du système externe dans Orchestrator afin de pouvoir interagir et mettre leurs fonctionnalités à disposition dans Orchestrator. Si le système externe fournit un mécanisme d'intégration, vous pouvez l'utiliser afin d'exposer plus facilement les fonctionnalités du système dans Orchestrator. Cependant, outre leur fonction de représentation des éléments du système externe dans Orchestrator, les plug-ins pour systèmes peuvent également servir à améliorer l'évolutivité, à fournir un mécanisme de mise en cache, à gérer les événements et notifications, etc.
<b>Taille</b>	Les plug-ins pour systèmes sont de taille intermédiaire à importante. Les plug-ins pour systèmes demandent de nombreuses classes en plus de l'ensemble classique de classes, car ils proposent généralement un grand nombre d'objets de script. Les plug-ins pour systèmes peuvent nécessiter des systèmes d'aide et des classes auxiliaires supplémentaires pour interagir.
<b>Inventaire</b>	En règle générale, les plug-ins pour systèmes présentent un grand nombre d'objets que vous devez exposer correctement dans l'inventaire pour pouvoir facilement les localiser et les exploiter dans Orchestrator. Étant donné le grand nombre d'objets que les plug-ins pour systèmes doivent exposer, il est conseillé de créer un outil ou un processus auxiliaire pour générer automatiquement autant de code que possible pour chaque plug-in. Par exemple, le plug-in vCenter Server propose un outil de ce type.

## Plug-ins pour les systèmes orientés objets

Les systèmes orientés objets proposent un mécanisme d'interaction basés sur les objets et RPC.

Le modèle de système orienté objets le plus utilisé est le modèle de service Web s'appuyant sur SOAP. Les objets de ce modèle possèdent un ensemble d'attributs relatifs à l'état des objets et proposent des méthodes distantes que le système cible peut invoquer.

**Figure 1-8. Plug-ins pour les systèmes orientés objets**

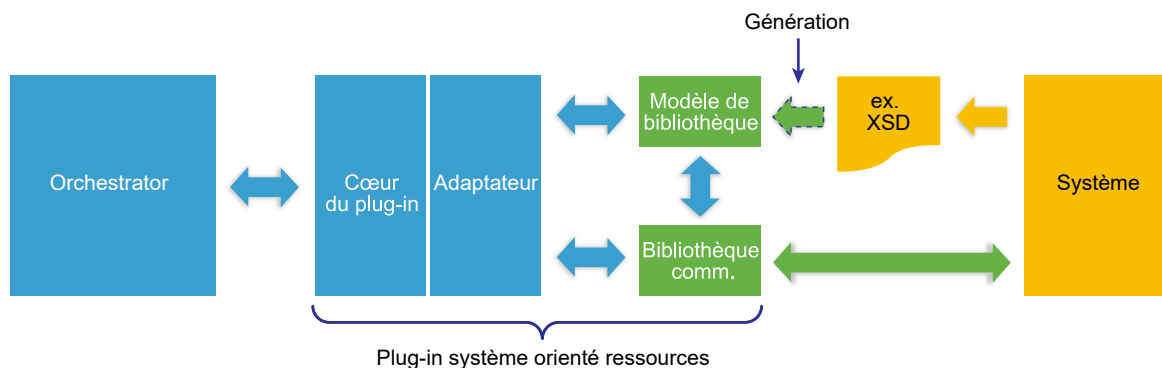
Il est conseillé de tenir compte des points suivants lors de l'implémentation de plug-ins pour les systèmes orientés objets.

- Si vous utilisez SOAP, vous pouvez utiliser le fichier WSDL pour générer un ensemble de catégories qui associent le modèle d'objet au mécanisme de communication.
- Ce modèle d'objet constitue quasiment le seul élément à exposer dans Orchestrator.

### Plug-ins pour les systèmes orientés ressources

Les systèmes orientés ressources confèrent un mécanisme d'interaction basé sur les ressources et les opérations simples qui utilisent les méthodes HTTP.

Le modèle le plus représentatif de système orienté ressources est le modèle REST associé, par exemple, à XML. Les objets de ce modèle possèdent un ensemble d'attributs relatifs à leur état. Pour invoquer des méthodes sur le système cible (mécanisme de communication), vous devez utiliser des méthodes HTTP standard telles que GET, POST, PUT, etc., et suivre certaines conventions.

**Figure 1-9. Plug-ins pour les systèmes orientés ressources**

Il est conseillé de tenir compte des points suivants lors du développement de plug-ins pour les systèmes orientés ressources.

- Si vous utilisez REST ou uniquement HTTP avec XML, vous obtenez un ou plusieurs fichiers de schéma pour pouvoir lire et écrire des messages. À partir de ces schémas, vous pouvez générer un



ensemble de classes qui définissent le modèle d'objet. Cet ensemble de classes définit uniquement l'état des objets, car les opérations sont définies de façon implicite avec les méthodes HTTP (par exemple, telles qu'elles sont définies dans le plug-in vCloud Director) ou de façon explicite avec certains messages XML (par exemple, le plug-in Cisco UCSM).

- Vous devez implémenter le mécanisme de communication dans un autre ensemble de catégories. Cet ensemble de classes définit un nouveau modèle d'objet qui interagit avec le modèle d'objet d'origine. Le modèle d'objet du mécanisme de communication se compose uniquement d'objets et de méthodes.
- Vous pouvez exposer aussi bien le modèle d'objet d'origine que le modèle d'objet pour le mécanisme de communication dans Orchestrator. Cette opération risque d'ajouter de la complexité ; cela dépend de la façon dont les deux modèles d'objet sont exposés et de si vous décidez de fusionner les objets liés des deux côtés (afin de simuler un système orienté objet) ou de les maintenir séparés.

## Implémentation des plug-ins

Vous pouvez utiliser certaines pratiques et techniques utiles pour structurer vos plug-ins, implémenter les catégories Java et les objets JavaScript demandés, développer les workflows et les actions des plug-ins et rédiger la présentation d'un workflow.

- [Structure des projets](#)

Vous pouvez appliquer une structure standard aux projets de vos plug-ins Orchestrator.

- [Opérations internes des projets](#)

Vous pouvez appliquer certaines approches lorsque vous implémentez votre plug-in ; par exemple, en mettant des objets en cache, en plaçant des objets en arrière-plan, en clonant des objets, etc. Ces approches sont susceptibles d'améliorer les performances de vos plug-ins, d'éviter les problèmes de concurrence et d'améliorer la réactivité du client Orchestrator.

- [Opérations internes aux workflows](#)

Vous pouvez implémenter un workflow pour surveiller les opérations à long terme réalisées par votre plug-in Orchestrator.

- [Workflows et actions](#)

Vous pouvez appliquer certaines recommandations pour faciliter le développement et l'utilisation des workflows.

- [Présentation d'un workflow](#)

Lorsque vous créez la présentation d'un workflow, vous devez appliquer une certaine structure et certaines règles.

## Structure des projets

Vous pouvez appliquer une structure standard aux projets de vos plug-ins Orchestrator.

Vous pouvez utiliser une structure Maven standard avec les modules de vos projets de plug-ins afin de clarifier l'emplacement de chaque élément fonctionnel.

**Tableau 1-49. Structure d'un projet de plug-in**

Module	Description
/myAwesomePlugin-plugin	Racine du projet du plug-in.
/o11nplugin-myAwesomePlugin	Module qui compose le fichier final DAR du plug-in.
/o11nplugin-myAwesomePlugin-config	Module qui comporte l'application Web de configuration des plug-ins. Il génère un fichier WAR standard.
/o11nplugin-myAwesomePlugin-core	Module comportant l'ensemble des classes qui implémentent les interfaces des plug-ins Orchestrator et leurs classes annexes. Il génère un fichier JAR standard.
/o11nplugin-myAwesomePlugin-model	Module contenant l'ensemble des classes qui vous aident à intégrer la technologie tierce à Orchestrator via le plug-in. Les classes ne doivent pas contenir de référence directe aux API des plug-ins Orchestrator standard.
/o11nplugin-myAwesomePlugin-package	Module important un fichier de module Orchestrator externe avec des actions et des workflows à intégrer au fichier DAR final du plug-in. Ce module est facultatif.

## Opérations internes des projets

Vous pouvez appliquer certaines approches lorsque vous implémentez votre plug-in ; par exemple, en mettant des objets en cache, en plaçant des objets en arrière-plan, en clonant des objets, etc. Ces approches sont susceptibles d'améliorer les performances de vos plug-ins, d'éviter les problèmes de concurrence et d'améliorer la réactivité du client Orchestrator.

### Mettre des objets en cache

Votre plug-in peut interagir avec un service distant, et cette interaction est rendue possible par les objets locaux qui représentent les objets distants du côté du service. Pour optimiser les performances du plug-in et la réactivité de l'UI Orchestrator, vous pouvez mettre les objets locaux en cache au lieu de les récupérer à chaque fois auprès du service distant. Vous pouvez tenir compte de la portée du cache : par exemple, un cache pour l'ensemble des plug-ins clients, un cache par utilisateur du plug-in et un cache par utilisateur du service tiers. Une fois implémenté, votre mécanisme de mise en cache est intégré à l'interface du plug-in pour rechercher et invalider des objets.

### Placer des objets en arrière-plan

Si vous devez faire apparaître des listes importantes d'objets dans l'inventaire du plug-in et que vous ne disposez d'aucune manière pour récupérer ces objets rapidement, vous pouvez les placer en arrière-plan. Vous pouvez placer les objets en arrière-plan avec, par exemple, des objets dans deux états : *fake* et *loaded*. Supposons que les objets *fake* soient extrêmement faciles à créer et fournissent un minimum d'informations à faire apparaître dans l'inventaire, comme le nom et l'ID. Il serait donc possible de toujours renvoyer les objets *fake* et, lorsque toutes les informations (l'objet réel) sont demandées, l'entité utilisatrice ou le plug-in pourrait appeler automatiquement une méthode *load* pour obtenir l'objet réel. Vous pouvez même configurer le processus de chargement des objets de manière qu'il démarre automatiquement après le renvoi des objets factices afin d'anticiper les actions de l'entité utilisatrice.

## Cloner des objets afin d'éviter les problèmes de concurrence

Si vous utilisez un cache pour votre plug-in, vous devez cloner des objets. L'utilisation d'un cache qui renvoie toujours la même instance d'un objet à chaque entité qui le demande peut provoquer des effets indésirables. Par exemple, l'entité A demande l'objet O et l'entité consulte l'objet dans l'inventaire avec l'ensemble de ses attributs. Au même moment, l'entité B demande également l'objet O, mais l'entité A exécute un workflow qui commence à modifier les attributs de l'objet O. À la fin de cette exécution, le workflow appelle la méthode `update` de l'objet pour mettre ce dernier à jour du côté du serveur. Si l'entité A et l'entité B obtiennent la même instance de l'objet O, l'entité A peut voir dans l'inventaire toutes les modifications que l'entité B réalise, avant même que les modifications soient envoyées du côté du serveur. Cela ne pose aucun problème si l'exécution se déroule normalement. Par contre, si l'exécution échoue, les attributs de l'objet O pour l'entité A ne sont pas rétablis. Dans ce cas, si le cache (les opérations `find` du plug-in) renvoie un clone de l'objet au lieu de la même instance à chaque fois, les entités utilisatrices consultent et modifient leur propre copie et les problèmes de concurrence sont évités, au moins dans Orchestrator.

## Notifier les modifications aux autres intervenants

Des problèmes peuvent survenir lorsque vous utilisez simultanément un objet en cache et un objet cloné. Le problème le plus important est que l'objet qui apparaît à l'entité risque de ne pas s'afficher dans sa dernière version disponible. Par exemple, si une entité affiche l'inventaire, les objets sont chargés une seule fois au même moment. Donc si une autre entité modifie certains objets, la première entité ne voit pas ces modifications. Pour éviter ce problème, vous pouvez utiliser les méthodes `PluginWatcher` et `IPluginPublisher` de l'API du plug-in Orchestrator afin de notifier les modifications apportées et que ces dernières apparaissent dans les autres instances des clients Orchestrator. Cette solution s'applique également à une instance unique du client Orchestrator lorsque les modifications d'un objet de l'inventaire affectent les autres objets. Ces modifications doivent également être notifiées. Les opérations sujettes aux notifications sont l'ajout, la mise à jour et la suppression d'objets lorsque ces objets ou certaines de leurs propriétés apparaissent dans l'inventaire.

## Activer la recherche d'un objet à tout moment

Vous devez implémenter la méthode `find` de l'interface `IPluginFactory` pour rechercher des objets uniquement par type et par ID. La méthode `find` peut être appelée directement après le redémarrage d'Orchestrator et la reprise d'un workflow.

## Simuler un service de requête si vous n'en possédez pas

Dans certains cas, le client Orchestrator peut demander l'utilisation de requêtes pour certains objets, ou leur affichage en liste ou en tableau plutôt qu'en arborescence, par exemple. Cela signifie que votre plug-in doit être en mesure d'envoyer des requêtes à tout moment pour certains ensembles d'objets. Si la technologie tierce propose un service de requête, vous devez adapter et utiliser ce service. Si ce n'est pas le cas, vous devez pouvoir simuler un service de requête malgré la complexité accrue et les performances réduites de la solution.

## Les méthodes de recherche ne doivent pas renvoyer d'exceptions d'exécution

Les méthodes de l'interface `IPluginFactory` qui implémentent les recherches dans le plug-in ne doivent pas générer d'exceptions d'exécution contrôlées ou non contrôlées. Cela risque de causer des échecs de type *erreur de validation* étonnants au cours de l'exécution d'un workflow. Par exemple, entre les deux nœuds d'un workflow, la méthode `find` est appelée si une sortie du premier nœud est une entrée du deuxième nœud. À ce moment, si l'objet est introuvable à cause d'une exception d'exécution, vous risquez d'obtenir uniquement une *erreur de validation* sans information complémentaire dans le client Orchestrator. La quantité d'informations apparaissant dans les fichiers journaux dépend de la façon dont le plug-in enregistre les exceptions.

## Opérations internes aux workflows

Vous pouvez implémenter un workflow pour surveiller les opérations à long terme réalisées par votre plug-in Orchestrator.

Vous pouvez implémenter un workflow pour la surveillance des opérations de longue durée telles que la surveillance des tâches. Ce workflow peut être basé sur des déclencheurs Orchestrator et des événements en attente. Vous devez considérer qu'un workflow bloqué en attente d'une tâche peut reprendre dès que le serveur Orchestrator démarre. Le plug-in doit être en mesure d'obtenir l'ensemble des informations requises pour reprendre correctement le processus de surveillance.

Le workflow de surveillance ou la tâche qu'il peut utiliser en interne doit fournir un mécanisme pour indiquer le taux d'interrogation et un délai d'expiration éventuel.

Le processus de débogage d'un code de script au sein d'un workflow n'est pas simple, notamment si le code n'appelle aucun code Java. C'est la raison pour laquelle, parfois, la seule option est d'utiliser les méthodes de journalisation proposées par les objets de script Orchestrator par défaut.

## Workflows et actions

Vous pouvez appliquer certaines recommandations pour faciliter le développement et l'utilisation des workflows.

### Commencer à développer des workflows en tant que blocs constitutifs

Un bloc constitutif peut être un workflow élémentaire qui nécessite seulement quelques paramètres d'entrée et renvoie une sortie simple. Si vous disposez d'un ensemble important de blocs constitutifs, vous pouvez créer facilement des workflows spécialisés et proposer un ensemble d'outils amélioré pour la création de workflows complexes.

### Créer des workflows spécialisés basés sur des composants plus petits

Si vous devez développer un workflow complexe avec plusieurs entrées et étapes internes, vous pouvez le fractionner en workflows et actions plus petits et plus simples de blocs constitutifs.

### Créer des actions dès que possible

Vous pouvez créer des actions pour ajouter de la flexibilité au développement des workflows.

- Pour créer facilement des objets ou des paramètres complexes pour les méthodes de script
- Pour éviter la répétition permanente des parties de code identiques
- Pour réaliser les validations de l'IU

## Les workflows doivent appeler des actions autant que possible

Les actions peuvent être appelées directement en tant que nœuds dans le schéma de workflow. Cette fonction évite de complexifier le schéma de workflow, car vous ne devez pas ajouter des composants de code de script pour appeler chaque action.

## Renseigner les informations demandées

Renseignez les informations de chaque élément d'un workflow ou d'une action.

- Fournissez une description du workflow ou de l'action.
- Fournissez une description des paramètres d'entrée.
- Fournissez une description des sorties.
- Fournissez une description des attributs des workflows.

## Mettre à jour régulièrement les informations relatives à la version

Lorsque vous versionnez des plug-ins, ajoutez des commentaires pertinents avec des informations telles que les mises à jour principales du plug-in, les détails d'implémentation importants, etc.

## Présentation d'un workflow

Lorsque vous créez la présentation d'un workflow, vous devez appliquer une certaine structure et certaines règles.

Utilisez les propriétés suivantes pour les entrées de workflow de la présentation du workflow.

**Tableau 1-50. Propriétés des entrées de workflow**

Propriétés	Utilisation
Show in Inventory	Utilisez cette propriété pour aider l'utilisateur à exécuter un workflow à partir de la vue Inventaire.
Specify a root object to be shown in the chooser	Utilisez cette propriété pour aider l'utilisateur à sélectionner des entrées. Si l'objet root peut être actualisé dans la présentation, qu'il représente un attribut ou qu'il est récupéré par une méthode d'objet, vous devez créer ou définir une action appropriée pour actualiser l'objet dans la présentation.
Maximum string length	Utilisez cette propriété pour les chaînes longues telles que les noms, les descriptions, les chemins d'accès des fichiers, etc.
Minimum string length	Utilisez cette propriété pour éviter les chaînes vides des outils de test.
Custom validation	Implémentez des validations autres que des validations simples avec des actions.

Organisez les entrées avec des étapes et affichez le groupe. Une telle organisation aide l'utilisateur à identifier et à distinguer l'ensemble des paramètres d'entrée d'un workflow.

## Recommandations pour le développement de plug-ins Orchestrator

Lors du développement des différents composants de vos plug-ins Orchestrator, le respect de certaines pratiques vous permettra d'améliorer la qualité de ces plug-ins.

**Tableau 1-51. Pratiques utiles pour l'implémentation des plug-ins**

Composant	Élément	Description
Général	Accéder à une API tierce	Dès que cela est possible, les plug-ins doivent fournir des méthodes simplifiées pour accéder à l'API tierce.
	Interface	Les plug-ins doivent fournir une interface cohérente et normalisée aux utilisateurs, même lorsque l'API ne respecte pas ce principe.
Action	Objets de scripts	Vous devez créer des actions pour chaque création, modification et suppression, ainsi que pour toutes les autres méthodes disponibles pour un objet de scripts.
	Description	La description d'une action doit décrire ce que fait une action et non comment elle fonctionne.
	Scripts	Lorsque vous utilisez les scripts pour obtenir les propriétés ou les méthodes d'un objet, vous pouvez vérifier si la valeur de l'objet est différente de null ou de undefined.
	Obsolescence	Si une action est obsolète, le comment ou la déclaration throw doit indiquer l'action de remplacement, ou l'action doit appeler une nouvelle action de remplacement pour que les solutions créées sur la version obsolète ne rencontrent pas de problème.
Workflow	Opérations de l'interface utilisateur dans la technologie orchestrée	Vous devez créer un workflow pour chaque opération disponible dans l'interface utilisateur de la technologie orchestrée.
	Description	La description d'un workflow doit décrire ce que fait ce workflow et non comment il fonctionne.
	Propriété de présentation mandatory input	Vous devez définir la propriété mandatory input pour toutes les entrées de workflow obligatoires.
	Propriété de présentation default value	Si vous développez un workflow qui configure une entité, la présentation du workflow doit charger les valeurs de configuration par défaut de cette entité. Par exemple, si vous développez un workflow nommé Configuration d'hôte, la présentation du workflow doit charger les valeurs par défaut de la configuration d'hôte.
	Propriété de présentation Show in inventory	Vous devez définir la propriété Show in inventory de manière à obtenir des workflows contextuels sur les objets d'inventaire.
	Propriété de présentation specify a root parameter	Vous devez utiliser cette propriété dans les workflows lorsqu'il n'est pas nécessaire de parcourir l'inventaire à partir de la racine de l'arborescence.
	Validation des workflows	Vous devez valider les workflows et corriger toutes les erreurs.
	Création d'objets	Tous les workflows qui créent un nouvel objet doivent renvoyer le nouvel objet en tant que paramètre de sortie.
	Obsolescence	Si un workflow est obsolète, le comment ou la déclaration throw doit indiquer le workflow de remplacement, ou le workflow obsolète doit appeler un nouveau workflow de remplacement pour que les solutions créées sur les anciennes versions ne rencontrent pas de problème.

**Tableau 1-51. Pratiques utiles pour l'implémentation des plug-ins (suite)**

Composant	Élément	Description
Inventaire	Déconnexion de l'hôte	Si votre inventaire comporte une connexion à un hôte et que cet hôte devient indisponible, vous devez indiquer que l'hôte est déconnecté. Vous pouvez effectuer cette opération soit en renommant l'objet root avec l'ajout de <code>- disconnected</code> , soit en supprimant l'arborescence d'objets sous cet objet, de la même manière que procède le plug-in vCloud Director.
	Propriété <code>Select value as list</code>	Un objet d'inventaire doit être sélectionnable en tant que <code>treeview</code> ou <code>list</code> .
	Gestionnaire d'hôtes	Si le plug-in implémente un objet host pour le système cible, alors un objet <code>root hostmanager</code> parent doit exister avec des propriétés d'ajout, de suppression ou de modification des propriétés des hôtes.
	Obtention ou mise à jour d'objets	Si un service de requête s'exécute sur la technologie orchestrée, vous devez l'utiliser pour obtenir les différents objets.
	Découverte des objets enfants	Si vous devez récupérer des objets enfants séparément, le processus de récupération doit être parallélisé et ne doit pas se bloquer à chaque erreur.
	Modification d'objets Orchestrator	Tous les workflows pouvant modifier l'état d'un élément dans l'inventaire doivent mettre l'inventaire à jour pour éviter que des objets restent à l'écart de la synchronisation.
	Modification d'objets externes	Vous pouvez utiliser un mécanisme de notification pour indiquer les modifications apportées à la technologie orchestrée suite à des opérations ayant eu lieu en dehors d'Orchestrator. Si de telles opérations ont entraîné la suppression d'objets de la technologie orchestrée, vous devez actualiser l'inventaire de manière appropriée afin d'éviter tout échec ou toute perte de données. Par exemple, si une machine virtuelle est supprimée de vCenter Server, le plug-in vCenter Server met à jour l'inventaire pour supprimer l'objet de la machine virtuelle supprimée.
Objet de script	Objet de l'outil de recherche	Les objets du système de recherche présentent des propriétés qu'il est possible d'utiliser pour différencier les objets. Il s'agit généralement des propriétés qui apparaissent dans l'interface utilisateur.
	Implémentation	La méthode <code>equals</code> doit être implémentée pour que l'opération <code>==</code> fonctionne sur le même objet. En effet, l'objet peut parfois présenter deux instances.
	Propriétés des objets des plug-ins	Les objets présentant des objets parents doivent implémenter une propriété <code>parent</code> .
	Propriétés des objets des plug-ins	Les objets possédant des objets enfants doivent implémenter des méthodes <code>GET</code> qui renvoient des tableaux d'objets enfants.
	Objets d'inventaire	<p>Les objets d'inventaire doivent pouvoir être recherchés avec <code>Server.find</code>.</p> <p>Tous les objets d'inventaire doivent pouvoir être mis en série afin d'être utilisés en tant qu'attributs d'entrée ou de sortie dans un workflow.</p>

**Tableau 1-51. Pratiques utiles pour l'implémentation des plug-ins (suite)**

Composant	Élément	Description
	Constructeur et méthodes	Dans la plupart des cas, les objets inscriptibles possèdent un constructeur ou doivent être renvoyés par d'autres attributs ou méthodes d'objets.
	ID objet	Les objets possédant un ID créé par un système externe doivent utiliser un ID interne afin de garantir qu'aucun doublon d'ID ne se produit lorsque vous orchestrez plusieurs serveurs.
	Rechercher des objets	Les méthodes <code>search</code> ou <code>find</code> doivent implémenter un filtre pour que le nom ou l'ID indiqué puisse être recherché. Cela évite d'effectuer une recherche dans tous les objets. Par exemple, le serveur Orchestrator possède une méthode <code>Server.FindForId</code> qui permet de rechercher un objet de plug-in par son ID. Afin de parvenir à cela, la méthode doit être implémentée pour chaque objet pouvant être recherché dans le plug-in.
	Déclencher	Si possible, des déclencheurs doivent être disponibles pour les objets qui subissent des modifications afin qu'Orchestrator puisse déclencher ses stratégies sur plusieurs événements. Par exemple, Orchestrator peut surveiller un déclencheur ou un événement dans le plug-in vCenter de l'objet Datacenter afin de pouvoir déterminer le moment où une nouvelle machine virtuelle est ajoutée, mise en service, désactivée, etc.
	Propriétés des objets	Les objets hébergés dans les autres plug-ins doivent posséder des propriétés qui leur permettent d'être facilement convertis d'un objet de plug-in en un autre. Par exemple, les objets des machines virtuelles doivent disposer d'un <code>moref</code> (ID de référence d'objet géré).
	Gestionnaire de session	Si vous vous connectez à un serveur distant susceptible de présenter une session différente, le plug-in doit implémenter une session partagée et une session par utilisateur.
Déclencher	Déclencher	Toutes les opérations et méthodes de blocage de longue durée doivent pouvoir démarrer de manière asynchrone dès qu'une tâche est renvoyée, et générer un événement de déclenchement une fois terminées.
Énumérations	Énumérations	Les énumérations d'un type donné doivent présenter un objet d'inventaire qui permet la sélection de différentes valeurs.
Journalisation	Journaux	Les méthodes doivent implémenter différents niveaux de journaux.
Contrôle de version	Version de plug-in	La version du plug-in doit respecter les normes et être mise à jour parallèlement à la mise à jour du plug-in.
Documentation sur l'API	Méthodes	Les méthodes sont décrites dans la documentation sur l'API et ne doivent jamais générer l'exception <code>no xyz method / property</code> sur un objet. Au contraire, les méthodes doivent renvoyer <code>null</code> lorsqu'aucune propriété n'est disponible et être documentées avec des détails.
	<code>vso.xml</code>	Tous les objets et toutes les méthodes et propriétés doivent être documentés dans <code>vso.xml</code> .



## Documentation des chaînes et des API de l'interface utilisateur du plug-in

Veillez à suivre les règles validées de style et de format lorsque vous écrivez des chaînes d'interface utilisateur pour les plug-ins Orchestrator et la documentation de l'API associée.

### Recommandations générales

- Utilisez les noms officiels des produits VMware impliqués dans le plug-in. Par exemple, utilisez les noms officiels des produits suivants et la terminologie VMware.

Terme correct	Ne pas utiliser
vCenter Server	VC ou vCenter
vCloud Director	vCloud

- Terminez toutes les descriptions de workflows avec un point. Par exemple, `Creates a new organization.` est une description de workflow.
- Utilisez un éditeur de texte doté d'un correcteur orthographique pour écrire les descriptions, puis déplacez ces dernières dans le plug-in.
- Assurez-vous que le nom du plug-in correspond exactement au nom du produit tiers approuvé auquel il est associé.

### Workflows et actions

- Écrivez des descriptions informatives. Une ou deux phrases suffisent pour la plupart des actions et des workflows.
- Les workflows spécialisés peuvent inclure des descriptions et des commentaires plus détaillés.
- Commencez les descriptions par un verbe, par exemple `Creates....` N'utilisez pas d'auto-référence telle que `This workflow creates.`
- Mettez un point à la fin des descriptions formées de phrases complètes.
- Décrivez les effets d'un workflow ou d'une action plutôt que ses modalités d'implémentation.
- Les workflows et les actions sont généralement inclus dans les dossiers et les modules. Incluez également une brève description de ces dossiers et de ces modules. Par exemple, un dossier de workflow peut présenter une description de type `Set of workflows related to vApp Template management.`

### Paramètres des workflows et des actions

- Commencez les descriptions des workflows et des actions par une phrase nominative descriptive, par exemple `Name of.` N'utilisez pas d'expressions telles que `It's the name of.`
- Ne mettez pas de point à la fin des descriptions des paramètres et des actions. Il ne s'agit pas de phrases complètes.

- Les paramètres d'entrée des workflows doivent présenter une étiquette affichant les noms appropriés dans la vue de présentation. Dans de nombreuses situations, vous pouvez associer les entrées connexes dans un groupe d'affichage. Par exemple, au lieu de créer deux entrées avec les étiquettes Nom de l'entreprise et Nom complet de l'entreprise, vous pouvez créer un groupe d'affichage avec l'étiquette Entreprise et placer les entrées Nom et Nom complet dans le groupe Entreprise.
- Pour les étapes et les groupes d'affichage, ajoutez également les descriptions et les commentaires qui apparaissent dans la présentation du workflow.

### API du plug-in

- La documentation de l'API se rapporte à l'ensemble de la documentation du fichier `vso.xml` et des fichiers Java source.
- Pour le fichier `vso.xml`, utilisez les mêmes règles pour les descriptions des objets de recherche et de script que pour les méthodes employées avec les workflows et les actions. Les descriptions des attributs des objets et des paramètres des méthodes utilisent les mêmes règles que pour les paramètres des workflows et des actions.
- Évitez les caractères spéciaux dans le fichier `vso.xml` et placez les descriptions à l'intérieur d'une balise `<![CDATA[insert your description here!]]>`.
- Utilisez le style Javadoc standard pour les fichiers Java source.

## Obtention des paramètres d'entrée auprès des utilisateurs lors du démarrage d'un workflow

Si un workflow nécessite des paramètres d'entrée, il ouvre une boîte de dialogue dans laquelle les utilisateurs saisissent les valeurs des paramètres d'entrée requis lorsqu'il s'exécute. Vous pouvez organiser le contenu et l'agencement, ou présentation, de cette boîte de dialogue dans l'onglet **Présentation** de l'éditeur de workflow.

La façon d'organiser les paramètres dans l'onglet **Présentation** se traduit dans la boîte de dialogue des paramètres d'entrée lorsque le workflow s'exécute.

L'onglet **Présentation** vous permet également d'ajouter des descriptions des paramètres utilisateur pour assister les utilisateurs lors de la saisie des paramètres. Dans l'onglet **Présentation**, vous avez aussi la possibilité de définir des propriétés et des contraintes sur les paramètres pour restreindre les paramètres que les utilisateurs fournissent. Si les paramètres que l'utilisateur fournit ne satisfont pas aux contraintes que vous définissez dans l'onglet **Présentation**, le workflow ne s'exécute pas.

### ■ [Création de la boîte de dialogue des paramètres d'entrée dans l'onglet Présentation](#)

L'onglet **Présentation** de l'éditeur de workflow vous permet de définir la mise en forme de la boîte de dialogue dans laquelle les utilisateurs saisissent les paramètres d'entrée.

## ■ Définition des propriétés du paramètre

Orchestrator vous permet de définir des propriétés pour qualifier les valeurs du paramètre d'entrée que les utilisateurs renseignent lorsqu'ils exécutent des workflows. Les propriétés de paramètre que vous définissez imposent des limites aux types et aux valeurs des paramètres d'entrée que les utilisateurs fournissent.

## Création de la boîte de dialogue des paramètres d'entrée dans l'onglet Présentation

L'onglet **Présentation** de l'éditeur de workflow vous permet de définir la mise en forme de la boîte de dialogue dans laquelle les utilisateurs saisissent les paramètres d'entrée.

L'onglet **Présentation** vous permet de regrouper les paramètres d'entrée en catégories et de définir l'ordre de ces catégories dans la boîte de dialogue des paramètres d'entrée.

### Descriptions de la présentation

Vous pouvez ajouter une description pour chaque paramètre ou groupe de paramètres qui apparaît dans la boîte de dialogue des paramètres d'entrée. La description fournit des informations aux utilisateurs pour les aider à saisir les paramètres d'entrée adéquats. Vous pouvez optimiser la mise en forme du texte descriptif en utilisant un formatage HTML.

### Définition des étapes d'entrée de la présentation

Par défaut, la boîte de dialogue des paramètres d'entrée répertorie les paramètres d'entrée nécessaires dans une seule liste. Pour aider les utilisateurs à saisir les paramètres d'entrée, vous pouvez définir des modes (appelés étapes d'entrée) dans l'onglet Présentation. Les étapes d'entrée regroupent les paramètres d'entrée de même nature. Les paramètres d'entrée d'une étape d'entrée apparaissent dans une section distincte de la boîte de dialogue lors de l'exécution du workflow.

### Définition des groupes d'affichage de la présentation

Chaque étape d'entrée dispose de ses propres nœuds. Ces nœuds sont appelés « groupes d'affichage ». Les groupes d'affichage définissent l'ordre d'apparition des zones de texte des paramètres d'entrée dans leur section de la boîte de dialogue. Vous pouvez définir les groupes d'affichage indépendamment des étapes d'entrée.

## Créer la présentation de la boîte de dialogue des paramètres d'entrée

Vous pouvez créer la présentation de la boîte de dialogue dans laquelle les utilisateurs définissent les paramètres d'entrée dans l'onglet **Présentation** de l'éditeur de workflow.

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Vérifiez que le workflow possède une liste définie de paramètres d'entrée.

## Procédure

- 1 Cliquez sur l'onglet **Présentation** de l'éditeur de workflow.

Par défaut, tous les paramètres du workflow apparaissent sous le même nœud **Présentation**, dans leur ordre de création.

- 2 Cliquez avec le bouton droit sur le nœud **Présentation**, puis sélectionnez **Créer une nouvelle étape**.

Un nœud **Nouvelle étape** apparaît sous le nœud **Présentation**.

- 3 Entrez un nom approprié pour cette étape, puis appuyez sur Entrée.

Ce nom apparaît en tant qu'en-tête de section dans la boîte de dialogue des paramètres d'entrée lorsque le workflow s'exécute.

- 4 Cliquez sur l'étape d'entrée, puis ajoutez une description dans l'onglet **Général** situé dans la partie inférieure de l'onglet **Présentation**.

La description apparaît dans la boîte de dialogue des paramètres d'entrée afin de fournir des informations aux utilisateurs et de les aider ainsi à saisir les paramètres d'entrée adéquats. Vous pouvez optimiser la mise en forme du texte descriptif en utilisant un formatage HTML.

- 5 Cliquez avec le bouton droit sur l'étape d'entrée que vous avez créée, puis sélectionnez **Créer un groupe d'affichage**.

Un nœud **Nouveau groupe** apparaît sous le nœud de l'étape d'entrée.

- 6 Entrez un nom approprié pour le groupe d'affichage, puis appuyez sur Entrée.

Ce nom apparaît en tant qu'en-tête de sous-section dans la boîte de dialogue des paramètres d'entrée lorsque le workflow s'exécute.

- 7 Cliquez sur le groupe d'affichage, puis ajoutez une description dans l'onglet **Général** situé dans la partie inférieure de l'onglet **Présentation**.

Cette description apparaît dans la boîte de dialogue des paramètres d'entrée. Vous pouvez optimiser la mise en forme du texte descriptif en utilisant un formatage HTML. Vous pouvez ajouter une valeur de paramètre à une description de groupe à l'aide d'une instruction OGNL, telle que `${#param}`.

- 8 Répétez les étapes précédentes jusqu'à ce que vous ayez créé toutes les étapes d'entrée et tous les groupes d'affichage qui doivent apparaître dans la boîte de dialogue des paramètres d'entrée lors de l'exécution du workflow.

- 9 Faites glisser les paramètres du nœud **Présentation** vers les étapes et les groupes de votre choix.

Vous avez créé la mise en forme de la boîte de dialogue dans laquelle les utilisateurs entrent les valeurs des paramètres d'entrée pendant l'exécution du workflow.

## Étape suivante

Vous devez maintenant définir les propriétés des paramètres.

## Définition des propriétés du paramètre

Orchestrator vous permet de définir des propriétés pour qualifier les valeurs du paramètre d'entrée que les utilisateurs renseignent lorsqu'ils exécutent des workflows. Les propriétés de paramètre que vous définissez imposent des limites aux types et aux valeurs des paramètres d'entrée que les utilisateurs fournissent.

Tous les paramètres peuvent posséder plusieurs propriétés. Les propriétés d'un paramètre d'entrée se définissent dans l'onglet **Propriétés** d'un paramètre donné sur l'onglet **Présentation**.

Les propriétés de paramètre valident les paramètres d'entrée et modifient la façon dont les zones de texte apparaissent dans la boîte de dialogue des paramètres d'entrée. Certaines propriétés de paramètre peuvent créer des dépendances entre les paramètres.

## Valeurs des propriétés de paramètre statiques et dynamiques

La valeur d'une propriété de paramètre peut être soit statique, soit dynamique. Les valeurs d'une propriété statique restent constantes. Si vous définissez la valeur d'une propriété comme statique, vous définissez ou sélectionnez la valeur de la propriété dans une liste que l'éditeur de workflow génère en fonction du type du paramètre.

Les valeurs d'une propriété dynamique dépendent de la valeur d'un autre paramètre ou attribut. Vous définissez les fonctions par lesquelles les propriétés dynamiques obtiennent des valeurs au moyen d'une expression OGNL (le langage de navigation par courbes d'objets). Si la valeur d'une propriété dynamique dépend de la valeur d'une autre valeur de propriété de paramètre et que cette autre valeur de propriété de paramètre change, l'expression OGNL recalcule et modifie la valeur dynamique de la propriété.

## Définir les propriétés de paramètre

Lorsqu'un workflow se lance, il valide les valeurs de paramètre d'entrée provenant des utilisateurs par rapport à toute propriété de paramètre que vous auriez définie.

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Vérifiez que le workflow possède une liste définie de paramètres d'entrée.

### Procédure

- 1 Dans l'éditeur de workflow, cliquez sur l'onglet **Présentation**.
- 2 Cliquez sur un paramètre dans l'onglet **Présentation**.

Les onglets **Général** et **Propriétés** du paramètre apparaissent au bas de l'onglet **Présentation**.



- 3 Cliquez sur l'onglet **Propriétés** du paramètre.
- 4 Cliquez avec le bouton droit dans l'onglet **Propriétés** et sélectionnez **Ajouter une propriété**.

Une boîte de dialogue s'ouvre qui affiche une liste des propriétés possibles pour le paramètre dont vous avez sélectionné le type.

- 5 Sélectionnez une propriété dans la liste figurant dans la boîte de dialogue, puis cliquez sur **OK**.



La propriété s'affiche alors dans l'onglet **Propriétés**.

- 6 Sous **Valeur**, affectez soit la valeur statique, soit la valeur dynamique pour la propriété en sélectionnant le symbole qui correspond dans le menu déroulant.

Option	Description
	Propriété statique
	Propriété dynamique

- 7 Si vous réglez la valeur de la propriété sur statique, vous sélectionnez la valeur de la propriété en fonction du type du paramètre pour lequel vous définissez les propriétés.
- 8 Si vous réglez la valeur de la propriété sur dynamique, vous définissez la fonction servant à obtenir la valeur de la propriété du paramètre à l'aide d'une expression OGNL.

L'éditeur de workflow facilite l'écriture d'expressions OGNL.

- Cliquez sur l'icône  pour obtenir la liste de tous les attributs et paramètres définis par le workflow auquel cette expression peut faire appel.
- Cliquez sur l'icône  pour obtenir la liste de toutes les actions dans l'API Orchestrator renvoyant un paramètre de sortie du type pour lequel vous définissez les propriétés.

Cliquer sur des éléments des listes proposées de paramètres et d'actions a pour effet de les ajouter à l'expression OGNL.

- 9 Cliquez sur **Enregistrer** au bas de l'éditeur de workflow.

Vous avez donc défini les propriétés des paramètres d'entrée du workflow.

### Étape suivante

Valider et déboguer un workflow

## Object Missing

This object is not available in the repository.

## Valeurs constantes prédéfinies pour les expressions OGNL

Vous pouvez utiliser des constantes prédéfinies lorsque vous créez des expressions OGNL afin d'obtenir les valeurs des propriétés de paramètres dynamiques.

Orchestrator définit les constantes suivantes à utiliser dans les expressions OGNL.

**Tableau 1-52. Valeurs constantes prédéfinies pour les expressions OGNL**

Valeur constante	Description
<code>\${#__current}</code>	Valeur actuelle de propriété de la validation personnalisée ou de propriété de l'expression correspondante
<code>\${#__username}</code>	Nom de l'utilisateur qui a démarré le workflow
<code>\${#__userdisplayname}</code>	Nom complet de l'utilisateur qui a démarré le workflow
<code>\${#__serverurl}</code>	URL contenant l'adresse IP du serveur à partir duquel l'utilisateur démarre le workflow. L'URL est constituée de l'adresse IP du serveur et d'un port de consultation, selon le format suivant :  <code>{IPserveur}:{portConsultation}</code>
<code>\${#__datetime}</code>	Date et heure actuelles
<code>\${#__date}</code>	Date actuelle, avec heure réglée sur 00:00:00
<code>\${#__timezone}</code>	Fuseau horaire actuel

## Demander des interactions utilisateurs pendant l'exécution d'un workflow

Au cours de son exécution, un workflow peut parfois demander des paramètres d'entrée supplémentaires provenant d'une source externe. Ces paramètres d'entrée peuvent provenir d'une autre application ou d'un autre workflow. L'utilisateur peut également les fournir directement.

Par exemple, si un événement se produit au cours de l'exécution d'un workflow, le workflow peut demander une interaction humaine pour décider l'action à entreprendre. Avant de continuer, le workflow attend soit que l'utilisateur réponde à la demande d'informations, soit que le temps d'attente atteigne la fin d'un éventuel délai d'expiration. Si le temps d'attente dépasse le délai d'expiration, le workflow renvoie une exception.

Les attributs par défaut des interactions utilisateurs sont `security.group` et `timeout.date`. Lorsque vous définissez l'attribut `security.group` sur un groupe d'utilisateurs LDAP précis, vous limitez l'autorisation de répondre à la demande d'interaction utilisateur aux membres de ce groupe.

Lorsque vous définissez l'attribut `timeout.date`, vous indiquez une date et une heure limites jusqu'auxquelles le workflow attendra les informations de la part de l'utilisateur. Vous pouvez définir une date absolue ou vous pouvez créer un élément de workflow en script pour calculer une durée à partir de l'heure de début de l'attente.

### Procédure

#### 1 (Facultatif) Ajouter une interaction utilisateur à un workflow

Vous pouvez demander des paramètres d'entrée à des utilisateurs au cours de l'exécution d'un workflow en ajoutant un élément de schéma **Interaction utilisateur** au workflow. Lorsqu'un workflow rencontre un élément **Interaction utilisateur**, il interrompt son exécution et attend que l'utilisateur lui fournisse les données demandées.

**2 (Facultatif) Définir l'attribut `security.group` de l'interaction utilisateur**

L'attribut `security.group` d'un élément d'interaction utilisateur indique les utilisateurs ou les groupes d'utilisateurs autorisés à répondre.

**3 (Facultatif) Définir l'attribut `timeout.date` sur une date absolue**

Vous pouvez définir l'attribut `timeout.date` d'une interaction utilisateur afin de déterminer combien de temps le workflow doit attendre la réponse de l'utilisateur à cette interaction.

**4 (Facultatif) Calculer un délai d'expiration relatif pour les interactions utilisateur**

Dans un objet `Date`, vous pouvez calculer une date et une heure relatives pour l'expiration de l'interaction utilisateur.

**5 (Facultatif) Définir l'attribut `timeout.date` sur une date relative**

Vous pouvez définir l'attribut `timeout.date` d'un élément d'**Interaction utilisateur** sur une date et une heure relatives en le connectant à un objet `Date`. Vous définissez l'objet dans une fonction en script.

**6 (Facultatif) Définir les entrées externes d'une interaction utilisateur**

Vous indiquez les informations que les utilisateurs doivent fournir lors d'une exécution de workflow sous forme de paramètres d'entrée d'une interaction utilisateur.

**7 (Facultatif) Définir le comportement d'exception dans l'interaction utilisateur**

Si un utilisateur ne fournit pas les paramètres d'entrée avant le délai d'expiration, l'interaction utilisateur renvoie une exception. Vous pouvez définir le comportement d'exception dans une fonction en script.

**8 (Facultatif) Créer la boîte de dialogue des paramètres d'entrée pour l'interaction utilisateur**

Lors de l'exécution du workflow, les utilisateurs indiquent les paramètres d'entrée dans une boîte de dialogue en accomplissant les mêmes étapes que lorsqu'il s'agit du premier démarrage d'un workflow.

**9 (Facultatif) Répondre à une demande d'interaction utilisateur**

Les workflows qui nécessitent des interactions utilisateurs au cours de leur exécution suspendent cette exécution jusqu'à ce que l'utilisateur fournisse l'information nécessaire ou jusqu'à ce qu'ils atteignent le délai d'expiration.

## Ajouter une interaction utilisateur à un workflow

Vous pouvez demander des paramètres d'entrée à des utilisateurs au cours de l'exécution d'un workflow en ajoutant un élément de schéma **Interaction utilisateur** au workflow. Lorsqu'un workflow rencontre un élément **Interaction utilisateur**, il interrompt son exécution et attend que l'utilisateur lui fournisse les données demandées.

### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.



- Ajoutez des éléments au schéma de workflow.

#### Procédure

- 1 Faites glisser un élément **Interaction utilisateur** vers la position appropriée dans le schéma du workflow.
- 2 Cliquez sur l'icône **Modifier** (✎) de l'élément **Interaction utilisateur**.
- 3 Indiquez un nom et une description pour l'interaction utilisateur dans l'onglet **Infos**, puis cliquez sur **Fermer**.
- 4 Cliquez sur **Enregistrer**.

Vous avez ajouté un élément d'interaction utilisateur à un workflow. Lorsque le workflow atteindra cet élément, il attendra les informations de l'utilisateur avant de reprendre son exécution.

#### Étape suivante

Définissez l'attribut `security.group` de l'interaction utilisateur afin de restreindre l'autorisation de répondre à un seul utilisateur ou à un groupe d'utilisateurs. Reportez-vous à [Définir l'attribut `security.group` de l'interaction utilisateur](#).

## Définir l'attribut `security.group` de l'interaction utilisateur

L'attribut `security.group` d'un élément d'interaction utilisateur indique les utilisateurs ou les groupes d'utilisateurs autorisés à répondre.

#### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.
- Ajoutez des éléments et une interaction utilisateur au schéma de workflow.
- Identifiez un groupe d'utilisateurs LDAP pour répondre à la demande d'interaction utilisateur.

#### Procédure

- 1 Cliquez sur l'icône **Modifier** (✎) de l'élément **Interaction utilisateur** dans le schéma de workflow.
- 2 Cliquez sur l'onglet **Attributs** de l'interaction utilisateur.
- 3 Cliquez sur **Non définis** pour que le paramètre source `security.group` définisse les utilisateurs autorisés à répondre à l'interaction.
- 4 (Facultatif) Sélectionnez **NUL** pour permettre à l'ensemble des utilisateurs de répondre à la demande d'interaction.
- 5 Pour limiter l'autorisation de répondre à un utilisateur ou à un groupe d'utilisateurs spécifique, cliquez sur **Créer un paramètre/attribut dans le workflow**.

La boîte de dialogue **Informations relatives aux paramètres** s'ouvre.

- 6 Nommez le paramètre.
- 7 Sélectionnez **Créer un ATTRIBUT de workflow avec le même nom** pour créer l'attribut `LdapGroup` dans le workflow.
- 8 Cliquez sur **Non défini** pour que la valeur de paramètre ouvre la zone de sélection **LdapGroup**.
- 9 Dans la zone de texte **Filtre**, saisissez le nom du groupe d'utilisateurs LDAP.
- 10 Sélectionnez le groupe d'utilisateurs LDAP dans la liste, puis cliquez sur **Sélectionner**.

Par exemple, la sélection du groupe **Administrateurs** signifie que seuls les membres de ce groupe peuvent répondre à cette demande d'interaction utilisateur.

Vous avez limité l'autorisation de répondre à la demande d'interaction utilisateur.

- 11 Cliquez sur **OK** pour fermer la boîte de dialogue **Informations relatives aux paramètres**.

Vous avez défini l'attribut `security.group` de l'interaction utilisateur.

### Étape suivante

Définissez l'attribut `timer.date` pour configurer le délai d'expiration de l'interaction utilisateur.

- Pour définir le délai d'expiration sur une date et une heure absolues, reportez-vous à [Définir l'attribut `timeout.date` sur une date absolue](#).
- Pour créer une fonction qui calcule un délai d'expiration en fonction de la date et de l'heure du moment, reportez-vous à [Calculer un délai d'expiration relatif pour les interactions utilisateur](#).

## Définir l'attribut `timeout.date` sur une date absolue

Vous pouvez définir l'attribut `timeout.date` d'une interaction utilisateur afin de déterminer combien de temps le workflow doit attendre la réponse de l'utilisateur à cette interaction.

Vous pouvez définir une date et une heure absolues l'objet `Date`. Le workflow en attente de l'interaction utilisateur expire à la date et à l'heure indiquées en présentant l'état `Failed`. Par exemple, vous pouvez indiquer que l'interaction utilisateur expirera le 12 février à midi. Pour calculer un délai d'expiration relatif à la date et à l'heure du moment, reportez-vous à [Calculer un délai d'expiration relatif pour les interactions utilisateur](#).

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Ajoutez un élément d'interaction utilisateur au schéma de workflow.
- Définissez l'attribut `security.group` de l'interaction utilisateur.

### Procédure

- 1 Cliquez sur l'icône **Modifier** (✎) de l'élément **Interaction utilisateur** dans le schéma de workflow.
- 2 Cliquez sur l'onglet **Attributs** de l'interaction utilisateur.

- 3 Cliquez sur **Non définis** pour que le paramètre source `timeout.date` définisse la valeur du délai d'expiration.
- 4 (Facultatif) Sélectionnez **NUL** pour que l'interaction utilisateur permette au workflow d'attendre la réponse de l'utilisateur sans limite de temps.
- 5 Cliquez sur **Créer le paramètre/l'attribut dans le workflow** pour indiquer que le workflow doit échouer après son délai d'expiration.  
  
La boîte de dialogue **Informations relatives aux paramètres** s'ouvre.
- 6 Nommez le paramètre.
- 7 Sélectionnez **Créer un ATTRIBUT de workflow avec le même nom** pour créer un attribut `Date` dans le workflow.
- 8 Cliquez sur **Non définie** pour le paramètre **Valeur**.
- 9 Utilisez le calendrier pour sélectionner une date et une heure absolues jusqu'auxquelles le workflow doit attendre la réponse de l'utilisateur.
- 10 Cliquez sur **OK** pour fermer le calendrier.
- 11 Cliquez sur **OK** pour fermer la boîte de dialogue **Informations relatives aux paramètres**.

Vous avez défini l'attribut `timeout.date` sur une date absolue. Le workflow expire si l'utilisateur ne répond pas à l'interaction avant la date et l'heure fixées.

#### Étape suivante

Définissez les paramètres d'entrée externes que l'interaction utilisateur demande à l'utilisateur. Reportez-vous à [Définir les entrées externes d'une interaction utilisateur](#).

## Calculer un délai d'expiration relatif pour les interactions utilisateur



Dans un objet `Date`, vous pouvez calculer une date et une heure relatives pour l'expiration de l'interaction utilisateur.

Vous pouvez définir une date et une heure absolues dans un objet `Date`. La demande d'interaction utilisateur expire à la date et à l'heure indiquées. Vous pouvez également créer un élément de workflow qui calcule et génère un objet `Date` relatif selon une fonction que vous définissez. Par exemple, vous pouvez créer un objet `Date` relatif qui ajoute 24 heures à l'heure actuelle.

#### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Ajoutez un élément d'interaction utilisateur au schéma de workflow.
- Définissez l'attribut `security.group` de l'interaction utilisateur.

## Procédure

- 1 Faites glisser un élément **Tâche inscriptible** du menu **Générique** et déposez-le sur le schéma d'un workflow, à la position qui précède l'élément réclamant l'objet Date relative pour son attribut `timeout.date`.
- 2 Cliquez sur l'icône **Modifier** () de l'élément **Tâche inscriptible** du schéma de workflow.
- 3 Renseignez le nom et la description de l'élément de workflow en script dans l'onglet de propriétés **Infos**.
- 4 Cliquez sur l'onglet de propriétés **OUT** et cliquez sur l'icône **Créer une liaison avec le paramètre/attribut du workflow** ()
- 5 Cliquez sur **Créer un paramètre/attribut dans le workflow** pour créer un attribut de workflow.
  - a Nommez l'attribut `timerDate`.
  - b Sélectionnez Date dans la liste des types d'attribut.
  - c Sélectionnez **Créer un ATTRIBUT de workflow du même nom**.
  - d Laissez la valeur de l'attribut définie sur **Non définie** car c'est la fonction en script qui renseignera cette valeur.
  - e Cliquez sur **OK**.
- 6 Cliquez sur l'onglet **Script** pour l'élément de workflow en script.
- 7 Définissez une fonction pour calculer et générer un objet Date nommé `timerDate` sur la zone de scripts de l'onglet **Script**.

Vous pouvez, par exemple, créer un objet Date en mettant en œuvre la fonction JavaScript suivante, dans laquelle le délai d'expiration est un compte à rebours relatif en millisecondes.

```
timerDate = new Date();
System.log( "Current date : '" + timerDate + "'" );
timerDate.setTime( timerDate.getTime() + (86400 * 1000) );
System.log( "Timer will expire at '" + timerDate + "'" );
```

L'exemple de fonction JavaScript qui précède définit un objet Date qui obtient la date et l'heure actuelles au moyen de la méthode `getTime` et y ajoute 86 400 000 millisecondes ou 24 heures. L'élément **Tâche inscriptible** génère comme paramètre de sortie cette valeur.

- 8 Cliquez sur **Fermer**.
- 9 Cliquez sur **Enregistrer**.

Vous avez créé une fonction qui calcule une date et une heure relatives en fonction de la date et de l'heure du moment, et qui génère un objet Date. Un élément d'**Interaction utilisateur** peut recevoir cet objet Date en tant que paramètre d'entrée pour définir la durée pendant laquelle il doit attendre l'entrée de l'utilisateur. Lorsque le workflow parvient à l'élément de l'**Interaction utilisateur**, il suspend son exécution et attend soit que l'utilisateur fournisse les informations requises, soit les 24 heures à l'issue desquelles il expirera.

## Étape suivante

Vous devez relier l'objet `Date` au paramètre `timeout.date` de l'élément **Interaction utilisateur**. Reportez-vous à [Définir l'attribut `timeout.date` sur une date relative](#).

## Définir l'attribut `timeout.date` sur une date relative

Vous pouvez définir l'attribut `timeout.date` d'un élément d'**Interaction utilisateur** sur une date et une heure relatives en le connectant à un objet `Date`. Vous définissez l'objet dans une fonction en script.

Si vous créez un objet `Date` relatif dans une fonction en script, vous pouvez connecter l'attribut `timeout.date` de l'interaction utilisateur à cet objet `Date`. Par exemple, si vous connectez l'attribut `timeout.date` à un objet `Date` qui ajoute 24 heures à l'heure du moment, l'interaction utilisateur expire au bout de 24 heures d'attente.

### Conditions préalables

- Add a user interaction element to the workflow schema.
- Set the `security.group` attribute for the user interaction.
- Créez une fonction en script qui calcule une date et une heure relatives, puis qui les encapsule dans un objet `Date` du workflow. Reportez-vous à [Calculer un délai d'expiration relatif pour les interactions utilisateur](#).

### Procédure

- 1 Cliquez sur l'icône **Modifier** (✎) de l'élément **Interaction utilisateur** dans le schéma de workflow.
- 2 Cliquez sur l'onglet **Attributs** de l'interaction utilisateur.
- 3 Cliquez sur **Non définis** pour que le paramètre source `timeout.date` définisse la valeur du délai d'expiration.
- 4 Sélectionnez l'objet `Date` qui encapsule une date et une heure relatives que vous avez définies dans une fonction en script, puis cliquez sur **Sélectionner**.

Vous avez défini l'attribut `timeout.date` sur une date et une heure relatives calculées par une fonction en script.

## Étape suivante

Définissez les paramètres d'entrée externes que l'interaction utilisateur demande à l'utilisateur. Reportez-vous à [Définir les entrées externes d'une interaction utilisateur](#).

## Définir les entrées externes d'une interaction utilisateur



Vous indiquez les informations que les utilisateurs doivent fournir lors d'une exécution de workflow sous forme de paramètres d'entrée d'une interaction utilisateur.

Lorsqu'un workflow atteint un élément d'interaction utilisateur, il attend qu'un utilisateur fournisse les informations requises par l'interaction utilisateur sous forme de paramètres d'entrée.

## Conditions préalables

- Ajoutez un élément d'interaction utilisateur au schéma de workflow.
- Set the `security.group` attribute for the user interaction.
- Définir l'attribut `timer.date` pour l'interaction utilisateur

## Procédure

- 1 Cliquez sur l'icône **Modifier** () de l'élément **Interaction utilisateur** dans le schéma de workflow.
- 2 Cliquez sur l'onglet **Entrées externes**.
- 3 Cliquez sur l'icône **Relier au paramètre/à l'attribut du workflow** () afin de définir les paramètres que l'utilisateur doit fournir dans l'interaction utilisateur.
- 4 (Facultatif) Si vous avez déjà défini les paramètres d'entrée dans le workflow, sélectionnez les paramètres dans la liste proposée.
- 5 Cliquez sur **Créer un paramètre/attribut dans le workflow** afin de créer un attribut de workflow à relier au paramètre d'entrée fourni par l'utilisateur.
- 6 Nommez le paramètre de façon appropriée.
- 7 Sélectionnez le type de paramètre d'entrée dans la liste de types en recherchant un type d'objet dans la zone **Filtrer**.  
  
Par exemple, si l'interaction utilisateur exige que l'utilisateur indique une machine virtuelle en tant que paramètre d'entrée, sélectionnez `VC:VirtualMachine`.
- 8 Sélectionnez **Créer un ATTRIBUT de workflow avec le même nom** afin de relier le paramètre d'entrée fourni par l'utilisateur à un nouvel attribut du workflow.
- 9 Laissez la valeur du paramètre définie sur **Non définie**.  
  
L'utilisateur indique cette valeur lorsqu'il répond à l'interaction utilisateur lors de l'exécution du workflow.
- 10 Cliquez sur **OK** pour fermer la boîte de dialogue **Informations relatives aux paramètres**.

Vous avez défini les paramètres d'entrée que l'utilisateur doit fournir lors d'une interaction utilisateur.

## Étape suivante

Définir le comportement d'exception si l'interaction utilisateur rencontre une erreur. Reportez-vous à [Définir le comportement d'exception dans l'interaction utilisateur](#).

## Définir le comportement d'exception dans l'interaction utilisateur

Si un utilisateur ne fournit pas les paramètres d'entrée avant le délai d'expiration, l'interaction utilisateur renvoie une exception. Vous pouvez définir le comportement d'exception dans une fonction en script.

Si vous ne définissez pas l'action que le workflow doit entreprendre lorsque l'interaction utilisateur expire, le workflow se termine avec l'état `Failed`. La définition d'un comportement d'exception est une bonne pratique de développement de workflows.

### Conditions préalables

- Ajoutez un élément d'interaction utilisateur au schéma de workflow.
- Définissez les attributs `security.group` et `timer.date` pour l'interaction utilisateur.
- Définissez les paramètres d'entrée externes de l'interaction utilisateur.

### Procédure

- 1 Cliquez sur l'icône **Modifier** (✎) de l'élément **Interaction utilisateur** dans le schéma de workflow.
- 2 Cliquez sur l'onglet **Exception**.
- 3 Cliquez sur **Non définie** pour la liaison de l'exception de sortie.
- 4 Cliquez sur **Créer un paramètre/attribut dans le workflow** afin de créer un attribut d'exception à relier à l'interaction utilisateur.

La boîte de dialogue **Informations relatives aux paramètres** s'ouvre.

- 5 Créez un attribut `errorCode`.

Par défaut, un attribut `errorCode` présente les propriétés suivantes :

- Nom : **errorCode**
- Type : chaîne
- Créer : **Créer un ATTRIBUT de workflow avec le même nom**
- Valeur : tapez un message d'erreur approprié.

- 6 Cliquez sur **OK** pour fermer la boîte de dialogue **Informations relatives aux paramètres**.
- 7 Faites glisser un élément de tâche inscriptible dans l'élément d'interaction utilisateur du schéma de workflow.

Une flèche en pointillés rouge représentant le lien d'exception s'affiche entre les deux éléments. L'élément de tâche inscriptible se lie automatiquement à l'attribut `errorCode` à partir de l'interaction utilisateur.

- 8 Double-cliquez sur l'élément de tâche inscriptible et indiquez un nom approprié.

Par exemple, **Délai d'expiration du journal**.

- 9 L'onglet **Scripts** de l'élément de tâche inscriptible vous permet d'écrire une fonction JavaScript pour gérer l'exception.

Par exemple, pour enregistrer le délai d'expiration dans le journal Orchestrator, rédigez la fonction suivante :

```
System.log("No response from user. Timed out.");
```

**10** Reliez l'élément de tâche inscriptible qui gère les exceptions à l'élément qui le suit dans le workflow.

Par exemple, reliez l'élément de tâche inscriptible à l'élément **Générer une exception** afin de terminer le workflow avec une erreur.

Vous avez défini le comportement d'exception si l'interaction utilisateur expire.

### Étape suivante

Créez la boîte de dialogue dans laquelle les utilisateurs indiquent les paramètres d'entrée. Reportez-vous à [Créer la boîte de dialogue des paramètres d'entrée pour l'interaction utilisateur](#).

## Créer la boîte de dialogue des paramètres d'entrée pour l'interaction utilisateur

Lors de l'exécution du workflow, les utilisateurs indiquent les paramètres d'entrée dans une boîte de dialogue en accomplissant les mêmes étapes que lorsqu'il s'agit du premier démarrage d'un workflow.

Vous devez créer la mise en forme de la boîte de dialogue dans l'onglet **Présentation** de l'élément d'interaction utilisateur et non pas dans l'onglet **Présentation** de l'ensemble du workflow. L'onglet **Présentation** de l'ensemble du workflow permet de créer la mise en forme de la boîte de dialogue des paramètres d'entrée qui apparaît lorsque vous démarrez un workflow. L'onglet **Présentation** de l'élément d'interaction utilisateur permet de créer la mise en forme de la boîte de dialogue des paramètres d'entrée qui s'ouvre lorsqu'un workflow arrive au niveau d'un élément d'interaction utilisateur au cours de son exécution.

### Conditions préalables

- Ajoutez un élément d'interaction utilisateur au schéma de workflow.
- Définissez les attributs `security.group` et `timer.date` de l'interaction utilisateur.
- Définissez les paramètres d'entrée externes de l'interaction utilisateur.
- Définissez le comportement d'exception.

### Procédure

- 1 Cliquez sur l'icône **Modifier** (✎) de l'élément **Interaction utilisateur** dans le schéma de workflow.
- 2 Cliquez sur l'onglet **Présentation** de l'élément d'interaction utilisateur.

L'onglet **Présentation** affiche les paramètres d'entrée externes que vous avez créés pour l'interaction utilisateur.

- 3 (Facultatif) Cliquez avec le bouton droit sur le nœud **Présentation** de l'onglet **Présentation**, puis sélectionnez **Créer une étape**.

Les étapes vous permettent de créer des sections dans la boîte de dialogue, avec des descriptions et des en-têtes sous lesquels vous pouvez organiser les paramètres d'entrée.



- 4 (Facultatif) Cliquez avec le bouton droit sur le nœud **Présentation** de l'onglet **Présentation**, puis sélectionnez **Créer un groupe d'affichage**.

Les groupes d'affichage vous permettent de définir l'ordre selon lequel les paramètres d'entrée s'affichent dans les étapes, ainsi que d'ajouter des en-têtes secondaires et des instructions à la boîte de dialogue.

- 5 Cliquez sur un paramètre d'entrée dans la liste et ajoutez une description dans l'onglet **Général** de ce paramètre.

Le texte descriptif que vous saisissez apparaît en tant qu'étiquette dans la boîte de dialogue des paramètres d'entrée afin de renseigner l'utilisateur sur les informations à fournir lorsqu'il répond aux interactions.

- 6 Définissez les propriétés des paramètres d'entrée.

Les propriétés des paramètres d'entrée vous permettent de qualifier les valeurs des paramètres d'entrée que les utilisateurs peuvent indiquer et de déterminer les valeurs des paramètres de façon dynamique à l'aide d'expressions OGNL.

- 7 Cliquez sur **Enregistrer et fermer** pour fermer l'éditeur du workflow.

Vous avez créé la boîte de dialogue des paramètres d'entrée dans laquelle les utilisateurs indiquent les paramètres d'entrée à utiliser pour répondre à une interaction utilisateur au cours de l'exécution d'un workflow.

#### Étape suivante

Pour en savoir plus sur la création des étapes et des groupes de présentation, ainsi que sur la configuration des propriétés des paramètres d'entrée, reportez-vous à [Création de la boîte de dialogue des paramètres d'entrée dans l'onglet Présentation](#).

## Répondre à une demande d'interaction utilisateur

Les workflows qui nécessitent des interactions utilisateurs au cours de leur exécution suspendent cette exécution jusqu'à ce que l'utilisateur fournisse l'information nécessaire ou jusqu'à ce qu'ils atteignent le délai d'expiration.

Les workflows qui demandent des interactions utilisateurs définissent les utilisateurs qui peuvent fournir les informations requises et leur adressent les demandes d'interaction.

#### Conditions préalables

Vérifiez qu'au moins un workflow présente l'état En attente d'interaction utilisateur.


#### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Exécuter**.
- 2 Cliquez sur la vue **Orchestrator** dans le client Orchestrator.

### 3 Cliquez sur l'onglet **En attente d'entrée**.

L'onglet **En attente d'entrée** répertorie les workflows en attente d'entrées de votre part ou de la part des membres de votre groupe d'utilisateurs qui disposent d'une autorisation.

### 4 Double-cliquez sur un workflow attendant une entrée.

Le jeton du workflow en attente d'entrée apparaît dans la liste hiérarchique **Workflows** avec le symbole suivant : .

### 5 Cliquez avec le bouton droit sur le jeton du workflow et sélectionnez **Répondre**.

### 6 Suivez les instructions de la boîte de dialogue d'entrée de paramètres afin de fournir les informations demandées par le workflow.

Vous avez fourni des informations à un workflow qui attendait une entrée utilisateur pendant son exécution.

## Appeler des workflows dans des workflows

Les workflows peuvent appeler d'autres workflows au cours de leur exécution. Un workflow peut enclencher un autre workflow soit parce qu'il a besoin de son résultat en tant que paramètre d'entrée pour sa propre exécution, soit pour le démarrer et le laisser s'exécuter en autonomie. Les workflows peuvent également enclencher d'autres workflows à une heure ultérieure définie, ou démarrer plusieurs workflows simultanément.

#### ■ [Éléments du workflow appelant des workflows](#)

Il existe quatre moyens d'appeler d'autres workflows depuis un workflow. Chaque moyen d'appeler un ou des workflows est représenté par un élément différent du schéma de workflow.

#### ■ [Appeler un workflow de manière synchrone](#)

L'appel d'un workflow de manière synchrone permet d'exécuter le workflow appelé dans l'exécution du workflow à l'origine de l'appel. Le workflow à l'origine de l'appel peut utiliser les paramètres de sortie du workflow appelé en tant que paramètres d'entrée lorsqu'il exécutera ses éléments de schéma suivants.

#### ■ [Appeler un workflow de manière asynchrone](#)

L'appel d'un workflow de manière asynchrone permet d'exécuter le workflow appelé indépendamment du workflow à l'origine de l'appel. Le workflow à l'origine de l'appel continue son exécution sans attendre que le workflow appelé soit terminé.

#### ■ [Planifier un workflow](#)

Vous pouvez appeler un workflow à partir d'un autre workflow et le planifier qu'il s'exécute à une date et une heure ultérieures.

#### ■ [Conditions préalables pour appeler un workflow distant au sein d'un autre workflow](#)

Si le workflow que vous développez appelle un autre workflow hébergé sur un serveur Orchestrator distant, certaines conditions préalables doivent être satisfaites pour que le workflow distant s'exécute sans problème.

## ■ Appeler plusieurs workflows simultanément

L'appel de plusieurs workflows simultanément permet d'exécuter les workflows appelés de manière synchrone dans l'exécution du workflow à l'origine de l'appel. Le workflow à l'origine de l'appel attend la fin de tous les workflows appelés avant de reprendre. Le workflow à l'origine de l'appel pourra utiliser les résultats des workflows appelés en tant que paramètres d'entrée lorsqu'il exécutera ses éléments de schéma suivants.

## Éléments du workflow appelant des workflows

Il existe quatre moyens d'appeler d'autres workflows depuis un workflow. Chaque moyen d'appeler un ou des workflows est représenté par un élément différent du schéma de workflow.

### Workflows synchrones

Un workflow peut démarrer un autre workflow de façon synchrone. Le workflow appelé s'exécute au cœur même de l'exécution du workflow appelant et dans le même espace de mémoire que ce dernier. Le workflow appelant démarre un autre workflow, puis attend que l'exécution de celui-ci soit terminée avant de lancer l'exécution de l'élément suivant de son schéma. En général, vous appellerez un workflow de façon synchrone car le workflow appelant requiert la sortie du workflow appelé comme paramètre d'entrée de l'élément du schéma qui suit. Par exemple, un workflow peut appeler le workflow Démarrer la machine virtuelle et attendre que ce workflow démarre la machine virtuelle, puis obtenir l'adresse IP de celle-ci pour la transmettre à un autre élément ou à un utilisateur par e-mail.

### Workflows asynchrones

Un workflow peut démarrer un autre workflow de façon asynchrone. Le workflow appelant démarre un autre workflow, mais poursuit immédiatement l'exécution de l'élément suivant de son schéma, sans attendre le résultat du workflow appelé. Les workflows appelés s'exécutent avec des paramètres d'entrée définis par le workflow appelant, mais le cycle de vie du workflow appelé est indépendant de celui du workflow appelant. Les workflows asynchrones vous permettent de créer des chaînes de workflows qui transmettent les paramètres d'entrée d'un workflow au workflow suivant. Par exemple, un workflow peut créer divers objets au cours de son exécution. Le workflow peut alors démarrer des workflows asynchrones qui utilisent ces objets comme paramètres d'entrée dans leur propre exécution. Lorsque le workflow d'origine a démarré l'ensemble des workflow requis et exécuté les éléments qu'il lui reste, il s'achève. Toutefois, les workflows asynchrones qu'il a démarrés poursuivent leur exécution indépendamment du workflow qui les a démarrés.

Pour que le workflow appelant attende le résultat du workflow appelé, vous pouvez soit utiliser un workflow imbriqué, soit créer une tâche inscriptible qui récupérera l'état de jeton du workflow appelé, puis récupérera le résultat du workflow à son achèvement.

**Workflows planifiés**

Un workflow peut appeler un autre workflow tout en différant le démarrage de cet autre workflow à une date et une heure ultérieures. Le workflow appelant poursuit alors son exécution jusqu'à la fin. Appeler un workflow planifié crée une tâche qui démarrera ce workflow à la date et l'heure données. Lorsque le workflow appelant s'est exécuté, vous pouvez voir le workflow planifié dans les vues **Planificateur** et **Mon Orchestrator** du client Orchestrator.

Les workflows planifiés ne s'exécutent qu'une fois. Vous pouvez planifier un workflow pour qu'il s'exécute de façon récurrente en appelant la méthode `Workflow.scheduleRecurrently` dans un élément Tâche inscriptible d'un workflow synchrone.

**Workflows intégrés**

Un workflow peut démarrer simultanément plusieurs autres workflows en imbriquant plusieurs workflows dans un seul élément du schéma. Tous les workflows répertoriés dans l'élément de workflow imbriqué démarreront simultanément lorsque le workflow appelant arrivera à l'élément de workflow imbriqué de son schéma. Il est important de noter que chaque workflow imbriqué démarre dans un espace mémoire différent de celui du workflow appelant. Le workflow appelant attend que l'exécution de tous les workflows intégrés soit terminée avant de lancer l'exécution de l'élément suivant de son schéma. Le workflow appelant pourra utiliser les résultats des workflows intégrés comme paramètres d'entrée lorsqu'il exécutera les éléments qu'il lui reste.

**Propagation des changements dans le workflow à d'autres workflows**

Si vous appelez un workflow depuis un autre workflow, Orchestrator importe les paramètres d'entrée du workflow enfant dans le workflow parent au moment où vous ajoutez l'élément de workflow au schéma.

Si vous modifiez le workflow enfant après l'avoir ajouté à l'autre workflow, le workflow parent fait appel à la nouvelle version du workflow enfant, mais n'importe aucun nouveau paramètre d'entrée. Pour empêcher que les changements opérés dans des workflow aient une incidence sur le comportement des autres workflows qui les appellent, Orchestrator ne propage pas automatiquement les nouveaux paramètres d'entrée aux workflows appelants.

Pour propager les paramètres d'un workflow à d'autres workflows qui l'appellent, vous devez rechercher ces derniers et synchroniser les workflows manuellement.

**Conditions préalables**

Vérifiez que vous avez bien un workflow appelé par un ou plusieurs autres workflows.

**Procédure**

- 1 Modifier et enregistrer un workflow qu'appellent d'autres workflows.
- 2 Fermez l'éditeur de workflow.

- 3 Localisez le workflow que vous avez modifié dans la liste hiérarchique de la vue **Workflows** du client Orchestrator.
- 4 Cliquez avec le bouton droit sur le workflow et sélectionnez **Références > Rechercher les éléments qui emploient cet élément**.  
Une liste des workflows qui appellent ce workflow apparaît.
- 5 Double-cliquez sur un workflow de la liste pour le mettre en évidence dans la vue **Workflows** du client Orchestrator.
- 6 Cliquez avec le bouton droit sur le workflow et sélectionnez **Modifier**.  
L'éditeur de workflow s'ouvre.
- 7 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.
- 8 Cliquez avec le bouton droit sur l'élément de workflow correspondant au workflow provenant du schéma de workflow et sélectionnez **Synchroniser > Paramètres de synchronisation**.
- 9 Dans la boîte de dialogue de confirmation, sélectionnez **Continuer**.
- 10 Enregistrez et fermez l'éditeur de workflow.
- 11 Répétez [Étape 5](#) à [Étape 10](#) pour tous les workflows utilisant le workflow modifié.

Vous avez donc propagé un workflow modifié aux autres workflows qui l'appellent.

## Répercuter les paramètres d'entrée et la présentation d'un workflow enfant dans un workflow parent

Si vous développez un workflow qui appelle d'autres workflows, vous pouvez répercuter les paramètres d'entrée et la présentation des workflows enfants dans le workflow parent.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Exécuter**.
- 2 Cliquez avec le bouton droit sur le workflow que vous souhaitez modifier, puis sélectionnez **Modifier**.  
L'éditeur de workflow s'ouvre.
- 3 Sélectionnez l'onglet **Schéma**.
- 4 Cliquez avec le bouton droit sur l'élément du workflow enfant dont vous souhaitez répercuter les paramètres d'entrée et la présentation dans le workflow parent, puis sélectionnez **Synchroniser > Synchroniser la présentation**.
- 5 Cliquez sur **OK** dans la boîte de dialogue de confirmation.
- 6 (Facultatif) Répétez [Étape 4](#) et [Étape 5](#) pour l'ensemble des workflows enfants dont vous souhaitez répercuter les paramètres d'entrée et la présentation dans le workflow parent.

Les paramètres d'entrée des workflows enfants sont ajoutés aux paramètres d'entrée du workflow parent. La présentation du workflow parent est étendue pour inclure les présentations des workflows enfants.

## Appeler un workflow de manière synchrone

L'appel d'un workflow de manière synchrone permet d'exécuter le workflow appelé dans l'exécution du workflow à l'origine de l'appel. Le workflow à l'origine de l'appel peut utiliser les paramètres de sortie du workflow appelé en tant que paramètres d'entrée lorsqu'il exécutera ses éléments de schéma suivants.

Vous pouvez appeler les workflows de manière synchrone à partir d'un autre workflow en utilisant l'élément **Workflow**.

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Ajoutez des éléments au schéma de workflow.

### Procédure

- 1 Faites glisser un élément **Workflow** depuis le menu **Générique** vers l'emplacement approprié du schéma de workflow.

La boîte de dialogue **Sélectionner un workflow** s'affiche.

- 2 Recherchez et sélectionnez le workflow que vous souhaitez, puis cliquez sur **OK**.

Si la recherche renvoie un résultat partiel, affinez vos critères de recherche ou augmentez le nombre de résultats dans le menu **Outils > Préférences utilisateur** du client.

- 3 Cliquez sur l'élément **Workflow** pour afficher ses onglets de propriétés dans la partie inférieure de l'onglet **Schéma**.
- 4 Cliquez sur l'icône **Modifier** (✎) de l'élément **Workflow** dans le schéma de workflow.
- 5 Reliez les paramètres d'entrée nécessaires au workflow dans l'onglet **Entrée** de l'élément de schéma du workflow.
- 6 Reliez les paramètres de sortie nécessaires au workflow dans l'onglet **Sortie** de l'élément de schéma du workflow.
- 7 Définissez le comportement en cas d'exception du workflow dans l'onglet **Exceptions**.
- 8 Cliquez sur **Fermer**.
- 9 Cliquez sur **Enregistrer** en bas de l'éditeur de workflow.

Vous avez appelé un workflow de manière synchrone à partir d'un autre workflow. Lorsque le workflow atteindra le workflow synchrone au cours de son exécution, le workflow synchrone démarrera et le workflow initial attendra la fin du workflow synchrone pour reprendre son exécution.

### Étape suivante

Vous pouvez appeler un workflow de manière asynchrone à partir d'un workflow.

## Appeler un workflow de manière asynchrone


L'appel d'un workflow de manière asynchrone permet d'exécuter le workflow appelé indépendamment du workflow à l'origine de l'appel. Le workflow à l'origine de l'appel continue son exécution sans attendre que le workflow appelé soit terminé.

Vous pouvez appeler les workflows de manière asynchrone à partir d'un autre workflow en utilisant l'élément **Workflow asynchrone**.

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Ajoutez des éléments au schéma de workflow.

### Procédure

- 1 Faites glisser un élément **Workflow asynchrone** depuis le menu **Générique** vers l'emplacement approprié du schéma de workflow.  
La boîte de dialogue **Sélectionner un workflow** s'affiche.
- 2 Recherchez et sélectionnez le workflow souhaité dans la liste, puis cliquez sur **OK**.
- 3 Cliquez sur l'icône **Modifier** () de l'élément **Workflow asynchrone** dans le schéma de workflow.
- 4 Reliez les paramètres d'entrée nécessaires au workflow dans l'onglet **Entrée** de l'élément de workflow asynchrone.
- 5 Reliez le paramètre de sortie nécessaire dans l'onglet **Sortie** de l'élément de workflow asynchrone.  
Le paramètre de sortie peut être relié soit au workflow appelé, soit au résultat de ce workflow.
  - Relier au workflow appelé pour renvoyer ce workflow en tant que paramètre de sortie
  - Reliez au jeton du workflow appelé pour renvoyer le résultat de l'exécution du workflow appelé.
- 6 Définissez le comportement en cas d'exception de l'élément de workflow asynchrone dans l'onglet **Exceptions**.
- 7 Cliquez sur **Fermer**.
- 8 Cliquez sur **Enregistrer** au bas de l'éditeur de workflow.

Vous avez appelé un workflow de manière asynchrone à partir d'un autre workflow. Lorsque le workflow atteindra le workflow asynchrone au cours de son exécution, le workflow asynchrone démarrera et le workflow initial poursuivra son exécution sans attendre la fin du workflow asynchrone.

### Étape suivante

Vous pouvez planifier un workflow pour qu'il s'exécute à une date et à une heure ultérieures.

## Planifier un workflow


Vous pouvez appeler un workflow à partir d'un autre workflow et le planifier qu'il s'exécute à une date et une heure ultérieures.

Les workflows se planifient dans un autre workflow à l'aide de l'élément **Workflow planifié**.

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Ajoutez des éléments au schéma de workflow.

### Procédure

- 1 Faites glisser un élément **Workflow planifié** depuis le menu **Générique** et déposez-le sur la position adéquate dans le schéma de workflow.
- 2 Recherchez le workflow à appeler en saisissant une partie de son nom dans la zone de texte.
- 3 Sélectionnez le workflow dans la liste et cliquez sur **OK**.
- 4 Cliquez sur l'icône **Modifier** () de l'élément **Workflow planifié** du schéma de workflow.
- 5 Cliquez sur l'onglet de la propriété **IN**.  
Un paramètre nommé `workFlowScheduleDate` apparaît dans la liste des propriétés à définir conjointement aux paramètres d'entrée du workflow appelant.
- 6 Cliquez sur **Non défini** pour le paramètre `workFlowScheduleDate` afin de le définir.
- 7 Cliquez sur **Créer un paramètre/attribut dans le workflow** pour créer le paramètre et définir sa valeur.
- 8 Cliquez sur **Non définie** pour la **Valeur** afin de définir la valeur du paramètre.
- 9 Utilisez le calendrier qui s'affiche pour définir la date et l'heure de démarrage du workflow planifié, puis cliquez sur **OK**.
- 10 Créez une liaison entre les paramètres d'entrée restants et le workflow planifié dans l'onglet **IN** de l'élément **Workflow planifié**.
- 11 Créez une liaison entre les paramètres de sortie obligatoires et l'objet `Task` dans l'onglet **OUT** de l'élément **Workflow planifié**.
- 12 Définissez un comportement en cas d'exception pour l'élément **Workflow planifié** dans l'onglet **Exceptions**.
- 13 Cliquez sur **Fermer**.
- 14 Cliquez sur **Enregistrer** au bas de l'éditeur de workflow.

Vous avez donc planifié un workflow pour qu'il se lance à une heure et une date données depuis un autre workflow.

### Étape suivante

Vous pouvez appeler simultanément de multiples workflows depuis un workflow.



## Conditions préalables pour appeler un workflow distant au sein d'un autre workflow

Si le workflow que vous développez appelle un autre workflow hébergé sur un serveur Orchestrator distant, certaines conditions préalables doivent être satisfaites pour que le workflow distant s'exécute sans problème.

- Tous les paramètres d'entrée du workflow distant doivent pouvoir s'appliquer sur le serveur Orchestrator distant.
- Tous les paramètres de sortie du workflow distant doivent pouvoir s'appliquer sur le serveur Orchestrator local.

Pour garantir que les paramètres du workflow distant sont applicables, les objets d'inventaire utilisés par le workflow doivent être disponibles aussi bien sur le serveur Orchestrator local que sur le serveur Orchestrator distant. Si un workflow distant utilise les objets d'un plug-in, ce plug-in doit être disponible sur les deux serveurs Orchestrator. Les inventaires du plug-in distant et du plug-in local doivent être identiques. Si le workflow distant utilise des objets système tels que des workflows et des actions dans Orchestrator, les mêmes workflows et actions doivent exister dans les inventaires des serveurs Orchestrator local et distant.

Par exemple, imaginons que vous insériez le workflow Renommer la machine virtuelle dans un élément Workflow intégré du workflow de test que vous développez. Vous souhaitez exécuter le workflow Renommer la machine virtuelle dans un serveur Orchestrator distant. Lorsque vous exécutez le workflow de test, le workflow Renommer la machine virtuelle est appelé pendant l'exécution du workflow de test. Vous indiquez la machine virtuelle à renommer dans l'inventaire du serveur Orchestrator local. Étant donné que le workflow Renommer la machine virtuelle s'exécute sur le serveur Orchestrator distant, cette machine virtuelle doit être disponible dans l'inventaire de ce serveur. Si ce n'est pas le cas, le workflow Renommer la machine virtuelle ne peut pas appliquer son paramètre d'entrée vm. Ainsi, le plug-in vCenter Server des serveurs Orchestrator local et distant doit être connecté à la même instance vCenter Server.

## Appeler plusieurs workflows simultanément

L'appel de plusieurs workflows simultanément permet d'exécuter les workflows appelés de manière synchrone dans l'exécution du workflow à l'origine de l'appel. Le workflow à l'origine de l'appel attend la fin de tous les workflows appelés avant de reprendre. Le workflow à l'origine de l'appel pourra utiliser les résultats des workflows appelés en tant que paramètres d'entrée lorsqu'il exécutera ses éléments de schéma suivants.

Vous pouvez appeler plusieurs workflows simultanément à partir d'un autre workflow en utilisant l'élément **Workflows intégrés**. Vous pouvez utiliser des workflows intégrés pour exécuter des workflows avec des informations d'identification différentes de celles de l'utilisateur du workflow à l'origine de l'appel.

### Conditions préalables

- Ouvrez le workflow à modifier dans l'éditeur de workflow.
- Ajoutez des éléments au schéma de workflow.

## Procédure

- 1 Faites glisser un élément **Workflows intégrés** depuis le menu **Action et workflow** vers l'emplacement approprié du schéma de workflow.

La boîte de dialogue **Sélectionner un workflow** s'affiche.

- 2 Recherchez et sélectionnez le workflow à démarrer, puis cliquez sur **OK**.
- 3 Cliquez sur l'icône **Modifier** (✎) de l'élément **Workflows intégrés** du schéma de workflow.
- 4 Cliquez sur l'onglet **Workflows**.

Le workflow que vous avez sélectionné en [Étape 2](#) s'affiche dans l'onglet.

- 5 Définissez les liaisons d'entrée et de sortie de ce workflow dans les onglets **Entrée** et **Sortie** situés dans le panneau droit de l'onglet des propriétés de l'élément de schéma **Workflows**.
- 6 Cliquez sur l'onglet **Informations de connexion** dans le panneau droit de l'onglet de propriétés de l'élément de schéma **Workflows**.

L'onglet **Informations de connexion** vous permet d'accéder aux workflows stockés dans un autre serveur que le serveur local avec les informations d'identification appropriées.

- 7 Pour accéder aux workflows d'un serveur distant, sélectionnez **Distant**, puis cliquez sur **Non défini** afin d'indiquer le nom d'hôte ou l'adresse IP du serveur distant.

---

**Note** Vous pouvez utiliser le plug-in à nœuds multiples vRealize Orchestrator pour appeler des workflows sur un serveur distant.

---

- 8 Définissez les informations d'identification avec lesquelles vous souhaitez accéder au serveur distant.
  - Sélectionnez **Hériter** pour utiliser les mêmes informations d'identification que l'utilisateur exécutant le workflow à l'origine de l'appel.
  - Sélectionnez **Dynamiques**, puis cliquez sur **Non définies** pour sélectionner un ensemble d'informations d'identification dynamiques définies à un autre moment du workflow par un paramètre du type `credentials`.
  - Sélectionnez **Statiques**, puis cliquez sur **Non définies** pour entrer directement les informations d'identification.
- 9 Cliquez sur le bouton **Ajouter un workflow** dans l'onglet **Workflows** pour sélectionner davantage de workflows à ajouter à l'élément de workflows intégrés.
- 10 Répétez les étapes [Étape 2](#) à [Étape 8](#) pour définir les paramètres de chacun des workflows que vous ajoutez.
- 11 Cliquez sur l'élément de workflows intégrés dans le schéma de workflow.

Le nombre de workflows intégrés à l'élément apparaît sous forme de chiffre sur l'élément de workflows intégrés.

Vous avez appelé plusieurs workflows simultanément à partir d'un workflow.

## Étape suivante

Vous pouvez définir des workflows au long cours.

## Exécuter un workflow sur une sélection d'objets

Vous pouvez automatiser des tâches répétitives en exécutant un workflow sur une sélection d'objets. Par exemple, vous pouvez créer un workflow qui capture un snapshot de l'ensemble des machines virtuelles d'un dossier ou un workflow qui met hors tension l'ensemble des machines virtuelles sur un hôte précis.

Vous pouvez utiliser l'une des méthodes suivantes pour exécuter un workflow sur une sélection d'objets

- Exécuter le workflow **Bibliothèque > vCenter > Lot > Exécuter un workflow sur une sélection d'objets**.
- Créez une bibliothèque qui appelle les workflows **Bibliothèque > Orchestrator > Démarrer les workflows en série** ou **Démarrer les workflows en parallèle**.
- Créez un workflow qui obtient un tableau d'objets et s'exécute sur chacun de ces objets dans une boucle d'éléments de workflow.
- Exécutez un workflow à partir de JavaScript en appelant la méthode `workflow.execute()` dans une boucle `For` d'un élément en script d'un workflow.

Le choix de la méthode à utiliser pour exécuter un workflow sur une sélection d'objets dépend de ce workflow et peut en affecter les performances. Par exemple, le workflow Exécuter un workflow sur une sélection d'objets constitue le moyen le plus simple d'exécuter un workflow sur plusieurs objets. Aucun développement de workflow n'est nécessaire mais seuls les workflows présentant un paramètre d'entrée unique peuvent être exécutés.

La création d'un workflow qui appelle les workflows Démarrer les workflows en série ou Démarrer les workflows en parallèle vous permet d'exécuter plusieurs workflows d'objets qui comportent plusieurs paramètres d'entrée. Le workflow d'appel doit créer un tableau des propriétés pour transmettre les paramètres d'entrée au workflow Démarrer les workflows en série ou Démarrer les workflows en parallèle. Ces workflows peuvent uniquement être utilisés dans d'autres workflows. Ne les exécutez pas directement.

L'exécution d'un workflow dans une boucle `For` d'un élément en script est plus rapide que l'exécution d'un workflow dans une boucle d'éléments, mais cette première approche est moins flexible et limite les possibilités de réutilisation. Plus important encore, l'exécution d'un workflow dans une boucle en script fait disparaître les contrôles exécutés par Orchestrator au démarrage de chaque élément de l'exécution d'un workflow. Ainsi, si le serveur Orchestrator s'arrête alors que la boucle en script est en cours d'exécution, lorsqu'il redémarrera le workflow reprendra au début de l'élément en script et répétera l'ensemble de la boucle. Si le serveur Orchestrator s'arrête pendant l'exécution d'un workflow avec une boucle d'éléments, le workflow reprendra au niveau de l'élément de la boucle qui était en train de s'exécuter lors de l'interruption.

Pour plus d'informations sur les workflows par lots reportez-vous à *Utilisation des plug-ins VMware vRealize Orchestrator*.

La création d'un workflow qui s'exécute sur un tableau d'objets dans une boucle d'éléments de workflow est illustrée dans [Développer un workflow complexe](#).

L'exécution d'un workflow dans une boucle For en script est illustrée dans [Exemples de scripts de workflow](#).

## Implémenter les workflows Démarrer les workflows en série et Démarrer les workflows en parallèle

Vous pouvez utiliser les workflows Démarrer les workflows en série et Démarrer les workflows en parallèle pour exécuter un workflow sur une sélection d'objets.

Vous ne pouvez pas exécuter directement les workflows Démarrer les workflows en série et Démarrer les workflows en parallèle. Vous devez les inclure dans un autre workflow que vous créez. Si vous souhaitez utiliser les workflows Démarrer les workflows en série et Démarrer les workflows en parallèle pour exécuter un workflow sur une sélection d'objets, vous devez d'abord obtenir ces objets. Vous transmettez ces objets et tous les autres paramètres d'entrée demandés au workflow sous forme de tableau de propriétés. Les workflows Démarrer les workflows en série et Démarrer les workflows en parallèle affichent les résultats de l'exécution du workflow sur la sélection d'objets sous forme de tableau d'objets WorkflowToken.

Vous implémentez les workflows Démarrer les workflows en série et Démarrer les workflows en parallèle de la même façon. Le workflow Démarrer les workflows en série exécute le workflow sur chaque objet l'un après l'autre. Le workflow Démarrer les workflows en parallèle exécute ce workflow simultanément sur tous les objets.

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Dans le schéma de workflow, ajoutez un élément de tâche ou une action inscriptible afin d'obtenir la liste des objets sur lesquels le workflow doit s'exécuter.

Par exemple, pour exécuter un workflow sur l'ensemble des machines virtuelles d'un dossier de machines virtuelles, vous pouvez ajouter l'action `getAllVirtualMachinesByFolder`.

- 2 Reliez l'élément ou l'action en script, puis reliez l'entrée et la sortie de cet élément ou de cette action aux entrées ou attributs du workflow.

Par exemple, vous pouvez relier l'entrée `vmFolder` de l'action `getAllVirtualMachinesByFolder` à un paramètre d'entrée et la sortie `actionResult` à un attribut dans le workflow de l'appel.

- 3 Ajoutez un élément de tâche inscriptible pour répartir la liste d'objets dans un tableau de propriétés.

Par exemple, si les objets sur lesquels le workflow doit s'exécuter apparaissent sous forme de tableau de machines virtuelles `allVMs` renvoyé par la sortie `actionResult` de l'action `getAllVirtualMachinesByFolder`, vous pouvez écrire le script suivant pour répartir les objets dans un tableau de propriétés.

```
propsArray = new Array();

for each (var vm in allVMs) {
    var prop = new Properties();
    prop.put("vm", vm);
    propsArray.push(prop);
}
```

- 4 Reliez les entrées et les sorties de l'élément de tâche inscriptible aux attributs du workflow.

Dans l'exemple d'élément de tâche inscriptible de [Étape 3](#), vous reliez l'entrée au tableau `allVMs` de machines virtuelles et vous créez l'attribut de sortie `propsArray` sous forme de tableau d'objets `Properties`.

- 5 Ajoutez un élément de workflow au schéma de workflow.

- 6 Sélectionnez soit le workflow Démarrer les workflows en série, soit le workflow Démarrer les workflows en parallèle, puis reliez l'élément de workflow aux autres éléments.

- 7 Reliez l'entrée `wf` du workflow Démarrer les workflows en série ou Démarrer les workflows en parallèle au workflow à exécuter sur les objets.

Par exemple, pour supprimer les snapshots de l'ensemble des machines virtuelles renvoyées par l'action `getAllVirtualMachinesByFolder`, sélectionnez le workflow Supprimer tous les snapshots.

- 8 Reliez l'entrée `parameters` du workflow Démarrer les workflows en série ou Démarrer les workflows en parallèle au tableau d'objets `Properties` qui contient les objets sur lesquels le workflow sera exécuté.

Par exemple, reliez l'entrée `parameters` à l'attribut `propsArray` défini dans [Étape 4](#).

- 9 (Facultatif) Reliez la sortie `workflowTokens` du workflow Démarrer les workflows en série ou Démarrer les workflows en parallèle à un attribut du workflow.

- 10 (Facultatif) Continuez à ajouter d'autres éléments utilisant les résultats de l'exécution du workflow Démarrer les workflows en série ou Démarrer les workflows en parallèle.

Vous avez créé un workflow utilisant le workflow Démarrer les workflows en série ou Démarrer les workflows en parallèle pour exécuter un workflow sur une sélection d'objets.

## Développement de workflows au long cours

Un workflow en attente consomme des ressources système car il ne cesse de sonder l'objet auquel il demande une réponse. Si vous savez qu'un workflow attendra probablement pendant une longue période avant de recevoir une réponse, vous pouvez lui ajouter des éléments de workflow au long cours.

Tous les workflows en cours d'exécution consomment un thread système. Lorsqu'un workflow atteint un élément de workflow au long cours, cet élément met le workflow dans un état passif. L'élément de workflow au long cours envoie ensuite les informations du workflow à un thread qui sonde le système pour découvrir tous les éléments de workflow au long cours du serveur. Au lieu que chaque élément de workflow au long cours ne tente sans cesse de récupérer les informations dans le système, ils demeurent passifs pendant une période donnée tandis que le thread de workflows au long cours sonde le système pour eux.

Vous pouvez définir la durée de l'attente de l'une des manières suivantes :

- Définir un temporisateur encapsulé dans un objet `Date` qui interrompt le workflow jusqu'à une date et une heure données. Vous pouvez implémenter les éléments de workflow au long cours basés sur un temporisateur en intégrant un élément **Minuterie d'attente** au schéma.
- Définir un événement déclencheur encapsulé dans un objet `Trigger` qui redémarre le workflow après qu'il se soit produit. Vous pouvez implémenter les éléments de workflow au long cours basés sur un déclencheur en intégrant un élément **Événement d'attente** ou **Interaction utilisateur** au schéma.

## Définir une date et une heure relatives pour les workflows s'appuyant sur un temporisateur

Vous pouvez définir l'attribut `timer.date` d'un élément **Minuterie d'attente** sur une date et une heure relatives en lui créant une liaison avec un objet `Date`. Vous définissez l'objet `Date` dans une fonction en script.


Lorsque l'heure et la date données adviennent, le workflow au long cours qui repose sur un temporisateur se réactive et poursuit son cours. Par exemple, vous pouvez régler le workflow pour qu'il se réactive à midi le 12 février. Vous pouvez également créer un élément de workflow qui calcule et génère un objet `Date` relatif selon une fonction que vous définissez. Par exemple, vous pouvez créer un objet `Date` relatif qui ajoute 24 heures à l'heure actuelle.

### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.
- Ajoutez des éléments au schéma de workflow.

### Procédure

- 1 Faites glisser un élément **Tâche inscriptible** du menu **Générique** et déposez-le sur le schéma d'un workflow, à la position qui précède l'élément réclamant l'objet `Date` relative pour son attribut `timeout.date`.
- 2 Cliquez sur l'icône **Modifier** (✎) de l'élément **Tâche inscriptible** du schéma de workflow.
- 3 Renseignez le nom et la description de l'élément de workflow en script dans l'onglet de propriétés **Infos**.

- 4 Cliquez sur l'onglet de propriétés **OUT** et cliquez sur l'icône **Créer une liaison avec le paramètre/attribut du workflow** (.
- 5 Cliquez sur **Créer un paramètre/attribut dans le workflow** pour créer un attribut de workflow.
  - a Nommez l'attribut `timerDate`.
  - b Sélectionnez `Date` dans la liste des types d'attribut.
  - c Sélectionnez **Créer un ATTRIBUT de workflow du même nom**.
  - d Laissez la valeur de l'attribut définie sur **Non définie** car c'est la fonction en script qui renseignera cette valeur.
  - e Cliquez sur **OK**.
- 6 Cliquez sur l'onglet **Script** pour l'élément de workflow en script.
- 7 Définissez une fonction pour calculer et générer un objet `Date` nommé `timerDate` sur la zone de scripts de l'onglet **Script**.

Vous pouvez, par exemple, créer un objet `Date` en mettant en œuvre la fonction JavaScript suivante, dans laquelle le délai d'expiration est un compte à rebours relatif en millisecondes.

```
timerDate = new Date();
System.log( "Current date : '" + timerDate + "'" );
timerDate.setTime( timerDate.getTime() + (86400 * 1000) );
System.log( "Timer will expire at '" + timerDate + "'" );
```

L'exemple de fonction JavaScript qui précède définit un objet `Date` qui obtient la date et l'heure actuelles au moyen de la méthode `getTime` et y ajoute 86 400 000 millisecondes ou 24 heures. L'élément **Tâche inscriptible** génère comme paramètre de sortie cette valeur.

- 8 Cliquez sur **Fermer**.
- 9 Cliquez sur **Enregistrer**.

Vous avez donc créé une fonction qui calcule et génère un objet `Date`. Un élément **Minuterie d'attente** peut recevoir cet objet `Date` en paramètre d'entrée afin de suspendre un workflow au long cours jusqu'à la date encapsulée dans cet objet. Lorsque le workflow arrive sur l'élément **Minuterie d'attente**, il suspend son cours et patiente 24 heures avant de reprendre.

#### Étape suivante

Pour mettre en œuvre un workflow au long cours qui s'appuie sur un temporisateur, vous devez ajouter un élément **Minuterie d'attente** au workflow.

## Créer un workflow au long cours basé sur un temporisateur

Si vous savez qu'un workflow devra attendre une réponse d'une source externe pendant un temps prévisible, vous pouvez l'implémenter en temps que workflow au long cours basé sur un temporisateur. Un workflow au long cours basé sur un temporisateur attend jusqu'à la date et l'heure indiquées avant de reprendre.

Vous pouvez implémenter un workflow au long cours basé sur un temporisateur à l'aide de l'élément **Minuterie d'attente**.

### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.
- Ajoutez des éléments au schéma de workflow.

### Procédure

- 1 Faites glisser un élément **Minuterie d'attente** depuis le menu **Générique** vers la position du schéma de workflow à laquelle l'exécution doit s'interrompre.

Si vous implémentez une tâche inscriptible pour calculer la date et l'heure, cet élément doit précéder l'élément **Minuterie d'attente**.

- 2 Cliquez sur l'icône **Modifier** (✎) de l'élément **Minuterie d'attente** dans le schéma de workflow.
- 3 Entrez une description de la raison de l'implémentation du temporisateur dans l'onglet de propriétés **Infos**.
- 4 Cliquez sur l'onglet de propriétés **Attributs**.

Le paramètre `timer.date` s'affiche dans la liste des attributs.

- 5 Cliquez sur le bouton **Non défini** du paramètre `timer.date` pour relier le paramètre à un objet Date approprié.

La boîte de dialogue de sélection de la **Minuterie d'attente** s'ouvre pour proposer une liste des liaisons possibles.

- Sélectionnez un objet Date prédéfini dans la liste proposée ; par exemple un objet défini par un élément **Tâche inscriptible** ailleurs dans le workflow.
- Vous pouvez également créer un objet Date qui définit la date et l'heure spécifiques que le workflow doit attendre.

- 6 (Facultatif) Créez un objet Date qui définit la date et l'heure spécifiques que le workflow doit attendre.

- a Cliquez sur **Créer un paramètre/attribut dans le workflow** dans la boîte de dialogue **Minuterie d'attente**.

La boîte de dialogue **Informations relatives aux paramètres** s'ouvre.

- b Nommez le paramètre de façon appropriée.
- c Laissez le type sur Date.
- d Cliquez sur **Créer un ATTRIBUT de workflow avec le même nom**.
- e Cliquez sur le bouton **Non définie** de la propriété **Valeur** pour définir la valeur du paramètre.

Un calendrier s'affiche.



f Utilisez le calendrier pour définir une date et une heure de redémarrage du workflow.

g Cliquez sur **OK**.

7 Cliquez sur **Fermer**.

8 Cliquez sur **Enregistrer** en bas de l'éditeur de workflow.

Vous avez défini un temporisateur qui suspend un workflow au long cours basé sur un temporisateur jusqu'à une date et une heure déterminées.

### Étape suivante

Vous pouvez créer un workflow au long cours qui attend un événement déclencheur pour poursuivre.

## Créer un objet déclencheur

Les objets déclencheurs surveillent les déclencheurs d'événements définis par les plug-ins. Par exemple, le plug-in vCenter Server définit ces objets en tant qu'objets Task. À la fin de la tâche, le déclencheur envoie un message à l'élément de workflow au long cours basé sur un déclencheur en attente afin de redémarrer le workflow.

L'événement retardant qui suspend le workflow au long cours basé sur un déclencheur doit renvoyer un objet VC:Task. Par exemple, l'action startVM pour démarrer une machine virtuelle renvoie un objet VC:Task afin que les éléments suivants du workflow puissent surveiller sa progression. Un événement déclencheur d'un workflow au long cours a besoin de cet objet VC:Task en tant que paramètre d'entrée.

Vous créez un objet Trigger dans une fonction JavaScript d'un élément **Tâche inscriptible**. Cet élément **Tâche inscriptible** peut faire partie du workflow au long cours qui attend l'événement déclencheur. Il peut également faire partie d'un autre workflow qui fournit des paramètres d'entrée au workflow au long cours basé sur un déclencheur. La fonction de déclencheur doit implémenter la méthode `createEndOfTaskTrigger()` à partir de l'API Orchestrator.

---

**Important** Vous devez définir un délai d'expiration pour tous les déclencheurs car, sinon, le workflow peut attendre indéfiniment.

---


### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.
- Ajoutez des éléments au schéma de workflow.
- Dans le workflow, déclarez un objet VC:Task en tant qu'attribut ou paramètre d'entrée ; par exemple un objet VC:Task d'un workflow ou d'un élément de workflow qui démarre ou clone une machine virtuelle.

## Procédure

- 1 Faites glisser un élément **Tâche inscriptible** depuis le menu **Générique** vers le schéma d'un workflow.

L'un des éléments qui précède la **Tâche inscriptible** doit générer un objet VC:Task en tant que paramètre de sortie.

- 2 Cliquez sur l'icône **Modifier** () de l'élément **Tâche inscriptible** dans le schéma de workflow.
- 3 Entrez un nom et une description pour le déclencheur dans l'onglet de propriétés **Infos**.
- 4 Cliquez sur l'onglet de propriétés **Entrée**.

- 5 Cliquez sur l'icône **Relier à un paramètre/à un attribut du workflow** ()

La boîte de dialogue de sélection des paramètres d'entrée s'ouvre.

- 6 Sélectionnez ou créez un paramètre d'entrée de type **VC:Task**.  
Cet objet VC:Task représente un événement retardant lancé par un autre workflow ou élément.
- 7 (Facultatif) Sélectionnez ou créez un paramètre d'entrée de type Nombre pour définir un délai d'expiration en secondes.
- 8 Cliquez sur l'onglet de propriétés **Sortie**.

- 9 Cliquez sur l'icône **Relier à un paramètre/à un attribut du workflow** ()

La boîte de dialogue de sélection des paramètres de sortie s'ouvre.

- 10 Créez un paramètre de sortie avec les propriétés suivantes.
  - a Créez la propriété Nom avec la valeur trigger.
  - b Créez la propriété Type avec la valeur Trigger.
  - c Cliquez sur **Créer un ATTRIBUT avec le même nom** pour créer l'attribut.
  - d Laissez la valeur sur **Non définie**.

- 11 Définissez un comportement en cas d'exception dans l'onglet de propriétés **Exceptions**.

- 12 Définissez une fonction pour générer un objet Trigger dans l'onglet **Script**.

Par exemple, vous pouvez créer un objet Trigger en implémentant la fonction JavaScript suivante.

```
trigger = task.createEndOfTaskTrigger(timeout);
```

La méthode createEndOfTaskTrigger() renvoie un objet Trigger qui surveille un objet VC:Task nommé task.

- 13 Cliquez sur **Fermer**.
- 14 Cliquez sur **Enregistrer** en bas de l'éditeur de workflow.

Vous avez défini un élément de workflow qui crée un événement déclencheur pour un workflow au long cours. L'élément déclencheur génère un objet `Trigger` en tant que paramètre de sortie auquel un élément **Événement d'attente** peut être relié.

### Étape suivante

Vous devez maintenant relier cet événement déclencheur à un élément **Événement d'attente** dans un workflow au long cours basé sur un déclencheur.

## Créer un workflow au long cours basé sur un déclencheur

Si vous savez qu'un workflow devra attendre une réponse d'une source externe au cours de son exécution mais que vous ne savez pas combien de temps cette attente durera, vous pouvez implémenter ce workflow en temps que workflow au long cours basé sur un déclencheur. Un workflow au long cours basé sur un déclencheur attend qu'un événement déclencheur défini se produise pour reprendre son exécution.

Vous pouvez implémenter un workflow au long cours basé sur un déclencheur à l'aide de l'élément **Événement d'attente**. Lorsque le workflow au long cours basé sur un déclencheur arrive au niveau de l'élément **Événement d'attente**, il interrompt son exécution et attend passivement de recevoir un message du déclencheur. Pendant ce temps d'attente, le workflow passif n'utilise aucun thread mais c'est l'élément de workflow au long cours qui transmet les informations au thread chargé de surveiller tous les workflows au long cours du serveur.

### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.
- Ajoutez des éléments au schéma de workflow.
- Définissez un événement déclencheur encapsulé dans un objet `Trigger`.

### Procédure

- 1 Faites glisser un élément **Événement d'attente** depuis le menu **Générique** vers la position du schéma de workflow à laquelle l'exécution doit s'interrompre.

La tâche inscriptible qui déclare le déclencheur doit précéder immédiatement l'élément **Événement d'attente**.

- 2 Cliquez sur l'icône **Modifier** (✎) de l'élément **Événement d'attente** dans le schéma de workflow.
- 3 Entrez une description de la raison de l'attente dans l'onglet de propriétés **Infos**.
- 4 Cliquez sur l'onglet de propriétés **Attributs**.

Le paramètre `trigger.ref` s'affiche dans la liste des attributs.

- 5 Cliquez sur le lien **Non défini** du paramètre `trigger.ref` pour relier le paramètre à un objet Trigger approprié.

La boîte de dialogue de sélection du **Événement d'attente** s'ouvre pour proposer une liste des paramètres possibles pour la liaison.

- 6 Sélectionnez un objet Trigger prédéfini dans la liste proposée.

Cet objet Trigger représente un événement déclencheur défini par un autre workflow ou élément de workflow.

- 7 Définissez un comportement en cas d'exception dans l'onglet de propriétés **Exceptions**.

- 8 Cliquez sur **Fermer**.

- 9 Cliquez sur **Enregistrer** en bas de l'éditeur de workflow.

Vous avez défini un élément de workflow qui interrompt un workflow au long cours devant attendre un événement déclencheur spécifique pour redémarrer.

#### Étape suivante

Vous pouvez exécuter un workflow.

## Éléments de configuration

Un élément de configuration est une liste d'attributs que vous pouvez utiliser pour configurer les constantes tout au long d'un déploiement d'un serveur Orchestrator.

L'ensemble des workflows, actions et stratégies s'exécutant dans sur un serveur Orchestrator spécifique peuvent utiliser les attributs que vous définissez dans un élément de configuration. La définition des attributs dans les éléments de configuration vous permet de mettre les mêmes valeurs d'attributs à la disposition de l'ensemble des workflows, actions et stratégies s'exécutant sur le serveur Orchestrator.

Si vous créez un module contenant un workflow, une action ou une stratégie qui utilise un attribut provenant d'un élément de configuration, Orchestrator inclut automatiquement cet élément de configuration dans le module. Si vous importez un module contenant un élément de configuration dans un autre serveur Orchestrator, vous pouvez également importer les valeurs des attributs de l'élément de configuration. Par exemple, imaginons que vous ayez créé un workflow qui s'appuie sur des valeurs d'attribut dépendant du serveur Orchestrator sur lequel il s'exécute : la définition de ces attributs dans un élément de configuration vous permet d'exporter ce workflow de manière à ce qu'un autre serveur Orchestrator puisse l'utiliser. Les éléments de configuration vous permettent donc d'échanger plus facilement les workflows, les actions et les stratégies entre les serveurs.

---

**Note** Vous ne pouvez pas importer les valeurs d'un attribut d'élément de configuration à partir d'un élément de configuration provenant d'Orchestrator 5.1 ou d'une version antérieure.

---

## Créer un élément de configuration

Les éléments de configuration vous permettent de définir des attributs communs pour tout un serveur Orchestrator. Tous les éléments s'exécutant sur le serveur peuvent appeler les attributs que vous avez

définis dans un élément de configuration. La création d'éléments de configuration vous permet de définir les attributs communs une seule fois sur le serveur plutôt que pour chaque élément.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Configurations**.
- 3 Cliquez avec le bouton droit sur un dossier dans la liste hiérarchique, puis sélectionnez **Nouveau dossier** pour créer un nouveau dossier.
- 4 Entrez un nom pour le dossier, puis cliquez sur **OK**.
- 5 Cliquez avec le bouton droit sur le dossier que vous avez créé, puis sélectionnez **Nouvel élément**.
- 6 Entrez un nom pour l'élément de configuration, puis cliquez sur **OK**.  
L'éditeur des éléments de configuration s'ouvre.
- 7 Insérez le numéro de version en cliquant sur les chiffres de version dans l'onglet **Général** et entrez un commentaire sur la version.
- 8 Entrez une description de l'élément de configuration dans la zone de texte **Description** de l'onglet **Général**.
- 9 Cliquez sur l'onglet **Attributs**.
- 10 Cliquez sur l'icône **Ajouter un attribut** (A+) pour créer un nouvel attribut.
- 11 Cliquez sur les valeurs de l'attribut sous **Nom**, **Type**, **Valeur** et **Description** pour définir le nom, le type, la valeur et la description de l'attribut.
- 12 Cliquez sur l'onglet **Autorisations**.
- 13 Cliquez sur l'icône **Ajouter des droits d'accès** (🔑) pour accorder une autorisation d'accès à cet élément de configuration à un groupe d'utilisateurs.
- 14 Recherchez un groupe d'utilisateurs dans la zone de texte **Filtrer** et sélectionnez le groupe d'utilisateurs concerné dans la liste.
- 15 Cochez les cases appropriées pour accéder aux droits de ce groupe d'utilisateurs.

Vous pouvez définir les autorisations suivantes pour l'élément de configuration.

Autorisation	Description
<b>Afficher</b>	Les utilisateurs peuvent afficher l'élément de configuration mais ils ne peuvent pas afficher les schémas ou les scripts.
<b>Inspecter</b>	Les utilisateurs peuvent afficher l'élément de configuration avec les schémas et les scripts.
<b>Administrateur</b>	Les utilisateurs peuvent définir des autorisations concernant les éléments de l'élément de configuration et disposent de toutes les autres autorisations.

Autorisation	Description
Exécuter	Les utilisateurs peuvent exécuter les éléments de l'élément de configuration.
Modifier	Les utilisateurs peuvent modifier les éléments de l'élément de configuration.

**16** Cliquez sur **Sélectionner**.

**17** Cliquez sur **Enregistrer et fermer** pour quitter l'éditeur de l'élément de configuration.

Vous avez défini un élément de configuration qui détermine des attributs communs pour tout un serveur Orchestrator.

### Étape suivante

Vous pouvez utiliser cet élément de configuration pour fournir des attributs aux workflows ou aux actions.

## Autorisations de l'utilisateur de workflow

Orchestrator définit les niveaux d'autorisation que vous pouvez appliquer aux groupes pour leur accorder ou leur refuser l'accès aux workflows.

<b>Afficher</b>	L'utilisateur a la possibilité de visualiser les éléments se trouvant dans le workflow, sans en voir les schémas, ni les scripts.
<b>Inspecter</b>	L'utilisateur a la possibilité de visualiser les éléments se trouvant dans le workflow, schémas et scripts compris.
<b>Exécuter</b>	L'utilisateur a la possibilité d'exécuter le workflow.
<b>Modifier</b>	L'utilisateur a la possibilité de modifier le workflow.
<b>Administrateur</b>	L'utilisateur a la possibilité de définir les autorisations sur le workflow et détient tous les autres droits.

L'autorisation **Admin** inclut les autorisations **Afficher**, **Inspecter**, **Modifier** et **Exécuter**. Toutes les autorisations nécessitent l'autorisation **Afficher**.

Si vous ne définissez aucune autorisation sur un workflow, celui-ci héritera des autorisations du dossier qui le contient. Si vous définissez des autorisations sur un workflow, elles remplacent celles du dossier qui le contient, même si ces dernières sont plus restrictives.

## Définir les autorisations des utilisateurs sur un workflow

Vous définissez différents niveaux d'autorisation sur un workflow pour limiter l'accès que les groupes d'utilisateurs peuvent avoir à ce workflow.


Vous pouvez sélectionner les utilisateurs et groupes d'utilisateurs pour qui définir des autorisations dans le serveur LDAP d'Orchestrator.

### Conditions préalables

- Créez un workflow.

- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.
- Ajoutez des éléments au schéma de workflow.

### Procédure

- 1 Cliquez sur l'onglet **Autorisations**.
- 2 Cliquez sur l'icône **Ajouter des droits d'accès** () pour définir les autorisations d'un nouveau groupe d'utilisateurs.

Les résultats de la recherche renferment l'ensemble des groupes d'utilisateurs du serveur LDAP d'Orchestrator qui répondent à la recherche.
- 3 Recherchez un groupe d'utilisateurs.

Pour autoriser un utilisateur de ce groupe d'utilisateurs à visualiser le workflow, inspecter le schéma et les scripts, exécuter et modifier le workflow et changer les autorisations, vous devez cocher la totalité des cases.
- 5 Cliquez sur **Sélectionner**.

Le groupe d'utilisateurs apparaît alors dans la liste des autorisations.
- 6 Cliquez sur **Enregistrer et fermer** pour quitter l'éditeur.

## Validation des workflows

Orchestrator fournit un outil de validation des workflows. Valider un workflow aide à identifier les erreurs qui peuvent s'y trouver et contrôle que les données s'enchaînent correctement d'un élément au suivant.

Lorsque vous validez un workflow, l'outil de validation crée une liste de toutes les erreurs et avertissements éventuels. Cliquer sur une erreur de la liste met en évidence l'élément de workflow qui contient l'erreur.

Si vous lancez l'outil de validation dans l'éditeur de workflow, il proposera une suggestion de correctifs rapides pour les erreurs détectées. Certains correctifs rapides impliquent que vous fournissiez des informations supplémentaires ou des paramètres d'entrée. D'autres correctifs rapides régleront l'erreur pour vous.

La validation des workflows contrôle les liaisons et connexions de données entre les éléments. La validation des workflows ne contrôle pas le traitement des données qu'effectue chaque élément du workflow. Par conséquent, un workflow peut s'exécuter de façon incorrecte et produire des résultats erronés si une fonction au sein d'un élément du schéma est incorrecte.

Par défaut, Orchestrator procède systématiquement à la validation des workflows lorsque vous en exécutez un. Vous pouvez modifier le comportement par défaut de la validation dans le client Orchestrator. Consultez [Tester des workflows au cours du développement](#). Par exemple, il peut arriver qu'au cours du développement des workflows, vous souhaitiez exécuter à des fins de test un workflow que vous savez être non valide.

## Valider un workflow et corriger les erreurs de validation

Vous devez valider un workflow avant de pouvoir l'exécuter. Vous pouvez valider des workflows soit dans le client Orchestrator, soit dans l'éditeur de workflow. Toutefois, vous ne pourrez corriger les erreurs de validation que si vous avez ouvert le workflow pour modification dans l'éditeur de workflow.

### Conditions préalables

Vérifiez que vous disposez d'un workflow complet à valider, avec les éléments liés du schéma et les liaisons définies.

### Procédure

- 1 Cliquez sur la vue **Workflows**.
- 2 Localisez un workflow dans la liste hiérarchique **Workflows**.
- 3 (Facultatif) Cliquez avec le bouton droit sur le workflow et sélectionnez **Valider le workflow**.  
Si le workflow est valide, un message de confirmation s'affiche. Si le workflow n'est pas valide, une liste des erreurs apparaît.
- 4 (Facultatif) Fermez la boîte de dialogue Validation des workflows.
- 5 Cliquez avec le bouton droit sur le workflow et sélectionnez **Modifier** pour ouvrir l'éditeur de workflow.
- 6 Cliquez sur l'onglet **Schéma**.
- 7 Cliquez sur le bouton **Valider** dans la barre d'outils de l'onglet **Schéma**.  
Si le workflow est valide, un message de confirmation s'affiche. Si le workflow n'est pas valide, une liste des erreurs apparaît.
- 8 En cas de workflow non valide, cliquez sur l'un des messages d'erreur.

L'outil de validation met en évidence l'élément du schéma dans lequel l'erreur s'est produite en l'affublant d'une icône rouge. Lorsque c'est possible, l'outil de validation affiche une mesure de correction rapide.

- Si vous êtes d'accord avec la mesure de correction rapide proposée, cliquez dessus pour l'appliquer.
- Si vous n'êtes pas d'accord avec la mesure de correction rapide proposée, refermez la boîte de dialogue Validation des workflows et corrigez l'élément du schéma par vous-même.

---

**Important** Assurez-vous à chaque fois que le correctif proposé par Orchestrator convienne.

---



Par exemple, la mesure proposée pourrait être de supprimer un attribut inutilisé alors qu'en fait il se peut tout simplement que la liaison de cet attribut ne soit pas correcte.

**9** Répétez les étapes qui précèdent jusqu'à avoir éliminé la totalité des erreurs de validation.

Vous venez de valider un workflow et d'en avoir corrigé les erreurs de validation.

### Étape suivante

Vous pouvez désormais exécuter le workflow.

## Débogage de workflows

Orchestrator fournit un outil de débogage des workflows. Vous pouvez déboguer un workflow pour vérifier les paramètres et attributs d'entrée et de sortie au début de n'importe quelle activité, replacer les valeurs des paramètres ou des attributs au cours d'une exécution de workflow en mode modification et reprendre un workflow à partir de la dernière activité ayant échoué.

Vous pouvez déboguer des workflows à partir de la bibliothèque standard de workflows et des workflows personnalisés. Vous pouvez déboguer des workflows personnalisés alors que vous êtes en train de les développer dans l'éditeur de workflows.

### Déboguer un workflow

Vous pouvez déboguer les éléments d'un workflow en ajoutant des points de rupture aux éléments du schéma de workflow.

Lorsque vous atteignez un point de rupture, vous disposez de plusieurs options pour continuer le processus de débogage. Lorsque vous déboguez un élément du schéma de workflow, vous pouvez consulter les informations générales sur l'exécution du workflow, modifier les variables du workflow et afficher les messages du journal.

#### Conditions préalables

Connectez-vous au client Orchestrator comme un utilisateur qui peut exécuter des workflows.

#### Procédure





- 1** Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2** Cliquez sur la vue **Workflows**.
- 3** Sélectionnez un workflow dans la bibliothèque de workflows, puis cliquez sur l'onglet **Schéma**.
- 4** Pour ajouter des points de rupture à des éléments du schéma que vous souhaitez déboguer, cliquez avec le bouton droit sur un élément du workflow, puis sélectionnez **Insérer un point de rupture**.

Vous pouvez activer ou désactiver les points de rupture insérés.

- 5** Cliquez sur l'icône **Débogger le workflow** ()

Si le workflow demande des paramètres d'entrée, vous devez en indiquer.

- 6 Lorsque l'exécution du workflow s'interrompt à chaque point de rupture, sélectionnez l'une des options disponibles.

Option	Description
 <b>Reprendre</b>	Reprend l'exécution du workflow jusqu'au prochain point de rupture.
 <b>Entrer</b>	Vous permet d'entrer dans un élément de workflow.  <b>Note</b> Vous ne pouvez pas entrer dans un élément de workflow imbriqué lorsque vous déboguez un workflow dans l'éditeur de workflows.
 <b>Passer</b>	Passe l'élément concerné du schéma et interrompt l'exécution du workflow au prochain élément.
 <b>Étape retour</b>	Permet de quitter l'élément de workflow dans lequel vous êtes entré.

- 7 (Facultatif) L'onglet **Points de rupture** permet de modifier les points de rupture.

Vous pouvez activer, désactiver ou supprimer les points de rupture existants.

- 8 (Facultatif) L'onglet **Variables** permet de consulter les variables.

Vous pouvez modifier les valeurs de certaines variables au cours du processus de débogage.

## Exemple de débogage de workflow


Vous pouvez déboguer un workflow à partir de la bibliothèque standard de workflow.

Par exemple, si vous indiquez une adresse de destination incorrecte, vous pouvez corriger la valeur lorsque vous déboguez l'Exemple d'interaction avec le workflow d'e-mail.

### Conditions préalables

Connectez-vous au client Orchestrator comme un utilisateur qui peut exécuter des workflows Mail.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Workflows**.
- 3 Dans la liste hiérarchique des workflows, ouvrez **Bibliothèque > Courrier**.
- 4 Sélectionnez l'Exemple d'interaction avec le workflow d'e-mail, puis cliquez sur l'onglet **Schéma**.
- 5 Cliquez avec le bouton droit sur l'élément de workflow **Envoyer e-mail (Interaction)**, puis sélectionnez **Insérer un point de rupture**.
- 6 Cliquez sur l'icône **Débogger le workflow** (.

- 7 Fournissez les informations demandées.
  - a Dans la zone de texte **Adresse de destination**, commencez à taper une adresse de destination.  
Par exemple, *nom@entreprise.c*.
  - b Sélectionnez un groupe LDAP d'utilisateurs autorisés à répondre à la requête.
  - c Cliquez sur **Envoyer**.
- 8 Lorsque le point de rupture est atteint, cliquez sur l'icône **Entrer dans** (↩).
- 9 Dans l'onglet **Variables**, vérifiez les valeurs.
- 10 Dans la zone de texte **À**, saisissez l'adresse correcte du destinataire.  
Par exemple, *nom@entreprise.com*.
- 11 Cliquez sur l'icône **Reprendre** (▶) pour poursuivre l'exécution du workflow.

Le workflow s'appuie sur la valeur que vous avez indiquée au cours du débogage et poursuit son exécution.

## Workflows en cours d'exécution

Un workflow Orchestrator s'exécute suivant un enchaînement logique d'événements.

Lorsque vous exécutez un workflow, chaque élément de schéma du workflow s'exécute suivant la séquence ci-dessous.

- 1 Liaison par le workflow des attributs de jeton et des paramètres d'entrée du workflow avec les paramètres d'entrée de l'élément de schéma.
- 2 Exécution de l'élément de schéma.
- 3 Copie des paramètres de sortie de l'élément de schéma dans les attributs de jeton et les paramètres de sortie du workflow.
- 4 Stockage des attributs de jeton et les paramètres de sortie du workflow dans la base de données.
- 5 Lancement de l'exécution de l'élément de schéma suivant.

Renouvellement de la séquence pour chaque élément de schéma jusqu'à la fin du workflow.

## Points de contrôle du jeton du workflow

Lorsqu'un workflow s'exécute, chaque élément de schéma constitue un point de contrôle. Après exécution de chaque élément de schéma, Orchestrator stocke les attributs du jeton de workflow dans la base de données et l'exécution de l'élément de schéma suivant se lance. Si le workflow s'interrompt de façon inattendue, le serveur Orchestrator redémarre, l'élément de schéma actif à ce moment-là s'exécute à nouveau et le workflow se poursuit en partant du début de l'élément de schéma qui s'exécutait au moment où l'interruption a eu lieu. Pour autant, Orchestrator ne met pas en œuvre de gestion des transactions, ni de fonction de restauration.

## Fin du workflow

Le workflow s'achève si l'élément de schéma actif à ce moment-là est un élément de fin. Une fois que le workflow atteint un élément de fin, d'autres workflows ou applications peuvent utiliser ses paramètres de sortie.

## Exécuter un workflow dans l'éditeur de workflow

Il est possible d'exécuter un workflow pendant que vous le développez.

L'exécution d'un workflow dans l'éditeur de workflow vous permet de vérifier que le workflow s'exécute correctement sans interrompre le processus de développement. Vous pouvez afficher des messages du journal pour obtenir des informations sur le workflow exécuté. Si l'exécution du workflow renvoie des résultats inattendus, vous pouvez le modifier et l'exécuter à nouveau sans fermer l'éditeur de workflow.

### Conditions préalables

- Créez un workflow.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflow.
- Validez le workflow.

### Procédure

- 1 Cliquez sur l'onglet **Schéma**.
- 2 Cliquez sur **Exécuter**.
- 3 (Facultatif) Consultez les messages dans l'onglet **Journaux**.

## Exécuter un workflow

Vous pouvez réaliser des opérations automatisées dans vCenter Server en exécutant des workflows à partir de la bibliothèque standard ou des workflows que vous créez.

Vous avez la possibilité, par exemple, de créer une machine virtuelle en exécutant le workflow Créer une machine virtuelle simple.

### Conditions préalables

Vérifiez que vous avez bien configuré le plug-in vCenter Server. Pour plus d'informations, consultez *Installation et configuration de VMware vCenter Orchestrator*.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Exécuter** ou **Concevoir**.
- 2 Cliquez sur la vue **Workflows**.
- 3 Dans la liste hiérarchique de workflows, ouvrez **Bibliothèque > vCenter > Gestion des machines virtuelles > Basique** pour naviguer jusqu'au workflow Créer une machine virtuelle simple.
- 4 Cliquez avec le bouton droit sur le workflow Créer une machine virtuelle simple et sélectionnez **Démarrer le workflow**.

- 5 Indiquez les informations suivantes dans la boîte de dialogue des paramètres d'entrée de **Démarrer le workflow** pour créer une machine virtuelle dans un plug-in vCenter Server connecté à Orchestrator.

Option	Action
Nom de la machine virtuelle	Nommez la machine virtuelle <b>orchestrator-test</b> .
Dossier de la machine virtuelle	<p>a Cliquez sur <b>Non définie</b> comme valeur du <b>Dossier de la machine virtuelle</b>.</p> <p>b Sélectionnez un dossier de machine virtuelle dans l'inventaire.</p> <p>Le bouton <b>Sélectionner</b> sera inactif jusqu'à ce que vous ayez sélectionné un objet du type correct, dans ce cas, VC:VmFolder.</p>
Taille du nouveau disque en Go	Saisissez une valeur en chiffres qui convient.
Taille de la mémoire en Mo	Saisissez une valeur en chiffres qui convient.
Nombre de CPU virtuels	Sélectionnez le nombre de CPU qui convient dans le menu déroulant <b>Nombre de CPU virtuels</b> .
SE invité de la machine virtuelle	Cliquez sur le lien <b>Non défini</b> et sélectionnez un système d'exploitation invité dans la liste.
Hôte sur lequel créer la machine virtuelle	Cliquez sur <b>Non définie</b> comme valeur de l' <b>Hôte sur lequel créer la machine virtuelle</b> et localisez une machine hôte au sein de la hiérarchie de l'infrastructure de vCenter Server.
Pool de ressources	Cliquez sur <b>Non définie</b> comme valeur du <b>Pool de ressources</b> et localisez un pool de ressources au sein de la hiérarchie de l'infrastructure de vCenter Server.
Le réseau auquel se connecter	<p>Cliquez sur <b>Non définie</b> comme valeur du <b>Réseau auquel se connecter</b> et sélectionnez un réseau.</p> <p>Appuyez sur Entrée dans la zone de texte <b>Filtrer</b> pour voir s'afficher la liste des réseaux à votre disposition.</p>
Banque de données dans laquelle stocker les fichiers de la machine virtuelle	Cliquez sur <b>Non définie</b> comme valeur de la <b>Banque de données dans laquelle stocker les fichiers de la machine virtuelle</b> et localisez une banque de données au sein de la hiérarchie de l'infrastructure de vCenter Server.

- 6 Cliquez sur **Envoyer** pour exécuter le workflow.

Un jeton de workflow apparaît sous le workflow Créer une machine virtuelle simple qui montre l'icône du workflow en cours d'exécution.

- 7 Cliquez sur le jeton de workflow pour afficher le statut du workflow au fur et à mesure qu'il s'exécute.
- 8 Cliquez sur l'onglet **Événements** dans la vue du jeton de workflow pour suivre la progression du jeton de workflow jusqu'à la fin de l'exécution.
- 9 Cliquez sur la vue **Inventaire**.
- 10 Localisez le pool de ressources que vous avez défini au sein de la hiérarchie de l'infrastructure de vCenter Server.

Si la machine virtuelle n'apparaît pas dans la liste, cliquez sur le bouton actualiser pour recharger l'inventaire.

La machine virtuelle de orchestrator-test est présente dans le pool de ressources.

- 11 (Facultatif) Cliquez avec le bouton droit sur la machine virtuelle de orchestrator-test dans la vue **Inventaire** pour voir s'afficher une liste contextuelle des workflows que vous pouvez exécuter sur la machine virtuelle de orchestrator-test

L'exécution du workflow Créer une machine virtuelle simple s'est déroulée avec succès.

#### Étape suivante

Vous pouvez vous connecter à vSphere Client et gérer la nouvelle machine virtuelle.

## Reprise d'une exécution de workflow ayant échoué

Si un workflow échoue, Orchestrator propose une option pour reprendre son exécution à partir de la dernière activité ayant échoué.

Vous pouvez modifier les paramètres du workflow et essayer de relancer son exécution, ou conserver ces paramètres et modifier les composants externes qui touchent à l'exécution du workflow. Par exemple, si une exécution de workflow échoue à cause d'un problème dans un système tiers, vous pouvez apporter des modifications au système et relancer l'exécution du workflow à partir de l'activité qui a échoué ; ceci sans modifier les paramètres du workflow et sans répéter les activités qui avaient réussi.

## Définir le comportement pour la reprise d'une exécution de workflow ayant échoué

Vous pouvez définir le comportement régissant la reprise d'une exécution ayant échoué pour tous les workflows personnalisés. Les workflows par défaut de la bibliothèque utilisent le paramètre système par défaut pour la reprise des exécutions ayant échoué.

Vous pouvez modifier le comportement système par défaut en changeant un fichier de configuration. Reportez-vous à [Définir des propriétés personnalisées pour permettre la reprise des exécutions de workflows ayant échoué](#).

#### Conditions préalables

Vérifiez que vous disposez des autorisations nécessaires pour modifier le workflow.

#### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Workflows**.
- 3 Développez la liste hiérarchique des workflows pour naviguer jusqu'au workflow dont vous souhaitez définir le comportement.
- 4 Cliquez avec le bouton droit sur le workflow et sélectionnez **Modifier**.

L'éditeur de workflow s'ouvre.

- 5 Dans l'onglet **Général**, sélectionnez une option du menu déroulant **Comportement de reprise après échec**.

Option	Description
Système par défaut	Suit le comportement par défaut.
Activé	Si une exécution de workflow échoue, une fenêtre contextuelle s'affiche pour proposer la reprise de cette exécution.
Désactivé	Si un workflow échoue, il ne peut pas reprendre.

- 6 Cliquez sur **Enregistrer et fermer**.

## Définir des propriétés personnalisées pour permettre la reprise des exécutions de workflows ayant échoué

Par défaut, Orchestrator n'est pas configuré pour permettre la reprise des exécutions de workflows ayant échoué. Vous pouvez paramétrer Orchestrator de manière à assurer la reprise des exécutions de workflows ayant échoué et définir un délai d'expiration personnalisé après lequel ces exécutions ne pourront plus reprendre.

### Procédure

- 1 Sur le système du serveur Orchestrator, accédez à `/etc/vco/app-server/`.
- 2 Ouvrez le fichier de configuration `vmo.properties` dans un éditeur de texte.
- 3 Paramétrez Orchestrator de manière à ce qu'il permette la reprise des exécutions de workflows ayant échoué en modifiant la ligne suivante du fichier `vmo.properties`.

```
com.vmware.vco.engine.execute.resume-from-failed=true
```

- 4 Définissez un délai d'expiration personnalisé pour la reprise des exécutions de workflows ayant échoué en modifiant la ligne suivante du fichier `vmo.properties`.

```
com.vmware.vco.engine.execute.resume-from-failed.timeout-sec=<seconds>
```

La valeur que vous indiquez remplace le paramètre par défaut fixé à 86400 secondes.

- 5 Enregistrez le fichier `vmo.properties`.
- 6 Redémarrez le serveur Orchestrator.

## Reprendre une exécution de workflow ayant échoué

Si la reprise des exécutions ayant échoué est activée dans le workflow, vous pouvez reprendre une exécution de workflow à partir de la dernière activité où elle a échoué.

Lorsque l'option de reprise des exécutions de workflow ayant échoué est activée, vous pouvez modifier les paramètres du workflow et essayer qu'il reprenne son exécution à l'aide des options de la fenêtre contextuelle qui apparaît suite à l'échec. Vous pouvez également conserver ces paramètres et modifier les composants externes qui touchent à l'exécution du workflow. Si vous ne sélectionnez aucune option, l'exécution du workflow expire et ne pourra pas reprendre. Pour modifier le délai d'expiration, reportez-vous à [Définir des propriétés personnalisées pour permettre la reprise des exécutions de workflows ayant échoué](#).

#### Procédure

- 1 Dans le menu déroulant de la fenêtre contextuelle, sélectionnez **Reprendre**, puis cliquez sur **Suivant**.

Si vous sélectionnez **Annuler**, le workflow ne pourra pas reprendre ultérieurement.

- 2 (Facultatif) Modifiez les paramètres du workflow.
- 3 Cliquez sur **Envoyer**.

## Générer la documentation sur les workflows

Vous pouvez utiliser le format PDF pour exporter à tout moment la documentation sur un workflow ou un dossier de workflows que vous sélectionnez.

Le document exporté contient des informations détaillées sur le workflow sélectionné ou les workflows du dossier. Les informations relatives à chaque workflow incluent le nom, l'historique de la version du workflow, les attributs, la présentation des paramètres, le schéma du workflow et les actions du workflow. En outre, la documentation fournit le code source des actions utilisées.

#### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Exécuter** ou **Concevoir**.
- 2 Cliquez sur la vue **Workflows**.
- 3 Naviguez jusqu'au workflow ou dossier du workflow pour lequel vous souhaitez générer la documentation, puis cliquez dessus avec le bouton droit.
- 4 Sélectionnez **Générer la documentation**.
- 5 Parcourez pour localiser le dossier dans lequel vous souhaitez enregistrer le fichier PDF, indiquez un nom de fichier, puis cliquez sur **Enregistrer**.

Le fichier PDF contenant les informations sur le workflow sélectionné ou les workflows du dossier est enregistré dans votre système.

## Utiliser l'historique des versions du workflow

Vous pouvez utiliser l'historique des versions pour restaurer un workflow à une version précédente. Vous pouvez restaurer l'état du workflow à une version antérieure ou ultérieure. Vous pouvez également comparer les différences entre l'état du workflow en cours et une version enregistrée de ce workflow.



Orchestrator crée un élément d'historique des versions pour chaque workflow dès que vous enrichissez et enregistrez une version du workflow. Les modifications apportées par la suite ne transforment pas la version enregistrée. Par exemple, lorsque vous créez la version 1.0.0 du workflow et que vous l'enregistrez, l'état du workflow est stocké dans l'historique des versions. Si vous apportez des modifications au workflow, vous pourrez enregistrer l'état du workflow dans le client Orchestrator, mais vous ne pourrez pas appliquer ces transformations à la version 1.0.0 du workflow. Pour stocker les modifications dans l'historique des versions, vous devez créer une version de workflow suivante et l'enregistrer. L'historique des versions est conservé dans la base de données aux côtés du workflow.

Lorsque vous supprimez un workflow, Orchestrator indique l'élément comme supprimé de la base de données, mais il ne supprime pas l'historique des versions de l'élément. Vous pouvez ainsi restaurer les workflows supprimés. Reportez-vous à [Restaurer des workflows supprimés](#).

### Conditions préalables

Ouvrez le workflow à modifier dans l'éditeur de workflow.

### Procédure

- 1 Cliquez sur l'onglet **Général** dans l'éditeur de workflow, puis cliquez sur **Afficher l'historique des versions**.
- 2 Sélectionnez une version de workflow, puis cliquez sur **Différences avec version actuelle** pour comparer les différences.  
Une fenêtre s'affiche avec les différences entre la version en cours et la version sélectionnée.
- 3 Sélectionnez une version de workflow, puis cliquez sur **Restaurer** pour restaurer l'état du workflow.

---

**Attention** Si vous n'avez pas enregistré la version de workflow en cours, elle est supprimée de l'historique des versions et vous ne pouvez pas la restaurer.

---

L'état du workflow est restauré à la version sélectionnée.

## Restaurer des workflows supprimés

Vous pouvez restaurer des workflows supprimés de la bibliothèque de workflows.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Exécuter** ou **Concevoir**.
- 2 Cliquez sur la vue **Workflows**.
- 3 Naviguez jusqu'au dossier de workflows dans lequel vous souhaitez restaurer les workflows supprimés.
- 4 Cliquez avec le bouton droit sur le dossier, puis sélectionnez **Restaurer des workflows supprimés**.
- 5 Sélectionnez le workflow ou les workflows que vous souhaitez restaurer, puis cliquez sur **Restaurer**.

Les workflows restaurés apparaissent dans le dossier sélectionné.

## Développer un exemple de workflow simple

Le développement d'un exemple de workflow simple permet de démontrer les étapes les plus courantes du processus de développement des workflows.

L'exemple de workflow que vous êtes sur le point de créer permet de démarrer une machine virtuelle existante dans vCenter Server et d'envoyer un e-mail à l'administrateur pour confirmer que la machine virtuelle a démarré.

L'exemple de workflow effectue les tâches suivantes :

- 1 Invite l'utilisateur à sélectionner une machine virtuelle à démarrer.
- 2 Invite l'utilisateur à entrer une adresse e-mail à laquelle il enverra les notifications.
- 3 Vérifie si la machine virtuelle sélectionnée est déjà sous tension.
- 4 Envoie une demande à l'instance vCenter Server pour démarrer la machine virtuelle.
- 5 Attend que vCenter Server démarre la machine virtuelle et renvoie une erreur si la machine virtuelle ne démarre pas ou si le processus de démarrage est trop long.
- 6 Attend que vCenter Server démarre VMware Tools sur la machine virtuelle et renvoie une erreur si la machine virtuelle ne démarre pas ou si le démarrage de VMware Tools est trop long.
- 7 Vérifie que la machine virtuelle est active.
- 8 Envoie une notification à l'adresse e-mail fournie afin d'informer l'utilisateur que la machine virtuelle a démarré ou qu'une erreur s'est produite.

Le fichier ZIP des exemples Orchestrator disponible au téléchargement dans la page d'accueil de la documentation Orchestrator contient une version complète du workflow Démarrer une VM et envoyer e-mail.

Le processus de développement de l'exemple de workflow comprend plusieurs tâches.

### Conditions préalables

Avant d'essayer de développer cet exemple de workflow simple, veuillez lire [Concepts clés des workflows](#).

### Procédure

#### 1 Créer l'exemple de workflow simple

Vous devez commencer le processus de développement du workflow en créant le workflow dans le client Orchestrator.

#### 2 Créer le schéma de l'exemple de workflow simple

Vous pouvez créer un schéma de workflow dans l'éditeur de workflow. Le schéma de workflow contient les éléments exécutés par le workflow et détermine le déroulement logique du workflow.

### 3 (Facultatif) Créer les zones de l'exemple de workflow simple

Vous pouvez mettre en valeur les différentes zones du workflow en ajoutant des notes de workflow de couleurs variées. La création de différentes zones de workflow facilite la lecture et la compréhension des schémas de workflow complexes.

### 4 Définir les paramètres de l'exemple de workflow simple

Dans cette phase de développement du workflow, vous définissez les paramètres d'entrée que le workflow doit exécuter. Pour l'exemple de workflow, vous avez besoin d'un paramètre d'entrée pour que la machine virtuelle se mette sous tension et d'un paramètre pour l'adresse e-mail de la personne à informer sur le résultat de l'opération. Lorsque les utilisateurs exécutent le workflow, il leur est demandé d'indiquer la machine virtuelle à mettre sous tension et une adresse e-mail.

### 5 Définir les liaisons de décision de l'exemple de workflow simple

Vous pouvez relier les éléments d'un workflow entre eux dans l'onglet **Schéma** de l'éditeur de workflow. Les liaisons de décision définissent la manière dont les éléments Décision comparent les paramètres d'entrée reçus dans l'instruction de décision et génèrent des paramètres de sortie en fonction de la correspondance entre les paramètres d'entrée et l'instruction de décision.

### 6 Relier les éléments d'action de l'exemple de workflow simple

Vous pouvez relier les éléments d'un workflow entre eux dans l'éditeur de workflow. Les liaisons définissent la manière dont les éléments d'action traitent les paramètres d'entrée et génèrent les paramètres de sortie.

### 7 Relier les éléments Tâche inscriptible de l'exemple de workflow simple

Vous pouvez relier les éléments d'un workflow entre eux dans l'onglet **Schéma** de l'éditeur de workflow. Les liaisons définissent la manière dont les éléments de tâche en script traitent les paramètres d'entrée et génèrent les paramètres de sortie. Vous pouvez également relier les éléments Tâche inscriptible à leurs fonctions JavaScript.

### 8 Définir les liaisons d'exception de l'exemple de workflow simple

Vous pouvez définir les liaisons d'exception dans l'onglet **Schéma** de l'éditeur de workflow. Les liaisons d'exception définissent la façon dont les éléments traitent les erreurs.

### 9 Définir les propriétés en lecture seule des attributs de l'exemple de workflow simple

Vous pouvez définir si les paramètres et attributs sont des constantes en lecture seule ou des variables sur lesquelles vous disposez de droits d'écriture. Vous avez aussi la possibilité de fixer des limitations aux valeurs que les utilisateurs peuvent renseigner dans les paramètres d'entrée.

### 10 Définir les propriétés du paramètre de l'exemple de workflow simple

Vous avez la possibilité de définir les propriétés du paramètre dans l'éditeur de workflow. Définir les propriétés du paramètre a une incidence sur le comportement du paramètre et fixe des contraintes aux valeurs possibles pour celui-ci.

## 11 Définir l'agencement de la boîte de dialogue des paramètres d'entrée d'un exemple de workflow simple

L'agencement ou la présentation de la boîte de dialogue des paramètres d'entrée se définit dans l'éditeur de workflow. La boîte de dialogue des paramètres d'entrée s'ouvre lorsque les utilisateurs exécutent un workflow qui a besoin d'une saisie de paramètres pour s'exécuter.

## 12 Valider et exécuter l'exemple de workflow simple

Une fois un workflow créé, vous avez la possibilité de le valider afin d'y déceler toute erreur éventuelle. Si le workflow est exempt d'erreurs, vous pouvez l'exécuter.

# Créer l'exemple de workflow simple

Vous devez commencer le processus de développement du workflow en créant le workflow dans le client Orchestrator.

### Conditions préalables

Vérifiez que les composants suivants sont installés et configurés sur le système.

- vCenter Server, contrôle de certaines machines virtuelles, dont au moins une hors tension
- Accéder à un serveur SMTP
- Adresse e-mail valide

Pour plus d'informations sur l'installation et la configuration de vCenter Server, consultez la documentation *Installation et configuration de vSphere*. Pour plus d'informations sur la configuration d'Orchestrator de manière à ce qu'il utilise un serveur SMTP, reportez-vous à *Installation et configuration d'VMware vRealize Orchestrator*.

Pour rédiger un workflow, vous devez disposer d'un compte d'utilisateur Orchestrator avec les autorisations **Afficher**, **Exécuter**, **Inspecter**, **Modifier** et, de préférence, **Administrateur** sur le serveur ou le dossier de workflow dans lequel vous travaillez.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
  - 2 Cliquez sur la vue **Workflows**.
  - 3 Cliquez avec le bouton droit sur la racine de la liste des workflows, puis sélectionnez **Ajouter un dossier**.
  - 4 Nommez le nouveau dossier **Exemples de workflows**, puis cliquez sur **OK**.
  - 5 Cliquez avec le bouton droit sur le dossier **Exemples de workflows**, puis sélectionnez **Nouveau workflow**.
  - 6 Nommez le nouveau workflow **Démarrage VM et envoyer e-mail**, puis cliquez sur **OK**.
- L'éditeur de workflow s'ouvre.

- 7 Dans l'onglet **Général**, cliquez sur les chiffres du numéro de version pour insérer le numéro de version.

Étant donné qu'il s'agit de la première création du workflow, définissez la version sur **0.0.1**.

- 8 Cliquez sur la valeur **Comportement de redémarrage du serveur** de l'onglet **Général** pour définir si le workflow reprendra en cas de redémarrage du serveur.

- 9 Saisissez la description des fonctions du workflow dans la zone de texte **Description** de l'onglet **Général**.

Par exemple, vous pouvez ajouter la description suivante.

**Ce workflow démarre une machine virtuelle et envoie un message de confirmation à l'administrateur d'Orchestrator.**

- 10 Cliquez sur **Enregistrer** en bas de l'onglet **Général**.

Vous avez créé un workflow appelé Démarrer une VM et envoyer un e-mail mais vous n'avez pas défini ses fonctions.

#### Étape suivante

Créez maintenant le schéma de workflow.

## Créer le schéma de l'exemple de workflow simple

Vous pouvez créer un schéma de workflow dans l'éditeur de workflow. Le schéma de workflow contient les éléments exécutés par le workflow et détermine le déroulement logique du workflow.

#### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

#### Procédure

- 1 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.
- 2 Dans le menu **Générique**, faites glisser un élément Décision vers la flèche qui relie l'élément Démarrer et l'élément End dans le schéma.
- 3 Double-cliquez sur l'élément Décision et changez son nom par **VM sous tension ?**.  
L'élément Décision correspond à une fonction booléenne qui vérifie si la machine virtuelle est déjà sous tension.
- 4 Dans le menu **Générique**, faites glisser un élément d'action vers la flèche rouge qui relie l'élément Décision et un élément End.  
La boîte de dialogue de sélection de l'action apparaît.

- 5 Tapez **démarrer** dans la zone de texte **Filtrer**, sélectionnez l'action **startVM** à partir de la liste filtrée des actions, puis cliquez sur **Sélectionner**.
- 6 Faites glisser les éléments d'action suivants l'un après l'autre vers la flèche bleue qui relie l'élément d'action startVM à un élément End.

<b>vim3WaitTaskEnd</b>	Interrompt l'exécution du workflow et effectue un test ping sur une tâche vCenter Server continue à des intervalles réguliers jusqu'à ce que cette tâche soit terminée. L'action startVM démarre une machine virtuelle et l'action vim3WaitTaskEnd interrompt le workflow pendant le démarrage de la machine virtuelle. Après le démarrage de la machine virtuelle, le vim3WaitTaskEnd laisse le workflow reprendre.
<b>vim3WaitToolsStarted</b>	Interrompt l'exécution du workflow et attend que VMware Tools démarre sur la machine virtuelle cible.

- 7 Dans le menu **Générique**, faites glisser un élément Tâche inscriptible vers la flèche bleue qui relie l'élément d'action vim3WaitToolsStarted à un élément End.
- 8 Double-cliquez sur l'élément Tâche inscriptible et renommez-le par **OK**.
- 9 Faites glisser un autre élément Tâche inscriptible vers la flèche verte qui relie l'élément Décision VM powered on? à un élément End, puis nommez cet élément Tâche inscriptible **Déjà démarré**.
- 10 Modifiez les liaisons de l'élément Tâche inscriptible Already started.
  - a Faites glisser l'élément Tâche inscriptible Already started sur la gauche de l'élément d'action startVM.
  - b Supprimez la flèche bleue qui relie l'élément Tâche inscriptible Already started à un élément End.
  - c Reliez l'élément Tâche inscriptible Already started à l'élément d'action vim3WaitToolsStarted avec la flèche bleue.
- 11 Faites glisser les éléments de tâche inscriptible depuis le menu **Générique** vers le schéma.
  - Faites glisser un élément Tâche inscriptible vers l'élément d'action startVM, puis nommez l'élément Tâche inscriptible **Échec démarrage VM**.
  - Faites glisser un élément Tâche inscriptible vers l'élément d'action vim3WaitTaskEnd, puis nommez l'élément Tâche inscriptible **Délai d'expiration 1**.
  - Faites glisser un élément Tâche inscriptible vers l'élément d'action vim3WaitToolsStarted, puis nommez l'élément Tâche inscriptible **Délai d'expiration 2**.
  - Faites glisser un élément Tâche inscriptible vers la flèche bleue qui relie l'élément Tâche inscriptible OK à un élément End, nommez le nouvel élément Tâche inscriptible **Envoyer e-mail**, puis faites-le glisser sur la droite de l'élément Tâche inscriptible OK.
  - Reliez les éléments Tâche inscriptible Start VM Failed, Timeout 1 et Timeout 2 à l'élément Tâche inscriptible Send Email avec les flèches bleues.

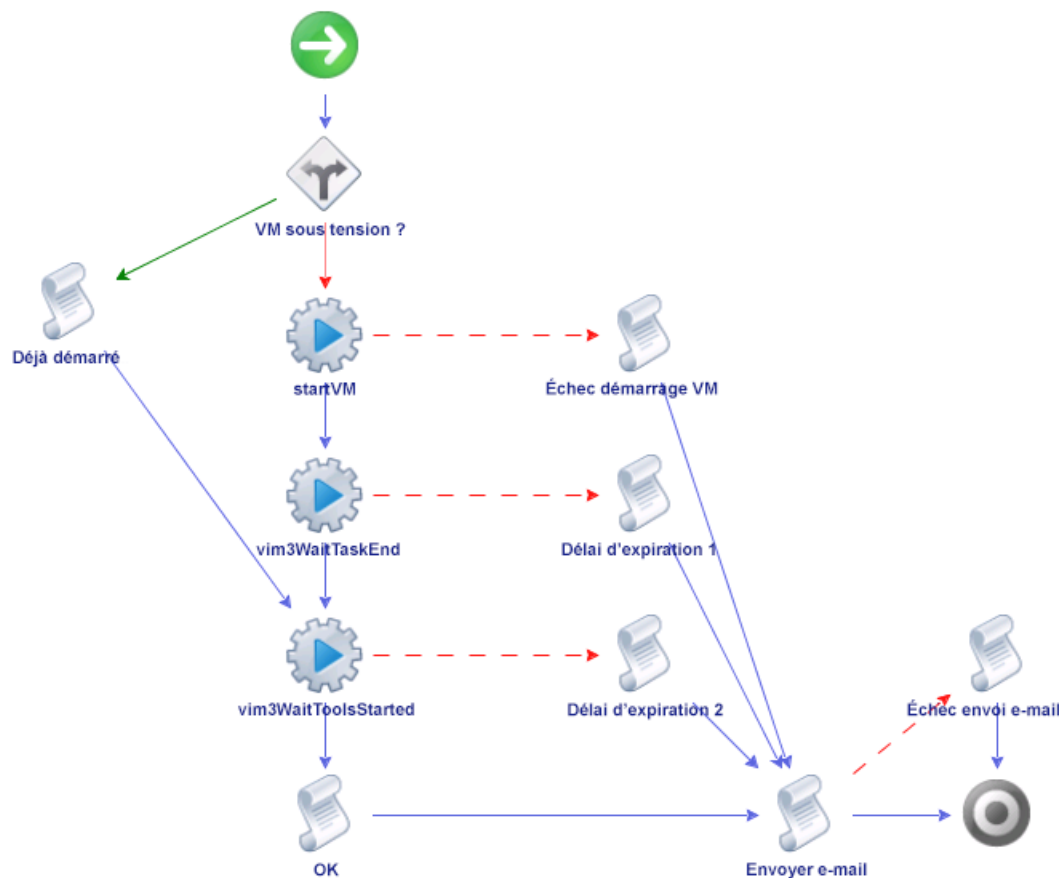
- Faites glisser un élément Tâche inscriptible vers l'élément Tâche inscriptible Send Email, nommez le nouvel élément Tâche inscriptible **Échec envoi e-mail**, faites-le glisser sur la droite de l'élément Tâche inscriptible Timeout 2, puis reliez-le à l'élément End avec une flèche bleue.

12 Faites glisser l'élément End sur la droite de l'élément Tâche inscriptible Send Email.

13 Cliquez sur **Enregistrer** en bas de l'onglet **Schéma**.

La figure ci-dessous illustre la mise en forme des éléments de schéma des workflows Démarrage VM et Envoyer e-mail.

**Figure 1-10. Liaison des éléments de l'exemple de workflow Démarrage VM et Envoyer e-mail.**



### Étape suivante

Vous pouvez mettre en surbrillance les différentes zones du workflow.

## Créer les zones de l'exemple de workflow simple

Vous pouvez mettre en valeur les différentes zones du workflow en ajoutant des notes de workflow de couleurs variées. La création de différentes zones de workflow facilite la lecture et la compréhension des schémas de workflow complexes.

## Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

## Procédure

- 1 Faites glisser un élément de note de workflow depuis le menu **Générique** vers l'éditeur de workflow.
- 2 Placez la note de workflow au-dessus de l'élément Tâche inscriptible `Already started`.
- 3 Faites glisser les bords de la note de workflow pour la redimensionner de manière à ce qu'elle entoure l'élément Tâche inscriptible `Already started`.
- 4 Double-cliquez sur le texte et ajoutez une description.

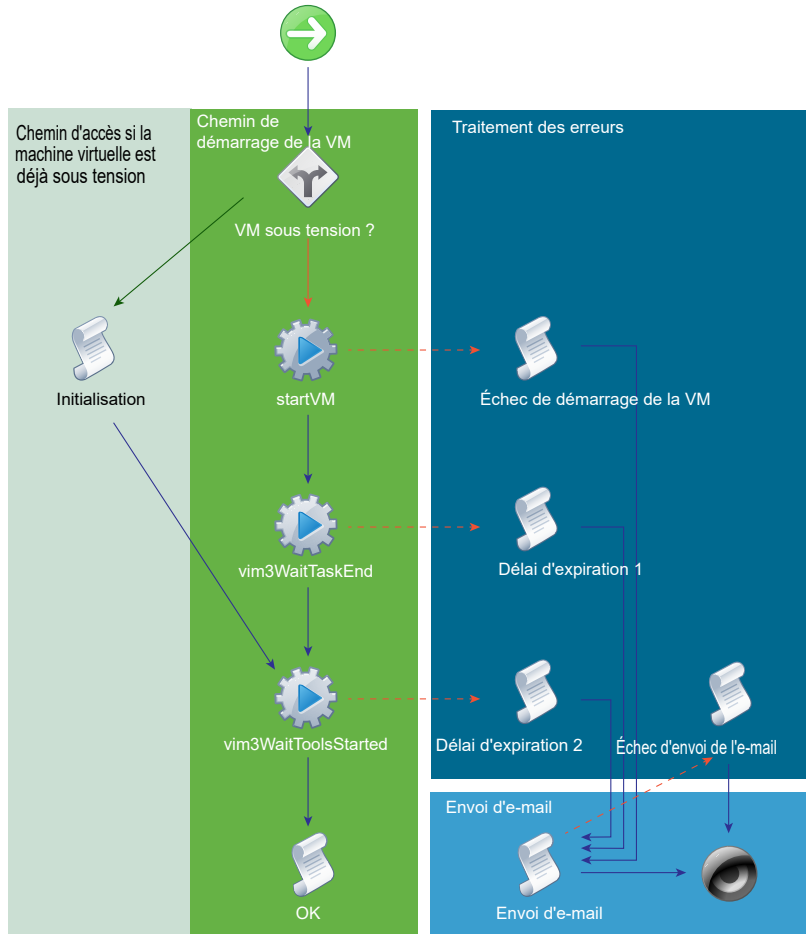
Par exemple, **Chemin d'accès si la machine virtuelle est déjà sous tension.**

- 5 Appuyez sur Ctrl+E pour sélectionner la couleur de l'arrière-plan.
- 6 Répétez les étapes précédentes pour mettre en surbrillance d'autres zones du workflow.
  - Insérez une note sur la séquence verticale d'éléments depuis l'élément Décision `VM powered on?` jusqu'à l'élément `OK`. Ajoutez la description **Chemin Démarrage VM**
  - Placez une note sur le `startVM failed`, les éléments Tâche inscriptible `Timeout` et l'élément Tâche inscriptible `Send Email Failed`. Ajoutez la description **Gestion des erreurs.**
  - Placez une note sur l'élément Tâche inscriptible `Send Email`. Ajoutez la description **Envoyer e-mail.**

La figure suivante illustre l'apparence que devraient revêtir vos zones de workflow.



Figure 1-11. Zones de l'exemple de workflow Démarrage VM et Envoyer e-mail.



### Étape suivante

Vous devez maintenant définir les attributs et les paramètres d'entrée et de sortie du workflow.

## Définir les paramètres de l'exemple de workflow simple

Dans cette phase de développement du workflow, vous définissez les paramètres d'entrée que le workflow doit exécuter. Pour l'exemple de workflow, vous avez besoin d'un paramètre d'entrée pour que la machine virtuelle se mette sous tension et d'un paramètre pour l'adresse e-mail de la personne à informer sur le résultat de l'opération. Lorsque les utilisateurs exécutent le workflow, il leur est demandé d'indiquer la machine virtuelle à mettre sous tension et une adresse e-mail.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

## Procédure

- 1 Cliquez sur l'onglet **Entrées** dans l'éditeur de workflow.
- 2 Cliquez avec le bouton droit dans l'onglet **Entrées**, puis sélectionnez **Ajouter un paramètre**.  
Un paramètre nommé `arg_in_0` apparaît dans l'onglet **Entrées**.
- 3 Cliquez sur `arg_in_0`.
- 4 Entrez le nom **vm** dans la boîte de dialogue Choisir le nom de l'attribut, puis cliquez sur **OK**.
- 5 Cliquez sur la zone de texte **Type**, puis entrez **vc:virtualm** dans la zone de texte de recherche de la boîte de dialogue Type de paramètre.
- 6 Sélectionnez **VC:VirtualMachine** dans la liste de paramètres proposée, puis cliquez sur **Accepter**.
- 7 Ajoutez une description du paramètre dans la zone de texte **Description**.  
Par exemple, entrez **machine virtuelle à mettre sous tension**.
- 8 Répétez les étapes de [Étape 2](#) à [Étape 7](#) pour créer un deuxième paramètre d'entrée avec les valeurs suivantes.
  - Nom : `toAddress`
  - Type : chaîne
  - Description :  
**Adresse e-mail de la personne à informer du résultat de ce workflow.**
- 9 Cliquez sur **Enregistrer** en bas de l'onglet **Entrées**.

Vous venez de définir les paramètres d'entrée du workflow.

## Étape suivante

Vous devez maintenant définir les liaisons entre les paramètres des éléments.

## Définir les liaisons de décision de l'exemple de workflow simple

Vous pouvez relier les éléments d'un workflow entre eux dans l'onglet **Schéma** de l'éditeur de workflow. Les liaisons de décision définissent la manière dont les éléments Décision comparent les paramètres d'entrée reçus dans l'instruction de décision et génèrent des paramètres de sortie en fonction de la correspondance entre les paramètres d'entrée et l'instruction de décision.

## Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- [Définir les paramètres de l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

## Procédure

- 1 Dans l'onglet **Schéma**, cliquez sur l'icône **Modifier** (✎) de l'élément Décision **VM Powered On?**.
- 2 Dans l'onglet **Décision**, cliquez sur le bouton **Non définie (NULL)**, puis sélectionnez **vm** en tant que paramètre d'entrée de l'élément Décision dans la liste proposée de paramètres.
- 3 Sélectionnez l'instruction **L'état de l'alimentation égale** dans la liste des instructions de décision du menu déroulant.

Un bouton **Non défini** apparaît dans la zone de texte Valeur. Il vous propose un choix limité de valeurs possibles.

- 4 Sélectionnez **poweredOn**.
- 5 Cliquez sur **Enregistrer** en bas de l'onglet **Schéma** de l'éditeur de workflow.

Vous venez de définir une instruction True ou False qui permettra à l'élément Décision de comparer les valeurs à la réception du paramètre d'entrée.

## Étape suivante

Vous devez maintenant définir les liaisons des autres éléments du workflow.

## Relier les éléments d'action de l'exemple de workflow simple

Vous pouvez relier les éléments d'un workflow entre eux dans l'éditeur de workflow. Les liaisons définissent la manière dont les éléments d'action traitent les paramètres d'entrée et génèrent les paramètres de sortie.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- [Définir les paramètres de l'exemple de workflow simple.](#)
- [Définir les liaisons de décision de l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

## Procédure

- 1 Dans l'onglet **Schéma**, cliquez sur l'icône **Modifier** (✎) de l'élément d'action **startVM**.
- 2 Définissez les informations générales suivantes de l'onglet **Infos**.

Option	Action
Interaction	Sélectionnez <b>Aucune interaction externe</b> .
État du groupe	Cochez la case et ajoutez le texte <b>Envoi du démarrage de la VM</b> .
Description	Laissez le texte Démarrer / Reprendre une VM. Renvoyez la tâche de démarrage.

**3 Cliquez sur l'onglet *Entrée*.**

L'onglet **Entrée** affiche les deux paramètres d'entrée possibles pour l'action `startVM`, `vm` et `host`.

Orchestrator relie automatiquement le paramètre `vm` à `vm[in-parameter]`, car l'action `startVM` peut uniquement accepter un objet `VC:VirtualMachine` en tant que paramètre d'entrée. Orchestrator détecte le paramètre `vm` que vous avez défini lorsque vous avez établi les paramètres d'entrée du workflow, et le relie automatiquement à l'action.

**4 Définissez `host` sur **NULL**.**

Ce paramètre étant facultatif, vous pouvez le définir sur `Null`. Cependant, si vous le laissez sur **Non défini**, la validation du workflow échoue.

**5 Cliquez sur l'onglet *Sortie*.**

`actionResult`, le paramètre de sortie par défaut que toutes les actions génèrent, s'affiche.

**6 Cliquez sur **Non défini** pour le paramètre `actionResult`.****7 Cliquez sur **Créer un paramètre/attribut dans le workflow**.**

La boîte de dialogue Informations relatives aux paramètres affiche les valeurs que vous pouvez définir pour ce paramètre de sortie. Le type de paramètre de sortie pour l'action `startVM` est un objet `VC:Task`.

**8 Nommez le paramètre `powerOnTask` et rédigez une description.**

Par exemple, **Comporte le résultat de la mise sous tension d'une VM**.

**9 Cliquez sur **Créer un ATTRIBUT de workflow avec le même nom**, puis cliquez sur **OK** pour quitter la boîte de dialogue Informations relatives aux paramètres.****10 Répétez les étapes précédentes pour relier les paramètres d'entrée et de sortie aux éléments d'action `vim3WaitTaskEnd` et `vim3WaitToolsStarted`.**

[Liaisons de l'élément Action d'un exemple de workflow simple](#) répertorie les liaisons pour les éléments d'action `vim3WaitTaskEnd` et `vim3WaitToolsStarted`.

**11 Cliquez sur **Enregistrer** en bas de l'onglet **Schéma** de l'éditeur de workflow.**

Les paramètres d'entrée et de sortie des éléments d'action sont liés aux types et valeurs de paramètres appropriés.

**Étape suivante**

Reliez maintenant les éléments Tâche inscriptible et définissez leurs fonctions.

**Liaisons de l'élément Action d'un exemple de workflow simple**

Les liaisons définissent la façon dont les éléments Action de l'exemple de workflow simple traitent les paramètres d'entrée et de sortie.

Lors de la définition de liaisons, Orchestrator présente les paramètres que vous avez déjà définis dans le workflow comme candidats à la liaison. Si vous n'avez pas encore défini le paramètre requis dans le workflow, le seul choix pour le paramètre sera NULL. Cliquez sur **Créer un paramètre/attribut dans le workflow** pour créer un nouveau paramètre.

### Action vim3WaitTaskEnd

L'élément Action vim3WaitTaskEnd déclare des constantes pour suivre la progression d'une tâche et un taux d'interrogation. Le tableau suivant montre les liaisons de paramètre d'entrée et de sortie que l'action vim3WaitTaskEnd requiert.

**Tableau 1-53. Valeurs de liaison de l'action vim3WaitTaskEnd**

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
task	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : powerOnTask</li> <li>■ Paramètre source : task[attribute]</li> <li>■ Type : VC:Task</li> <li>■ Description : <b>contient le résultat de la mise sous tension d'une VM.</b></li> </ul>
progress	IN	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : progress</li> <li>■ Paramètre source : progress[attribute]</li> <li>■ Type : booléen</li> <li>■ Valeur : non (false)</li> <li>■ Description : <b>progression du journal le temps que la tâche de vCenter Server s'achève.</b></li> </ul>

Tableau 1-53. Valeurs de liaison de l'action vim3WaitTaskEnd (suite)

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
pollRate	IN	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : pollRate</li> <li>■ Paramètre source : pollRate[attribute]</li> <li>■ Type : nombre</li> <li>■ Valeur : 2</li> <li>■ Description : <b>taux d'interrogation en secondes selon lequel vim3WaitTaskEnd vérifie l'avancement de la tâche de vCenter Server.</b></li> </ul>
actionResult	OUT	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : actionResult[attribute]</li> <li>■ Paramètre source : returnedManagedObject[attribute]</li> <li>■ Type : n'importe lequel</li> <li>■ description : <b>objet géré renvoyé par l'action waitTaskEnd.</b></li> </ul>

### Action vim3WaitToolsStarted

L'élément Action vim3WaitToolsStarted attend que les VMware Tools soient installés sur une machine virtuelle, puis définit un taux d'interrogation et un délai d'expiration. Le tableau suivant montre les liaisons de paramètre d'entrée que l'action vim3WaitToolsStarted exige.

L'élément Action vim3WaitToolsStarted étant dépourvu de sortie, il ne nécessite aucune liaison en sortie.

Tableau 1-54. Valeurs de liaison de l'action vim3WaitToolsStarted

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
vm	IN	Liaison automatique	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[in-parameter]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Valeur : non modifiable, la variable n'est pas un attribut de workflow.</li> <li>■ Description : <b>machine virtuelle à démarrer.</b></li> </ul>
pollingRate	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : pollingRate</li> <li>■ Paramètre source : pollingRate[attribute]</li> <li>■ Type : nombre</li> <li>■ Description : <b>taux d'interrogation en secondes selon lequel vim3WaitTaskEndvérifie l'avancement de la tâche de vCenter Server.</b></li> </ul>
timeout	IN	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : timeout</li> <li>■ Paramètre source : timeout[attribute]</li> <li>■ Type : nombre</li> <li>■ Valeur : <b>10</b></li> <li>■ Description : <b>délai d'expiration que vim3WaitToolsStarted attend pour générer une exception.</b></li> </ul>

## Relier les éléments Tâche inscriptible de l'exemple de workflow simple

Vous pouvez relier les éléments d'un workflow entre eux dans l'onglet **Schéma** de l'éditeur de workflow. Les liaisons définissent la manière dont les éléments de tâche en script traitent les paramètres d'entrée et génèrent les paramètres de sortie. Vous pouvez également relier les éléments Tâche inscriptible à leurs fonctions JavaScript.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- [Définir les paramètres de l'exemple de workflow simple.](#)

- Définir les liaisons de décision de l'exemple de workflow simple.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

- 1 Dans l'onglet **Schéma**, cliquez sur l'icône **Modifier** (✎) de l'élément Tâche inscriptible Already Started.
- 2 Définissez les informations générales suivantes de l'onglet **Infos**.

Option	Action
Interaction	Sélectionnez <b>Aucune interaction externe</b> .
État du groupe	Cochez la case et ajoutez le texte <b>VM déjà mise sous tension</b> .
Description	Laissez le texte La VM est déjà mise sous tension pour contourner startVM et waitTaskEnd, et vérifiez si les outils de la VM sont prêts à l'emploi.

- 3 Cliquez sur l'onglet **Entrée**.  
Étant donné qu'il s'agit d'un élément Tâche inscriptible personnalisé, aucune propriété n'est prédéfinie.
- 4 Cliquez sur l'icône **Relier à un paramètre/attribut du workflow** (🔗).
- 5 Sélectionnez vm dans la liste de paramètres proposée.
- 6 Laissez vides les onglets **Sortie** et **Exception**.  
Cet élément ne génère ni paramètres de sortie, ni exception.
- 7 Cliquez sur l'onglet **Script**.
- 8 Ajoutez la fonction JavaScript suivante.

```
//Writes the following event in the Orchestrator database
Server.log("VM '"+ vm.name +"' already started");
```

- 9 Répétez les étapes précédentes pour relier les paramètres d'entrée restants aux autres éléments Tâche inscriptible.

[Liaisons de l'élément Tâche inscriptible de l'exemple de workflow simple](#) répertorie les liaisons pour les éléments Tâche inscriptible Start VM failed, Timeout ou Error, Send Email Failed et OK.

- 10 Cliquez sur **Enregistrer** en bas de l'onglet **Schéma** de l'éditeur de workflow.

Vous avez relié les éléments Tâche inscriptible à leurs paramètres d'entrée et de sortie, et rédigé les scripts qui définissent leur fonction.

### Étape suivante

Vous devez maintenant définir le traitement des exceptions.



## Liaisons de l'élément Tâche inscriptible de l'exemple de workflow simple

Les liaisons définissent de quelle façon les éléments Tâche inscriptible de l'exemple de workflow simple traitent les paramètres d'entrée. Vous créerez aussi une liaison entre les éléments Tâche inscriptible et leurs fonctions JavaScript.

Lors de la définition de liaisons, Orchestrator présente les paramètres que vous avez déjà définis dans le workflow comme candidats à la liaison. Si vous n'avez pas encore défini le paramètre exigé dans le workflow, le seul choix pour le paramètre sera NULL. Cliquez sur **Créer un paramètre/attribut dans le workflow** pour créer un nouveau paramètre.

### Tâche inscriptible Échec de démarrage de la VM

L'élément Tâche inscriptible Échec de démarrage de la VM traite toute exception que génère l'action startVM en définissant le contenu de l'e-mail signalant l'échec du démarrage de la machine virtuelle et en inscrivant l'événement dans le journal d'Orchestrator.

Le tableau suivant montre les liaisons de paramètre d'entrée et de sortie que l'élément Tâche inscriptible Échec de démarrage de la VM exige.

**Tableau 1-55. Liaisons de l'élément Tâche inscriptible Échec de démarrage de la VM**

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
vm	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[in-parameter]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>machine virtuelle à allumer.</b></li> </ul>
errorCode	IN	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : errorCode</li> <li>■ Paramètre source : errorCode[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>saisit toute exception générée lors de la mise sous tension de la VM.</b></li> </ul>
body	OUT	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : body</li> <li>■ Paramètre source : body[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>corps de l'e-mail</b></li> </ul>

L'élément Tâche inscriptible Échec de démarrage de la VM réalise la fonction en script suivante.

```
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);
```

## Élément Tâche inscriptible Délai d'expiration 1

L'élément Tâche inscriptible Délai d'expiration 1 traite toute exception que génère l'action `vim3WaitTaskEnd` en définissant le contenu de l'e-mail signalant l'échec du démarrage de la machine virtuelle et en inscrivant l'événement dans le journal d'Orchestrator.

Le tableau suivant montre les liaisons des paramètres d'entrée et de sortie que l'élément Tâche inscriptible Délai d'expiration 1 exige.

**Tableau 1-56. Liaisons de l'élément Tâche inscriptible Délai d'expiration 1**

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
vm	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[in-parameter]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>machine virtuelle à démarrer.</b></li> </ul>
errorCode	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : errorCode</li> <li>■ Paramètre source : errorCode[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>saisit toute exception générée lors de la mise sous tension de la VM.</b></li> </ul>
body	OUT	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : body</li> <li>■ Paramètre source : body[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>corps de l'e-mail</b></li> </ul>

L'élément Tâche inscriptible Délai d'expiration 1 exige la fonction en script suivante.

```
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);
```

## Élément Tâche inscriptible Délai d'expiration 2

L'élément Tâche inscriptible Délai d'expiration 2 traite toute exception que génère l'action `vim3WaitToolsStarted` en définissant le contenu de l'e-mail signalant l'échec du démarrage de la machine virtuelle et en inscrivant l'événement dans le journal d'Orchestrator.

Le tableau suivant montre les liaisons des paramètres d'entrée et de sortie que l'élément Tâche inscriptible Délai d'expiration 2 exige.

Tableau 1-57. Liaisons de l'élément Tâche inscriptible Délai d'expiration 2

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
vm	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[in-parameter]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>machine virtuelle à allumer.</b></li> </ul>
errorCode	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : errorCode</li> <li>■ Paramètre source : errorCode[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>saisit toute exception générée lors de la mise sous tension de la VM.</b></li> </ul>
body	OUT	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : body</li> <li>■ Paramètre source : body[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>corps de l'e-mail</b></li> </ul>

L'élément Tâche inscriptible Délai d'expiration 2 exige la fonction en script suivante.

```
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);
```

### Élément Tâche inscriptible OK

L'élément Tâche inscriptible OK est averti du fait que le démarrage de la machine virtuelle a abouti, définit le contenu d'un e-mail signalant le démarrage réussi de la machine virtuelle et inscrit l'événement dans le journal d'Orchestrator.

Le tableau suivant montre les liaisons des paramètres d'entrée et de sortie que l'élément Tâche inscriptible OK exige.

Tableau 1-58. Liaisons de l'élément Tâche inscriptible OK

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
vm	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[in-parameter]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>machine virtuelle à allumer.</b></li> </ul>
body	OUT	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : body</li> <li>■ Paramètre source : body[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>corps de l'e-mail</b></li> </ul>

L'élément Tâche inscriptible OK exige la fonction en script suivante.

```
body = "The VM '"+vm.name+"' has started successfully and is ready for use";
//Writes the following event in the Orchestrator database
Server.log(body);
```

### Élément Tâche inscriptible Échec d'envoi de l'e-mail

L'élément Tâche inscriptible Échec d'envoi de l'e-mail est averti du fait que l'envoi de l'e-mail a échoué et inscrit l'événement dans le journal d'Orchestrator.

Le tableau suivant montre les liaisons de paramètre d'entrée et de sortie que l'élément Tâche inscriptible Échec d'envoi de l'e-mail exige.

**Tableau 1-59. Liaisons de l'élément Tâche inscriptible Échec d'envoi de l'e-mail**

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
vm	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[in-parameter]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>machine virtuelle à allumer.</b></li> </ul>
toAddress	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : toAddress</li> <li>■ Paramètre source : toAddress[in-parameter]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>adresse électronique de la personne à informer du résultat de ce workflow</b></li> </ul>
emailErrorCode	IN	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : emailErrorCode</li> <li>■ Paramètre source : emailErrorCode[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>saisit toute exception générée lors de l'envoi d'un e-mail.</b></li> </ul>

L'élément Tâche inscriptible Tâche inscriptible Échec d'envoi de l'e-mail exige la fonction en script suivante.

```
//Writes the following event in the Orchestrator database
Server.error("Couldn't send result email to '"+toAddress+"' for VM '"+vm.name+"', exception found: "+emailErrorCode);
```

### Élément Tâche inscriptible Envoyer e-mail

Le workflow Démarrer une VM et envoyer un e-mail a pour but d'informer un administrateur lorsque le workflow démarre une machine virtuelle. Pour ce faire, vous devez définir la tâche inscriptible qui enverra l'e-mail. Pour envoyer l'e-mail, l'élément Tâche inscriptible Envoyer e-mail a besoin d'un serveur SMTP, des adresses de l'expéditeur et du destinataire de l'e-mail, d'un objet et d'un contenu.

Le tableau suivant montre les liaisons des paramètres d'entrée et de sortie que l'élément Tâche inscriptible Envoyer e-mail exige.

Tableau 1-60. Liaisons de l'élément Tâche inscriptible Envoyer e-mail

Nom du paramètre	Type de liaison	Créer une liaison avec le paramètre existant ou en créer un nouveau ?	Valeurs de liaison
vm	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[in-parameter]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>machine virtuelle à allumer.</b></li> </ul>
toAddress	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : toAddress</li> <li>■ Paramètre source : toAddress[in-parameter]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>adresse électronique de la personne à informer du résultat de ce workflow</b></li> </ul>
body	IN	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : body</li> <li>■ Paramètre source : body[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>corps de l'e-mail</b></li> </ul>
smtpHost	IN	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : smtpHost</li> <li>■ Paramètre source : smtpHost[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>serveur SMTP pour l'envoi de l'e-mail</b></li> </ul>
fromAddress	IN	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : fromAddress</li> <li>■ Paramètre source : fromAddress[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>adresse électronique de l'expéditeur</b></li> </ul>
subject	IN	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : subject</li> <li>■ Paramètre source : subject[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>objet de l'e-mail</b></li> </ul>

L'élément Tâche inscriptible Envoyer e-mail exige la fonction en script suivante.

```
//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+"'");
```

## Définir les liaisons d'exception de l'exemple de workflow simple

Vous pouvez définir les liaisons d'exception dans l'onglet **Schéma** de l'éditeur de workflow. Les liaisons d'exception définissent la façon dont les éléments traitent les erreurs.

Les éléments suivants du workflow renvoient les exceptions : startVM, vim3WaitTaskEnd, Send Email et vim3WaitToolsStarted.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- [Définir les paramètres de l'exemple de workflow simple.](#)
- [Définir les liaisons de décision de l'exemple de workflow simple.](#)
- [Relier les éléments d'action de l'exemple de workflow simple.](#)
- [Relier les éléments Tâche inscriptible de l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

- 1 Dans l'onglet **Schéma**, cliquez sur l'icône **Modifier** (✎) de l'élément d'action **startVM**.
- 2 Cliquez sur l'onglet **Exception**.
- 3 Cliquez sur le bouton **Non définie**.
- 4 Sélectionnez **errorCode** dans la liste proposée.
- 5 Répétez les étapes précédentes pour définir la liaison d'exception vers **errorCode** pour **vim3WaitTaskEnd** et **vim3WaitToolsStarted**.
- 6 Cliquez sur l'icône **Modifier** (✎) de l'élément Tâche inscriptible **Send Email**.
- 7 Cliquez sur l'onglet **Exception**.
- 8 Cliquez sur le bouton **Non définie**.
- 9 Sélectionnez **emailErrorCode** dans la liste proposée.

10 Cliquez sur **Enregistrer** en bas de l'onglet **Schéma** de l'éditeur de workflow.

Vous avez défini la liaison d'exception pour les éléments qui renvoient des exceptions.

### Étape suivante

Vous devez maintenant définir les propriétés de lecture et d'écriture des attributs et paramètres.

## Définir les propriétés en lecture seule des attributs de l'exemple de workflow simple

Vous pouvez définir si les paramètres et attributs sont des constantes en lecture seule ou des variables sur lesquelles vous disposez de droits d'écriture. Vous avez aussi la possibilité de fixer des limitations aux valeurs que les utilisateurs peuvent renseigner dans les paramètres d'entrée.

Définir certains paramètres en lecture seule permet à d'autres développeurs d'adapter le workflow ou de le modifier sans endommager la fonction centrale du workflow.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- [Définir les paramètres de l'exemple de workflow simple.](#)
- [Définir les liaisons de décision de l'exemple de workflow simple.](#)
- [Relier les éléments d'action de l'exemple de workflow simple.](#)
- [Relier les éléments Tâche inscriptible de l'exemple de workflow simple.](#)
- [Définir les liaisons d'exception de l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

1 Cliquez sur l'onglet **Général** dans la partie supérieure de l'éditeur de workflow.

Sous **Attributs** figure une liste de tous les attributs définis avec des cases à cocher en regard de chacun d'entre eux. En cochant ces cases, vous définissez les attributs en lecture seule.

2 Cochez les cases pour faire des attributs qui suivent des constantes en lecture seule :

- progression
- pollRate
- délai d'expiration
- smtpHost
- fromAddress
- objet de l'e-mail



Vous avez donc défini lesquels des attributs du workflow sont des constantes et lesquels sont des variables.

### Étape suivante

Définissez les propriétés de paramètre et placez des contraintes sur les valeurs possibles pour ce paramètre.

## Définir les propriétés du paramètre de l'exemple de workflow simple


Vous avez la possibilité de définir les propriétés du paramètre dans l'éditeur de workflow. Définir les propriétés du paramètre a une incidence sur le comportement du paramètre et fixe des contraintes aux valeurs possibles pour celui-ci.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- [Définir les paramètres de l'exemple de workflow simple.](#)
- [Définir les liaisons de décision de l'exemple de workflow simple.](#)
- [Relier les éléments d'action de l'exemple de workflow simple.](#)
- [Relier les éléments Tâche inscriptible de l'exemple de workflow simple.](#)
- [Définir les liaisons d'exception de l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

- 1 Cliquez sur l'onglet **Présentation** dans l'éditeur de workflow.  
Les deux paramètres d'entrée que vous avez définis pour ce workflow sont répertoriés.
- 2 Cliquez sur le paramètre **(VC:VirtualMachine)vm**.
- 3 Ajoutez une description dans l'onglet **Général** situé dans la moitié inférieure de l'écran.  
Vous pouvez saisir par exemple **Machine virtuelle à démarrer**.
- 4 Cliquez sur l'onglet **Propriétés** situé dans la moitié inférieure de l'écran.  
Sur cet onglet, vous pouvez définir les propriétés du paramètre **(VC:VirtualMachine)vm**.
- 5 Cliquez sur l'icône **Ajouter une propriété** (+).

- 6 Dans la liste des propriétés proposées, sélectionnez la propriété **Saisie obligatoire**, cliquez sur **OK** et définissez sa valeur sur **Oui**.

Lorsque vous activez cette propriété, les utilisateurs ne peuvent exécuter le workflow Démarrer la VM et envoyer un e-mail, sans avoir indiqué de machine virtuelle à démarrer.

- 7 Cliquez sur l'icône **Ajouter une propriété** (➤+).

- 8 Dans la liste des propriétés proposées, sélectionnez **Sélectionner la valeur sous forme de**, cliquez sur **OK** et sélectionnez **liste** dans la liste des valeurs possibles.

En définissant cette propriété, vous définissez la façon dont l'utilisateur sélectionne la valeur du paramètre d'entrée (VC:VirtualMachine)vm.

- 9 Cliquez sur le paramètre **(string)toAddress** dans la partie supérieure de l'onglet **Présentation**.

- 10 Ajoutez une description dans l'onglet **Description** situé dans la moitié inférieure de l'écran.

Vous pouvez saisir par exemple **Adresse électronique de la personne à informer**.

- 11 Cliquez sur l'onglet **Propriétés** de (string)toAddress et cliquez sur l'icône **Ajouter une propriété** (➤+).

- 12 Dans la liste des propriétés proposées, sélectionnez la propriété **Saisie obligatoire**, cliquez sur **OK** et définissez sa valeur sur **OK**.

- 13 Cliquez sur l'icône **Ajouter une propriété** (➤+).

- 14 Dans la liste des propriétés proposées, sélectionnez **Répondant à l'expression régulière** et cliquez sur **OK**.

Cette propriété vous permet de fixer des contraintes à ce que les utilisateurs peuvent renseigner en entrée.

- 15 Cliquez sur la zone de texte **Valeur** de **Répondant à l'expression régulière** et définissez les contraintes sur `[a-zA-Z0-9_%-+.]+@[a-zA-Z0-9-+.]+\.[a-zA-Z]{2,4}`.

Définir ces contraintes restreint la saisie des utilisateurs aux seuls caractères qui conviennent dans une adresse électronique. Si l'utilisateur essaie de saisir un autre caractère dans l'adresse électronique du destinataire lorsqu'il démarre le workflow, celui-ci ne se lance pas.

Vous avez donc rendu les deux paramètres obligatoires, défini la manière dont l'utilisateur peut sélectionner la machine virtuelle à démarrer et restreint les caractères autorisés pour la saisie de l'adresse électronique du destinataire.

### Étape suivante

Il vous faut créer l'agencement, ou la présentation, de la boîte de dialogue des paramètres d'entrée dans laquelle les utilisateurs renseignent les valeurs de paramètre d'entrée du workflow lorsqu'ils l'exécutent.

## Définir l'agencement de la boîte de dialogue des paramètres d'entrée d'un exemple de workflow simple

L'agencement ou la présentation de la boîte de dialogue des paramètres d'entrée se définit dans l'éditeur de workflow. La boîte de dialogue des paramètres d'entrée s'ouvre lorsque les utilisateurs exécutent un workflow qui a besoin d'une saisie de paramètres pour s'exécuter.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- [Définir les paramètres de l'exemple de workflow simple.](#)
- [Définir les liaisons de décision de l'exemple de workflow simple.](#)
- [Relier les éléments d'action de l'exemple de workflow simple.](#)
- [Relier les éléments Tâche inscriptible de l'exemple de workflow simple.](#)
- [Définir les liaisons d'exception de l'exemple de workflow simple.](#)
- [Définir les propriétés en lecture seule des attributs de l'exemple de workflow simple.](#)
- [Définir les propriétés du paramètre de l'exemple de workflow simple.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

- 1 Cliquez sur l'onglet **Présentation** de l'éditeur de workflow.
- 2 Cliquez avec le bouton droit sur le nœud **Présentation** dans la liste hiérarchique de présentation et sélectionnez **Créer un groupe d'affichage**  
  
Un nœud **Nouvelle étape** et un sous-nœud **Nouveau groupe** apparaissent alors sous le nœud **Présentation**.
- 3 Cliquez avec le bouton droit sur **Nouvelle étape** et sélectionnez **Supprimer**.  
  
Ce workflow ne possédant que deux paramètres, vous n'avez pas besoin d'une multiplicité de couches de sections d'affichage dans la boîte de dialogue des paramètres d'entrée.
- 4 Double-cliquez sur **Nouveau groupe** pour personnaliser le nom du groupe, puis appuyez sur Entrée.  
  
Vous pouvez, par exemple, nommer le groupe d'affichage **Machine virtuelle**.  
  
Le texte que vous saisissez ici apparaîtra en titre dans la boîte de dialogue des paramètres d'entrée lorsque les utilisateurs lanceront le workflow.
- 5 Dans la zone de texte **Description** de l'onglet **Général** au bas de l'onglet **Présentation**, renseignez une description pour le nouveau groupe d'affichage.  
  
Vous pouvez saisir par exemple **Sélectionner la machine virtuelle à démarrer**.

Le texte que vous saisissez ici apparaîtra comme invite dans la boîte de dialogue des paramètres d'entrée lorsque les utilisateurs lanceront le workflow.

- 6 Glissez-déposez le paramètre **(VC:VirtualMachine)vm** sous le groupe d'affichage **Machine virtuelle**.

Dans la boîte de dialogue des paramètres d'entrée, une zone de texte dans laquelle l'utilisateur devra saisir le nom de la machine virtuelle apparaîtra alors sous le titre Machine virtuelle.

- 7 Répétez les étapes précédentes pour créer un groupe d'affichage pour le paramètre **toAddress** en définissant les propriétés suivantes :
  - a Créez un groupe d'affichage que vous nommerez **Adresse électronique du destinataire**.
  - b Ajoutez-y une description, par exemple :  
**Saisir l'adresse électronique de la personne pour lui signaler quand cette machine virtuelle est allumée.**
  - c Glissez-déposez le paramètre **toAddress** sous le groupe d'affichage **Adresse électronique du destinataire**.

Vous avez donc configuré l'agencement de la boîte de dialogue des paramètres d'entrée qui apparaît au moment où les utilisateurs lancent le workflow.

### Étape suivante

Vous avez terminé le développement de l'exemple de workflow simple. Vous pouvez maintenant valider et exécuter le workflow.

## Valider et exécuter l'exemple de workflow simple

Une fois un workflow créé, vous avez la possibilité de le valider afin d'y déceler toute erreur éventuelle. Si le workflow est exempt d'erreurs, vous pouvez l'exécuter.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow simple.](#)
- [Créer le schéma de l'exemple de workflow simple.](#)
- [Définir les paramètres de l'exemple de workflow simple.](#)
- [Définir les liaisons de décision de l'exemple de workflow simple.](#)
- [Relier les éléments d'action de l'exemple de workflow simple.](#)
- [Relier les éléments Tâche inscriptible de l'exemple de workflow simple.](#)
- [Définir les liaisons d'exception de l'exemple de workflow simple.](#)
- [Définir les propriétés en lecture seule des attributs de l'exemple de workflow simple.](#)
- [Définir les propriétés du paramètre de l'exemple de workflow simple.](#)

- Définir l'agencement de la boîte de dialogue des paramètres d'entrée d'un exemple de workflow simple.
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

#### Procédure

- 1 Cliquez sur **Valider** dans l'onglet **Schéma** de l'éditeur de workflow.

L'outil de validation décèle toute erreur dans la définition du workflow.

- 2 Une fois les éventuelles erreurs éliminées, cliquez sur **Enregistrer et fermer** au bas de l'éditeur de workflow.

Vous retournez alors au client Orchestrator.

- 3 Cliquez sur la vue **Workflows**.

- 4 Sélectionnez **Exemples de workflow > Démarrer la VM et envoyer e-mail** dans la liste hiérarchique des workflows.

- 5 Cliquez avec le bouton droit sur le workflow **Démarrer la VM et envoyer e-mail** et sélectionnez **Démarrer le workflow**.

La boîte de dialogue des paramètres d'entrée s'ouvre qui vous invite à indiquer une machine virtuelle à démarrer et une adresse électronique à laquelle envoyer les notifications.

- 6 Sélectionnez une machine virtuelle à démarrer dans l'inventaire de vCenter Server.

- 7 Saisissez une adresse électronique à laquelle envoyer les notifications.

- 8 Cliquez sur **Envoyer** pour démarrer les workflow.

Un jeton de workflow apparaît sous le workflow Démarrer la VM et envoyer e-mail.

- 9 Cliquez sur le jeton du workflow pour suivre la progression du workflow dans son exécution.

Si l'exécution du workflow aboutit, la machine virtuelle que vous avez sélectionnée se trouve à l'état sous tension et le destinataire de l'e-mail que vous avez défini reçoit un e-mail de confirmation.

#### Étape suivante

Vous pouvez générer un document dans lequel vous pourrez consulter les informations relatives au workflow. Consultez [Générer la documentation sur les workflows](#).

## Développer un workflow complexe

Le développement d'un exemple de workflow complexe permet de démontrer les étapes les plus courantes du processus de développement des workflows, ainsi que des scénarios plus avancés tels que la création de décisions et de boucles personnalisées.

Dans l'exercice sur les workflows complexes, vous développez un workflow qui capture un snapshot de toutes les machines virtuelles présentes dans un pool de ressources donné. Le workflow que vous créez effectue les tâches suivantes :

- 1 Dirige l'utilisateur vers le pool de ressources contenant les machines virtuelles pour la capture de snapshots.
- 2 Détermine si le pool de ressources contient des machines virtuelles actives.
- 3 Détermine le nombre de machines virtuelles actives que contient la ressource.
- 4 Vérifie si une machine virtuelle s'exécutant dans le pool répond aux critères spécifiques de capture de snapshots.
- 5 Capture le snapshot de la machine virtuelle.
- 6 Détermine si d'autres machines virtuelles se trouvent dans le pool pour la capture de snapshots.
- 7 Répète le processus de vérification et de capture jusqu'à ce que le workflow ait pris des snapshots de toutes les machines virtuelles éligibles du pool de ressources.

Le fichier ZIP des exemples Orchestrator que vous pouvez télécharger dans la page d'accueil de la documentation Orchestrator contient une version complète du workflow Créer des snapshots de toutes les machines virtuelles dans un pool de ressources.

### Conditions préalables

Avant de commencer à développer ce workflow complexe, effectuez les exercices dans [Développer un exemple de workflow simple](#). Les procédures de développement d'un workflow complexe renseignent sur les grandes étapes du processus de développement mais ne sont pas aussi détaillées que les exercices relatifs aux workflows simples.

### Procédure

#### 1 Créer l'exemple de workflow complexe

Vous devez commencer le processus de développement du workflow en créant le workflow dans le client Orchestrator.

#### 2 Créer une action personnalisée pour l'exemple de workflow complexe

L'élément inscriptible Check VM appelle une action qui n'existe pas dans l'API Orchestrator. Vous devez créer l'action getVMDiskModes.

#### 3 Créer le schéma de l'exemple de workflow complexe

Vous pouvez créer un schéma de workflow dans l'éditeur de workflow. Le schéma de workflow contient les éléments exécutés par le workflow et détermine le déroulement logique du workflow.

#### 4 (Facultatif) Créer les zones de l'exemple de workflow complexe

Si vous le souhaitez, vous pouvez mettre en surbrillance les différentes zones du workflow en ajoutant des notes de workflow. La création de différentes zones de workflow facilite la lecture et la compréhension des schémas de workflow complexes.

## 5 Définir les paramètres de l'exemple de workflow complexe

Vous pouvez définir les paramètres du workflow dans l'éditeur de workflow. Les paramètres d'entrée fournissent les données que le workflow doit traiter. Les paramètres de sortie correspondent aux données que le workflow renvoie à la fin de son exécution.

## 6 Définir les liaisons de l'exemple de workflow complexe

Vous pouvez relier les éléments d'un workflow entre eux dans l'éditeur de workflow. Les liaisons définissent l'enchaînement de données du workflow. Vous établirez également une liaison entre les éléments de tâche inscriptible et leurs fonctions JavaScript.

## 7 Définir les propriétés d'attribut d'un exemple de workflow complexe

Vous définirez les propriétés de l'attribut dans l'onglet **Général** de l'éditeur de workflow.

## 8 Créer la mise en forme des paramètres d'entrée de l'exemple de workflow complexe

Vous pouvez créer la mise en forme, ou présentation, de la boîte de dialogue des paramètres d'entrée dans l'onglet **Présentation** de l'éditeur de workflow. La boîte de dialogue des paramètres d'entrée s'ouvre lorsque les utilisateurs exécutent un workflow. Il s'agit du moyen par lequel les utilisateurs saisissent les paramètres d'entrée avec lesquels le workflow s'exécute.

## 9 Valider et exécuter l'exemple de workflow complexe

Une fois un workflow créé, vous avez la possibilité de le valider afin d'y déceler toute erreur éventuelle. Si le workflow est exempt d'erreurs, vous pouvez l'exécuter.

# Créer l'exemple de workflow complexe

Vous devez commencer le processus de développement du workflow en créant le workflow dans le client Orchestrator.

Pour plus d'informations sur l'installation et la configuration de vCenter Server, consultez la documentation *Installation et configuration de vSphere*. Pour plus d'informations sur la configuration d'Orchestrator, reportez-vous à *Installation et configuration d'VMware vRealize Orchestrator*.

### Conditions préalables

Vérifiez que les composants suivants sont installés et configurés sur le système.

- vCenter Server, qui contrôle un pool de ressources comportant certaines machines virtuelles
- Le dossier **Exemples de workflows** de la liste hiérarchique créé dans [Créer l'exemple de workflow simple](#).

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Sélectionnez **Workflows > Exemples de workflows**.
- 3 Cliquez avec le bouton droit sur le dossier **Exemples de workflows**, puis sélectionnez **Nouveau workflow**.

- 4 Nommez le nouveau workflow

**Créer des snapshots de toutes les machines virtuelles dans un pool de ressources**, puis cliquez sur **OK**.

L'éditeur de workflow s'ouvre.

- 5 Dans l'onglet **Général** de l'éditeur de workflow, cliquez sur les chiffres du numéro de version pour insérer le numéro de version.

Lors de la première création du workflow, définissez la version sur **0.0.1**.

- 6 Cliquez sur la valeur **Comportement de redémarrage du serveur** pour définir si le workflow reprendra en cas de redémarrage du serveur.

- 7 Dans la zone de texte **Description**, saisissez la description des fonctions du workflow.

- 8 Cliquez sur **Enregistrer** en bas de l'onglet **Général**.

Vous avez créé le workflow **Créer des snapshots de toutes les machines virtuelles dans un pool de ressources**.

#### Étape suivante

Vous devez maintenant créer une action personnalisée.

## Créer une action personnalisée pour l'exemple de workflow complexe

L'élément inscriptible `Check VM` appelle une action qui n'existe pas dans l'API Orchestrator. Vous devez créer l'action `getVMDiskModes`.

Pour en savoir plus sur la création d'actions, reportez-vous à [Chapitre 3 Développement d'actions](#).

#### Conditions préalables

Créez le workflow **Créer des snapshots de toutes les machines virtuelles dans un pool de ressources**. Reportez-vous à [Créer l'exemple de workflow complexe](#).

#### Procédure

- 1 Fermez l'éditeur de workflow en cliquant sur **Enregistrer et fermer**.
- 2 Cliquez sur la vue **Actions** dans le client Orchestrator.
- 3 Cliquez avec le bouton droit sur la racine de la liste hiérarchique des actions, puis sélectionnez **Nouveau module**.
- 4 Nommez le nouveau module **com.vmware.example**.
- 5 Cliquez avec le bouton droit sur le module **com.vmware.example**, puis sélectionnez **Ajouter une action**.
- 6 Créez une action appelée `getVMDiskModes`.
- 7 Insérez le numéro de version dans l'onglet **Général** de l'éditeur d'actions en cliquant sur les chiffres de la version.



## 8 Ajoutez la description suivante de l'action dans l'onglet **Général**.

```
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 Cliquez sur l'onglet **Script**.

## 10 Cliquez avec le bouton droit dans le volet supérieur de l'onglet **Script**, puis sélectionnez **Ajouter un paramètre** pour créer le paramètre d'entrée suivant.

- Nom : vm
- Type : VC:VirtualMachine
- Description : **Machine virtuelle pour laquelle les modes de disques sont renvoyés**

## 11 Ajoutez les scripts suivants en bas de l'onglet **Script**.

Le code suivant renvoie un groupe de modes de disques pour les disques de la machine virtuelle.

```
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 Cliquez sur **Enregistrer et fermer** pour quitter la palette **Actions**.

Vous avez défini l'action personnalisée nécessaire au workflow Créer des snapshots de toutes les machines virtuelles dans un pool de ressources.

### Étape suivante

Créez le schéma de workflow.

## Créer le schéma de l'exemple de workflow complexe

Vous pouvez créer un schéma de workflow dans l'éditeur de workflow. Le schéma de workflow contient les éléments exécutés par le workflow et détermine le déroulement logique du workflow.

## Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow complexe.](#)
- [Créer une action personnalisée pour l'exemple de workflow complexe.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

## Procédure

- 1 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.
- 2 Ajoutez les éléments suivants au schéma de workflow.

Type d'élément	Nom de l'élément	Emplacement dans le schéma
Tâche inscriptible	<b>Initializing</b>	Sous l'élément Start
Décision	<b>VMs to Process?</b>	Sous l'élément Tâche inscriptible Initializing
Tâche inscriptible	<b>Pool Has No VMs</b>	Sous l'élément Décision personnalisée VMs to Process?, relié avec une flèche rouge
Décision personnalisée	<b>Remaining VMs?</b>	À droite de l'élément Décision personnalisée VMs to Process?, relié avec une flèche verte
Action	getVMDiskModes	À droite de l'élément Décision personnalisée Remaining VMs?, relié avec une flèche verte
Décision personnalisée	<b>Create Snapshot?</b>	À droite de l'élément d'action getVMDiskModes, relié avec une flèche bleue
Workflow	Create a snapshot	Au-dessus de l'élément Décision personnalisée Create Snapshot?, relié avec une flèche verte
Tâche inscriptible	<b>VM Snapshots</b>	À gauche du workflow Create a snapshot, relié avec une flèche bleue
Tâche inscriptible	<b>Increment</b>	À gauche de l'élément Tâche inscriptible VM Snapshots, relié avec une flèche bleue
Tâche inscriptible	<b>Set Output</b>	À droite de l'élément Tâche inscriptible Pool Has No VMs, relié avec une flèche bleue

- 3 Ajoutez un élément Tâche inscriptible Log Exception.
  - a Créez un lien de gestion des exceptions à partir du workflow Créer un snapshot dans un élément End.
  - b Faites glisser un élément Tâche inscriptible vers la flèche discontinue rouge qui relie le workflow Créer un snapshot à un élément End.
  - c Double-cliquez sur l'élément Tâche inscriptible et renommez-le par **Exception de journal**.
  - d Déplacez l'élément Tâche inscriptible Log Exception au-dessus de l'élément Tâche inscriptible VM Snapshots.
- 4 Annulez le lien de tous les éléments End à l'exception de l'élément End situé à droite de l'élément Tâche inscriptible Set Output.

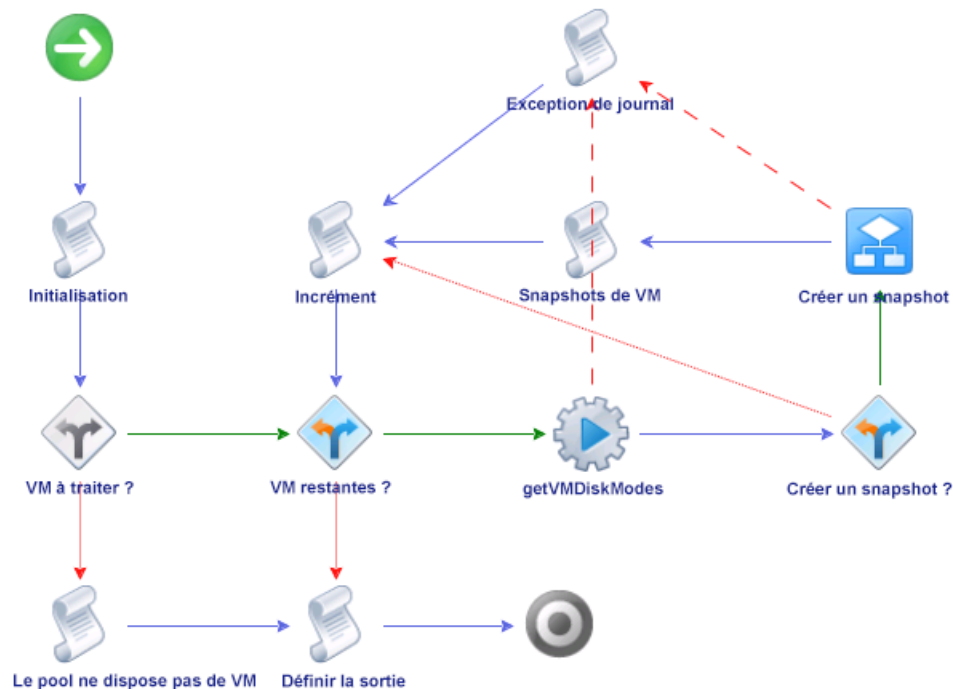
**5** Reliez les éléments restants comme indiqué dans le tableau suivant.

Élément	Relier à	Type de flèche	Description
Élément d'action getVMDiskModes	Élément Tâche inscriptible Log Exception	Discontinue rouge	Gestion des exceptions
Élément Décision personnalisée Create Snapshot?	Élément Tâche inscriptible Increment	Rouge	Résultat False
Élément Tâche inscriptible Log Exception	Élément Tâche inscriptible Increment	Bleue	Progression normale du workflow
Élément Tâche inscriptible Increment	Élément Décision personnalisée Remaining VMs?	Bleue	Progression normale du workflow
Élément Décision personnalisée Remaining VMs?	Élément Tâche inscriptible Set Output	Rouge	Résultat False

**6** Cliquez sur **Enregistrer** en bas de l'onglet **Schéma**.

La figure suivante montre à quoi devraient ressembler les éléments reliés du workflow Créer des snapshots de toutes les machines virtuelles dans un pool de ressources.

Figure 1-12. Relier l'exemple de workflow Créer des snapshots de toutes les machines virtuelles dans un pool de ressources



## Étape suivante

Si vous le souhaitez, vous pouvez définir des zones de workflow à l'aide des notes de workflow.

## Créer les zones de l'exemple de workflow complexe

Si vous le souhaitez, vous pouvez mettre en surbrillance les différentes zones du workflow en ajoutant des notes de workflow. La création de différentes zones de workflow facilite la lecture et la compréhension des schémas de workflow complexes.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow complexe.](#)
- [Créer le schéma de l'exemple de workflow complexe.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

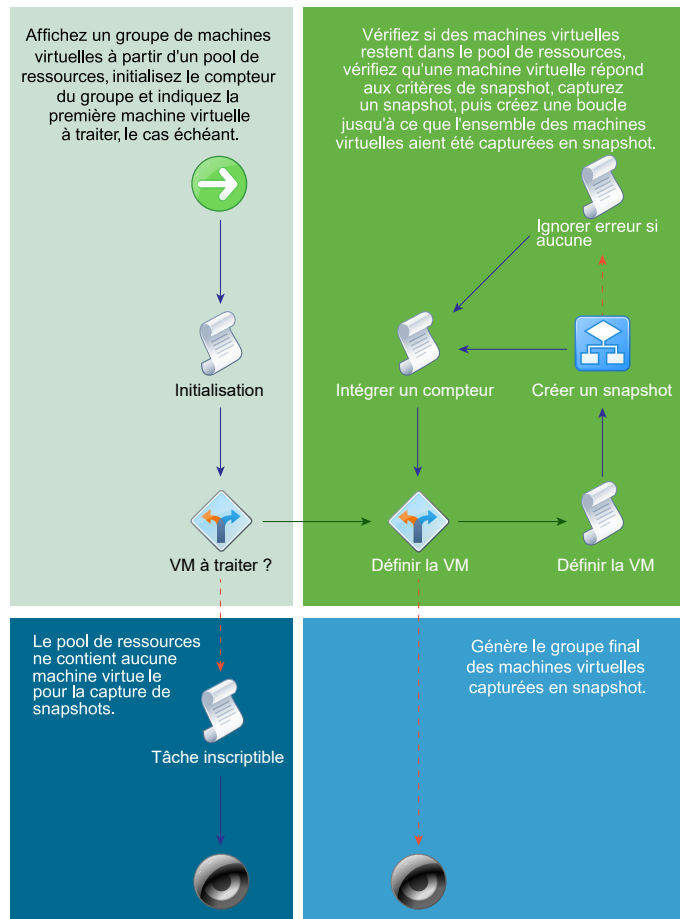
- 1 Créez les zones de workflow suivantes à l'aide des notes de workflow.

Éléments de la zone	Description
Démarrer élément ; Initialiser la tâche inscriptible ; VM à traiter ? Décision personnalisée	Affichez un groupe de machines virtuelles à partir d'un pool de ressources, initialisez le compteur du groupe et indiquez la première machine virtuelle à traiter, le cas échéant.
Aucune tâche inscriptible de VM dans le pool.	Le pool de ressources ne contient aucune machine virtuelle pour la capture de snapshots.
VM restantes ? décision personnalisée ; action getVMDisksModes, Créer un snapshot ? décision ; Créer un workflow de snapshots ; Tâche inscriptible des snapshots de VM ; Incrémenter une tâche inscriptible ; Tâche inscriptible Exception de journal	Vérifiez si des machines virtuelles restent dans le pool de ressources, vérifiez qu'une machine virtuelle répond aux critères de snapshot, capturez un snapshot, puis créez une boucle jusqu'à ce que l'ensemble des machines virtuelles aient été capturées en snapshot.
Tâche inscriptible Définir la sortie ; Élément Fin	Génère le groupe final des machines virtuelles capturées en snapshot.

- 2 Sélectionnez une note de workflow et appuyez sur Ctrl+E pour sélectionner la couleur de l'arrière-plan.
- 3 Cliquez sur **Enregistrer** en bas de l'onglet **Schéma** de l'éditeur de workflow.

Vos zones de workflow doivent ressembler au diagramme suivant.

**Figure 1-13. Diagramme de schéma pour l'exemple de workflow Créer des snapshots de toutes les machines virtuelles dans un pool de ressources**



### Étape suivante

Vous devez définir les paramètres d'entrée et de sortie du workflow.

## Définir les paramètres de l'exemple de workflow complexe

Vous pouvez définir les paramètres du workflow dans l'éditeur de workflow. Les paramètres d'entrée fournissent les données que le workflow doit traiter. Les paramètres de sortie correspondent aux données que le workflow renvoie à la fin de son exécution.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow complexe.](#)
- [Créer le schéma de l'exemple de workflow complexe.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

**Procédure**

- 1 Cliquez sur l'onglet **Entrées** dans l'éditeur de workflow.
- 2 Définissez le paramètre d'entrée suivant.
  - Nom : resourcePool
  - Type : VC:ResourcePool
  - Description :  
**Pool de ressources contenant les machines virtuelles pour la capture de snapshots.**
- 3 Cliquez sur l'onglet **Sorties** dans l'éditeur de workflow.
- 4 Définissez les paramètres de sortie suivants.
  - Nom : snapshotVmArrayOut
  - Type : Array/VC:VirtualMachine
  - Description : **Groupe de machines virtuelles ayant fait l'objet de snapshots.**

Vous avez défini les paramètres d'entrée et de sortie du workflow.

**Étape suivante**

Vous devez maintenant définir les liaisons entre les paramètres des éléments.

**Définir les liaisons de l'exemple de workflow complexe**

Vous pouvez relier les éléments d'un workflow entre eux dans l'éditeur de workflow. Les liaisons définissent l'enchaînement de données du workflow. Vous établirez également une liaison entre les éléments de tâche inscriptible et leurs fonctions JavaScript.

**Conditions préalables**

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow complexe.](#)
- [Créer le schéma de l'exemple de workflow complexe](#)
- [Définir les paramètres de l'exemple de workflow complexe](#)
- Passez en revue les liaisons que vous devez définir. Reportez-vous à [Exemples de liaisons de workflows complexes.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

**Procédure**

- 1 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.
- 2 Définissez les liaisons.
- 3 Cliquez sur **Enregistrer** en bas de l'onglet **Schéma**.

Tous les paramètres d'entrée et de sortie des éléments sont reliés aux types et valeurs de paramètres appropriés.

### Étape suivante

Définissez maintenant les propriétés des attributs.

## Exemples de liaisons de workflows complexes

Les liaisons définissent la manière dont les éléments d'action de l'exemple de workflow simple traitent les paramètres d'entrée et de sortie.

Le workflow Créer des snapshots de toutes les machines virtuelles dans un pool de ressources nécessite les liaisons suivantes de paramètres d'entrée et de sortie. Vous pouvez également définir les fonctions JavaScript des éléments Tâche inscriptible.

Si vous effectuez des liaisons avec des paramètres existants, la liaison hérite des valeurs de type et de description des paramètres d'origine.

### Tâche inscriptible Initialisation

L'élément Tâche inscriptible Initialisation permet d'initialiser les attributs du workflow. La table suivante montre les liaisons des paramètres d'entrée et de sortie nécessaires à l'élément Tâche inscriptible Initialisation.

**Tableau 1-61. Liaisons de l'élément Tâche inscriptible Initialisation**

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
resourcePool	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : resourcePool</li> <li>■ Paramètre source : resourcePool[in-parameter]</li> <li>■ Type : VC:ResourcePool</li> <li>■ Description : <b>Pool de ressources contenant les machines virtuelles devant faire l'objet de snapshots</b></li> </ul>
allVMs	SORTIE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : allVMs</li> <li>■ Paramètre source : allVMs[attribute]</li> <li>■ Type : Array/VC:VirtualMachine</li> <li>■ Description : <b>Machines virtuelles présentes dans le pool de ressources.</b></li> </ul>

Tableau 1-61. Liaisons de l'élément Tâche inscriptible Initialisation (suite)

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
numberOfVms	SORTIE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : numberOfVms</li> <li>■ Paramètre source : numberOfVms[attribute]</li> <li>■ Type : nombre</li> <li>■ Description : <b>Nombre de machines virtuelles trouvées dans le resourcePool</b></li> </ul>
vmCounter	SORTIE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : vmCounter</li> <li>■ Paramètre source : vmCounter[attribute]</li> <li>■ Type : nombre</li> <li>■ Description : <b>Compteur de machines virtuelles dans le groupe</b></li> </ul>
vm	SORTIE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[attribute]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>Machine virtuelle active avec snapshot capturé</b></li> </ul>
snapshotVmArray	SORTIE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : snapshotVmArray</li> <li>■ Paramètre source : snapshotVmArray[attribute]</li> <li>■ Type : Array/VC:VirtualMachine</li> <li>■ Description : <b>Groupe de machines virtuelles dont les snapshots ont été capturés</b></li> </ul>

L'élément Tâche inscriptible Initialiser exécute la fonction en script suivante.

```
//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();
```



## Élément Décision VM à traiter ?

L'élément Décision VM à traiter ? détermine si des machines virtuelles devant faire l'objet de snapshots se trouvent dans le pool de ressources. Le tableau suivant montre les liaisons dont l'élément Décision VM à traiter ? a besoin.

**Tableau 1-62. Liaisons de l'élément Décision VM à traiter ?**

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
numberOfVMs	Décision	Relier	<ul style="list-style-type: none"> <li>■ Paramètre source : numberOfVMs[attribute]</li> <li>■ Instruction de décision : supérieur à</li> <li>■ Valeur : 0.0</li> <li>■ Description : <b>Nombre de machines virtuelles trouvées dans le resourcePool</b></li> </ul>

## Élément Aucune tâche inscriptible de VM dans le pool

L'élément Aucune tâche inscriptible de VM dans le pool inscrit dans la base de données Orchestrator que le pool de ressources ne dispose pas de machine virtuelle éligible. Le tableau suivant montre les liaisons dont l'élément Aucune tâche inscriptible de VM dans le pool a besoin.

**Tableau 1-63. Liaisons de l'élément Aucune tâche inscriptible de VM dans le pool**

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
resourcePool	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : resourcePool</li> <li>■ Paramètre source : resourcePool[in-parameter]</li> <li>■ Type : VC:ResourcePool</li> <li>■ Description : <b>Pool de ressources contenant les machines virtuelles pour la capture de snapshots.</b></li> </ul>

L'élément Aucune tâche inscriptible de VM dans le pool exécute la fonction en script suivante.

```
//Writes the following event in the Orchestrator database
Server.warn("The specified ResourcePool "+resourcePool.name+" does not contain any VMs.");
```

## Élément Décision personnalisée VM restantes ?

L'élément Décision personnalisée VM restantes ? détermine si des machines virtuelles à capturer en snapshot demeurent encore dans le pool de ressources. Le tableau suivant montre les liaisons dont l'élément Décision personnalisée VM restantes ? a besoin.

**Tableau 1-64. Liaisons de l'élément Décision personnalisée VM restantes ?**

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
numberOfVMs	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre source : numberOfVMs[attribute]</li> <li>■ Instruction de décision : supérieur à</li> <li>■ Valeur : 0.0</li> <li>■ Description : <b>Nombre de machines virtuelles trouvées dans le resourcePool</b></li> </ul>
vmCounter	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vmCounter</li> <li>■ Paramètre source : vmCounter[attribute]</li> <li>■ Type : nombre</li> <li>■ Description : <b>Compteur de machines virtuelles dans le groupe</b></li> </ul>

L'élément Décision personnalisée VM restantes ? exécute la fonction en script suivante.

```
//Checks if the workflow has reached the end of the array of VMs
if (vmCounter < numberOfVMs) {
    return true;
} else {
    return false;
}
```

### Élément d'action getVMDisksModes

L'élément d'action getVMDisksModes récupère les modes des disques s'exécutant dans une machine virtuelle. Le tableau suivant montre les liaisons dont l'élément d'action getVMDisksModes a besoin.

Tableau 1-65. Liaisons de l'élément d'action getVMDisksModes

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
vm	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[attribute]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>Machine virtuelle active avec snapshot capturé</b></li> </ul>
actionResult	SORTIE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : actionResult</li> <li>■ Paramètre source : vmDisksModes[attribute]</li> <li>■ Type : Groupe/Chaîne</li> <li>■ Description : <b>Modes des disques actifs de la machine virtuelle</b></li> </ul>
errorCode	Exception	Créer	Paramètre local : errorCode

### Élément Décision personnalisée Créer un snapshot ?

L'élément Décision personnalisée Créer un snapshot ? détermine s'il est nécessaire de capturer des snapshots des machines virtuelles en fonction des modes des disques des machines virtuelles. Le tableau suivant montre les liaisons dont l'élément Décision personnalisée Créer un snapshot ? a besoin.

Tableau 1-66. Liaisons de l'élément Décision Créer un snapshot ?

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
vmDisksMode	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vmDisksMode</li> <li>■ Paramètre source : vmDisksMode[attribute]</li> <li>■ Type : Groupe/Chaîne</li> <li>■ Description : <b>Modes des disques actifs de la machine virtuelle</b></li> </ul>
vm	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[attribute]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>Machine virtuelle active avec snapshot capturé</b></li> </ul>

L'élément Décision personnalisée Créer un snapshot ? exécute la fonction en script suivante.

```
//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;
}
```

### Élément de workflow Créer un snapshot

L'élément de workflow Créer un snapshot permet de capturer les snapshots des machines virtuelles. Le tableau suivant montre les liaisons dont l'élément de workflow Créer un snapshot a besoin.

**Tableau 1-67. Liaisons de l'élément de workflow Créer un snapshot**

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
vm	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[attribute]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>Machine virtuelle active qui doit faire l'objet d'un snapshot.</b></li> </ul>
name	ENTRÉE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : name</li> <li>■ Paramètre source : snapshotName[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>Nom de ce snapshot. Le nom ne doit pas nécessairement être unique pour cette machine virtuelle.</b></li> </ul>

Tableau 1-67. Liaisons de l'élément de workflow Créer un snapshot (suite)

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
description	ENTRÉE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : description</li> <li>■ Paramètre source : snapshotDescription[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description : <b>Description de ce snapshot.</b></li> </ul>
memory	ENTRÉE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : memory</li> <li>■ Paramètre source : snapshotMemory[attribute]</li> <li>■ Type : booléen</li> <li>■ Valeur : non</li> <li>■ Description : <b>Si TRUE, un vidage de l'état interne de la machine virtuelle (vidage mémoire) est inclus dans le snapshot.</b></li> </ul>
quiesce	ENTRÉE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : quiesce</li> <li>■ Paramètre source : snapshotQuiesce[attribute]</li> <li>■ Type : booléen</li> <li>■ Valeur : oui</li> <li>■ Description : <b>Si TRUE et que la machine virtuelle est sous tension lorsque le snapshot est capturé, les VMware Tools sont utilisés pour mettre en veille le système de fichiers de la machine virtuelle.</b></li> </ul>
snapshot	SORTIE	Créer	<ul style="list-style-type: none"> <li>■ Paramètre local : snapshot</li> <li>■ Paramètre source : NULL</li> <li>■ Type : VC:VirtualMachineSnapshot</li> <li>■ Description : <b>Snapshot capturé.</b></li> </ul>
errorCode	Exception	Créer	Paramètre local : errorCode

### Élément Tâche inscriptible Snapshots de VM

L'élément Tâche inscriptible Snapshots de VM ajoute les snapshots à un groupe. Le tableau suivant montre les liaisons dont l'élément Tâche inscriptible Snapshots de VM a besoin.

**Tableau 1-68. Liaisons de l'élément Tâche inscriptible Snapshots de VM**

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
vm	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[attribute]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>Machine virtuelle active qui doit faire l'objet d'un snapshot.</b></li> </ul>
snapshotVmArray	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : snapshotVmArray</li> <li>■ Paramètre source : snapshotVmArray[attribute]</li> <li>■ Type : Array/VC:VirtualMachine</li> <li>■ Description : <b>Groupe de machines virtuelles ayant fait l'objet de snapshots</b></li> </ul>
snapshotVmArray	SORTIE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : snapshotVmArray</li> <li>■ Paramètre source : snapshotVmArray[attribute]</li> <li>■ Type : Array/VC:VirtualMachine</li> <li>■ Description : <b>Groupe de machines virtuelles ayant fait l'objet de snapshots</b></li> </ul>

L'élément Tâche inscriptible Snapshots de VM exécute la fonction en script suivante.

```
//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);
```

### Élément Tâche inscriptible Incrémenter

L'élément Tâche inscriptible Incrémenter permet d'incrémenter le compteur qui dénombre le nombre de machines virtuelles dans le groupe. Le tableau suivant montre les liaisons dont l'élément Tâche inscriptible Incrémenter a besoin.

Tableau 1-69. Liaisons de l'élément Tâche inscriptible Incrémenter

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
vmCounter	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vmCounter</li> <li>■ Paramètre source : vmCounter[attribute]</li> <li>■ Type : nombre</li> <li>■ Description : <b>Compteur de machines virtuelles dans le groupe</b></li> </ul>
allVMs	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : allVMs</li> <li>■ Paramètre source : allVMs[attribute]</li> <li>■ Type : Array/VC:VirtualMachine</li> <li>■ Description : <b>Machines virtuelles présentes dans le pool de ressources.</b></li> </ul>
vmCounter	SORTIE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vmCounter</li> <li>■ Paramètre source : vmCounter[attribute]</li> <li>■ Type : nombre</li> <li>■ Description : <b>Compteur de machines virtuelles dans le groupe</b></li> </ul>
vm	SORTIE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[attribute]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>Machine virtuelle active avec snapshot capturé</b></li> </ul>

L'élément Tâche inscriptible Incrémenter exécute la fonction en script suivante.

```
//Increases the array VM counter
vmCounter++;
//Sets the next VM to be snapshot in the attribute vm
vm = allVMs[vmCounter];
```

### Élément Tâche inscriptible Exception de journal

L'élément Tâche inscriptible Exception de journal traite les exceptions du workflow et des éléments d'action. Le tableau suivant montre les liaisons dont l'élément Tâche inscriptible Exception de journal a besoin.

Tableau 1-70. Liaisons de l'élément Tâche inscriptible Exception de journal

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
vm	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : vm</li> <li>■ Paramètre source : vm[attribute]</li> <li>■ Type : VC:VirtualMachine</li> <li>■ Description : <b>Machine virtuelle active avec snapshot capturé</b></li> </ul>
errorCode	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : errorCode</li> <li>■ Paramètre source : errorCode[attribute]</li> <li>■ Type : chaîne</li> <li>■ Description: <b>Exception identifiée lors de la capture du snapshot d'une machine virtuelle</b></li> </ul>

L'élément Tâche inscriptible Exception de journal exécute la fonction en script suivante.

```
//Writes the following event in the Orchestrator database
Server.error("Couldn't snapshot the VM '"+vm.name+"', exception: "+errorCode);
```

### Élément Tâche inscriptible Définir la sortie

La tâche inscriptible Définir la sortie génère le paramètre de sortie du workflow qui contient le groupe de machines virtuelles ayant fait l'objet de snapshots. Le tableau suivant montre les liaisons dont l'élément Tâche inscriptible Définir la sortie a besoin.



Tableau 1-71. Liaisons de l'élément de tâche Définir la sortie

Nom du paramètre	Type de liaison	Relier à un paramètre existant ou créer un paramètre ?	Valeurs de liaison
snapshotVmArray	ENTRÉE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : snapshotVmArray</li> <li>■ Paramètre source : snapshotVmArray[attribute]</li> <li>■ Type : Array/VC:VirtualMachine</li> <li>■ Description : <b>Groupe de machines virtuelles ayant fait l'objet de snapshots</b></li> </ul>
snapshotVmArrayOut	SORTIE	Relier	<ul style="list-style-type: none"> <li>■ Paramètre local : snapshotVmArrayOut</li> <li>■ Paramètre source : snapshotVmArrayOut[out-parameter]</li> <li>■ Type : Array/VC:VirtualMachine</li> <li>■ Description : <b>Groupe de machines virtuelles ayant fait l'objet de snapshots</b></li> </ul>

L'élément Tâche inscriptible Définir la sortie exécute la fonction en script suivante.

```
//Passes the value of the internal attribute to a workflow output parameter
snapshotVmArrayOut = snapshotVmArray;
```

## Définir les propriétés d'attribut d'un exemple de workflow complexe

Vous définirez les propriétés de l'attribut dans l'onglet **Général** de l'éditeur de workflow.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow complexe.](#)
- [Créer le schéma de l'exemple de workflow complexe.](#)
- [Définir les liaisons de l'exemple de workflow complexe.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

- 1 Cliquez sur l'onglet **Général**.
- 2 Cochez la case Lecture seule des attributs suivants pour en faire des constantes en lecture seule :
  - snapshotName

- snapshotDescription
- snapshotMemory
- snapshotQuiesce

Vous avez donc défini lesquels des attributs du workflow sont des constantes et lesquels sont des variables.

### Étape suivante

Il vous faut créer la présentation du workflow, ce qui donnera lieu à l'agencement de la boîte de dialogue des paramètres d'entrée dans laquelle les utilisateurs renseignent les valeurs de paramètres d'entrée du workflow lorsqu'ils l'exécutent.

## Créer la mise en forme des paramètres d'entrée de l'exemple de workflow complexe

Vous pouvez créer la mise en forme, ou présentation, de la boîte de dialogue des paramètres d'entrée dans l'onglet **Présentation** de l'éditeur de workflow. La boîte de dialogue des paramètres d'entrée s'ouvre lorsque les utilisateurs exécutent un workflow. Il s'agit du moyen par lequel les utilisateurs saisissent les paramètres d'entrée avec lesquels le workflow s'exécute.

### Conditions préalables

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow complexe.](#)
- [Créer le schéma de l'exemple de workflow complexe.](#)
- [Définir les paramètres de l'exemple de workflow complexe.](#)
- [Définir les liaisons de l'exemple de workflow complexe.](#)
- [Définir les propriétés d'attribut d'un exemple de workflow complexe.](#)
- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

- 1 Cliquez sur l'onglet **Présentation** dans l'éditeur de workflow.

Le workflow Créer des snapshots de toutes les machines virtuelles dans un pool de ressources ne possède qu'un seul paramètre d'entrée ; la création de la présentation est donc rapide.

- 2 Cliquez avec le bouton droit sur le nœud **Présentation** dans la liste hiérarchique de présentation, puis sélectionnez **Créer un groupe d'affichage**.
- 3 Supprimez l'élément **Nouvelle étape** qui apparaît au-dessus de l'élément **Nouveau groupe**.
- 4 Double-cliquez sur l'élément **Nouveau groupe**, puis modifiez le nom du groupe par **Pool de ressources**.

- 5 Entrez une description du groupe d'affichage **Pool de ressources** dans la zone de texte **Description** de l'onglet **Général** situé au bas de l'onglet **Présentation**.

Par exemple,

**Entrez le nom du pool de ressources contenant les machines virtuelles pour la capture de snapshots.**

- 6 Cliquez sur le paramètre (VC:ResourcePool)resourcePool.
- 7 Cliquez sur l'onglet **Propriétés** pour (VC:ResourcePool)resourcePool.
- 8 Cliquez avec le bouton droit dans l'onglet **Propriétés**, puis sélectionnez **Ajouter une propriété > Entrée obligatoire**.
- 9 Cliquez avec le bouton droit dans l'onglet **Propriétés**, puis sélectionnez **Ajouter une propriété > Sélectionner la valeur en tant que...**  
  
Lorsque vous définissez cette propriété, vous déterminez la manière dont l'utilisateur sélectionne la valeur du paramètre d'entrée (VC:ResourcePool)resourcePool.
- 10 Faites glisser le paramètre (VC:ResourcePool)resourcePool dans le groupe d'affichage **Pool de ressources**.

Vous avez créé la mise en forme de la boîte de dialogue qui s'affiche lorsque les utilisateurs exécutent le workflow.

#### Étape suivante

Vous avez terminé le développement de l'exemple de workflow complexe. Vous pouvez désormais valider et exécuter le workflow.

## Valider et exécuter l'exemple de workflow complexe

Une fois un workflow créé, vous avez la possibilité de le valider afin d'y déceler toute erreur éventuelle. Si le workflow est exempt d'erreurs, vous pouvez l'exécuter.

#### Conditions préalables

Créer un workflow, agencer son schéma, définir les liens et les liaisons, définir les propriétés de paramètre et créer la présentation de la boîte de dialogue des paramètres d'entrée.

Exécutez les tâches suivantes.

- [Créer l'exemple de workflow complexe.](#)
- [Créer une action personnalisée pour l'exemple de workflow complexe.](#)
- [Créer le schéma de l'exemple de workflow complexe.](#)
- [Définir les paramètres de l'exemple de workflow complexe.](#)
- [Définir les liaisons de l'exemple de workflow complexe.](#)
- [Définir les propriétés d'attribut d'un exemple de workflow complexe.](#)
- [Créer la mise en forme des paramètres d'entrée de l'exemple de workflow complexe.](#)

- Ouvrez le workflow pour le modifier dans l'éditeur de workflows.

### Procédure

- 1 Cliquez sur **Validation** dans l'onglet **Schéma** de l'éditeur de workflow.

L'outil de validation décèle toute erreur dans la définition du workflow.

- 2 Une fois les éventuelles erreurs éliminées, cliquez sur **Enregistrer et fermer** au bas de l'éditeur de workflow.

Vous retournez alors au client Orchestrator.

- 3 Cliquez sur la vue **Workflows**.

- 4 Dans la liste hiérarchique des workflows, sélectionnez **Exemples de workflows > Créer des snapshots de toutes les machines virtuelles dans un pool de ressources**.

- 5 Cliquez avec le bouton droit sur le workflow **Créer des snapshots de toutes les machines virtuelles dans un pool de ressources** et sélectionnez **Démarrer le workflow**.

La boîte de dialogue des paramètres d'entrée s'ouvre et vous invite à indiquer un pool de ressources qui contient les machines virtuelles dont il faut créer un snapshot.

- 6 Cliquez sur **Envoyer** pour exécuter le workflow.

Un jeton de workflow apparaît sous le workflow Créer des snapshots de toutes les machines virtuelles dans un pool de ressources.

- 7 Cliquez sur le jeton du workflow pour suivre la progression du workflow dans son exécution.

Si l'exécution du workflow aboutit, le workflow crée un snapshot de l'ensemble des machines virtuelles du pool de ressources sélectionné.

### Étape suivante

Vous pouvez générer un document dans lequel vous pourrez consulter les informations relatives au workflow. Consultez [Générer la documentation sur les workflows](#).

# Scripts

Orchestrator utilise le JavaScript pour créer des blocs constitutifs à partir desquels vous créez des actions, des éléments de workflow et des stratégies qui accèdent aux API des technologies que vous enfichez dans Orchestrator.

Orchestrator emploie le moteur JavaScript Mozilla Rhino 1.7R4 comme moteur de scripts. Le moteur de scripts offre une vérification des types de variable, une gestion des espaces de noms, une rédaction automatique et un traitement des exceptions.

Le moteur de workflow d'Orchestrator vous permet d'utiliser les fonctionnalités de base du langage JavaScript, comme les si, les boucles, les tableaux et les chaînes. Vous pouvez vous servir des objets des scripts que l'API Orchestrator propose ou des objets de tout autre API que vous importez dans Orchestrator au moyen d'un plug-in et que vous mappez aux objets JavaScript. Pour obtenir des informations sur Rhino, consultez le site Web Mozilla Rhino.

Ce chapitre contient les rubriques suivantes :

- [Éléments d'Orchestrator réclamant l'écriture de scripts](#)
- [Limites de l'implémentation Mozilla Rhino dans Orchestrator](#)
- [Utilisation de l'API de scripts d'Orchestrator](#)
- [Utilisation des expressions XPath avec le plug-in vCenter Server](#)
- [Directives de gestion des exceptions](#)
- [Exemples JavaScript d'Orchestrator](#)

## Éléments d'Orchestrator réclamant l'écriture de scripts

Vous n'aurez pas à écrire de scripts pour la totalité des éléments d'Orchestrator. Pour offrir le maximum de flexibilité à vos applications, vous pouvez personnaliser certains éléments par l'ajout de fonctions JavaScript.

Vous pouvez ajouter des scripts dans les éléments suivants d'Orchestrator.

### Actions

Les actions sont des fonctions en script. Vous pouvez limiter les scripts que vous écrivez pour une action à une seule opération afin de maximiser le potentiel de réutilisation des actions par d'autres éléments, tels que d'autres workflows. L'autre possibilité est qu'une action renferme de

nombreuses opérations afin de limiter la complexité des workflows, même si cela réduira les possibilités de réutilisation de l'action.

### Stratégies

Vous définissez des stratégies en utilisant des scripts qui guettent la survenue d'événements déclencheurs. Au moment où l'événement déclencheur survient, les stratégies lancent les opérations d'orchestration que vous définissez dans les scripts.

### Workflows

L'élément de workflow Tâche inscriptible vous permet d'écrire une opération ou séquence d'opérations personnalisée en script que vous pourrez utiliser dans les workflows. Vous avez également la possibilité de définir une instruction de décision booléenne pour des éléments de décision personnalisées renvoyant la valeur soit `true`, soit `false`.

## Limites de l'implémentation Mozilla Rhino dans Orchestrator

Orchestrator s'appuie sur le moteur JavaScript Mozilla Rhino 1.7R4. Toutefois, l'implémentation de Rhino dans Orchestrator présente quelques limites.

Lorsque vous écrivez des scripts pour des workflows, vous devez tenir compte des limites suivantes pour l'implémentation Mozilla Rhino dans Orchestrator.

- Lorsqu'un workflow s'exécute, les objets qui passent d'un élément de workflow à un autre ne sont pas des objets JavaScript. Ce qui est transmis d'un élément à l'élément suivant est la sérialisation d'un objet Java possédant une image JavaScript. Ainsi, vous ne pouvez pas utiliser l'ensemble du langage JavaScript, mais seulement les classes présentes dans l'explorateur de l'API. Vous ne pouvez pas transmettre les objets des fonctions d'un élément de workflow à un autre.
- Orchestrator exécute le code dans des éléments de tâche inscriptibles, dans un contexte autre que le contexte racine Rhino. Orchestrator insère les éléments de tâche et les actions de manière transparente dans les fonctions JavaScript, puis il les exécute. Un élément de tâche contenant `System.log(this);` ne s'affiche pas dans l'objet global `this` de la même manière qu'une implémentation Rhino standard.
- Vous pouvez uniquement appeler des actions qui renvoient des objets non sérialisables depuis le script, mais pas depuis les workflows. Pour appeler une action qui renvoie un objet non sérialisable, vous devez écrire un élément de tâche inscriptible qui appellera l'action à l'aide de la méthode `System.getModule(ModuleName).action()`.
- La validation des workflows ne vérifie pas si le type d'attribut d'un workflow est différent du type d'attribut d'une action ou d'un workflow secondaire. Si vous modifiez le type d'un paramètre d'entrée d'un workflow, par exemple de `VIM3:VirtualMachine` à `VC:VirtualMachine` et que vous ne mettez pas à jour les tâches ou actions inscriptibles qui utilisent le type d'entrée d'origine, le workflow est validé, mais ne s'exécute pas.

# Utilisation de l'API de scripts d'Orchestrator

L'API Orchestrator expose tous les objets et les fonctions des technologies auxquels Orchestrator accède via ses plug-ins, comme les méthodes et les objets JavaScript.

Par exemple, vous pouvez accéder aux implémentations JavaScript de l'API vCenter Server via l'API Orchestrator afin d'inclure des opérations de vCenter aux éléments en script que vous créez. Vous pouvez aussi accéder aux implémentations JavaScript d'objets issues de tous les autres plug-ins que vous installez dans le serveur Orchestrator. Si vous créez un plug-in personnalisé pour s'enficher dans une application tierce, vous mappez les objets de son API aux objets JavaScript que l'API Orchestrator expose alors.

## Procédure

### 1 [Accéder au moteur de scripts depuis l'éditeur de workflow](#)

Le moteur de scripts Orchestrator s'appuie sur le moteur JavaScript Mozilla Rhino 1.7R4 afin de vous aider à rédiger des scripts pour les éléments en script des workflows. Vous pouvez accéder au moteur de scripts pour les éléments en script des workflows à partir de l'onglet **Script** d'éditeur de workflow.

### 2 [Accéder au moteur de scripts depuis l'éditeur d'actions ou de stratégies](#)

Le moteur de scripts Orchestrator s'appuie sur le moteur JavaScript Mozilla Rhino pour vous aider à rédiger des scripts d'actions ou de stratégies. Vous pouvez accéder au moteur de scripts d'actions ou de stratégies à partir des onglets **Script** des éditeurs d'actions ou de stratégies.

### 3 [Accéder à l'explorateur d'API Orchestrator](#)

Orchestrator propose un explorateur d'API qui vous permet d'effectuer des recherches dans l'API Orchestrator et de consulter la documentation des objets JavaScript que vous pouvez utiliser dans les éléments en script.

### 4 [Servez-vous de l'Explorateur d'API Orchestrator pour trouver des objets.](#)

L'API Orchestrator expose l'API de l'ensemble des technologies en plug-in, dont l'intégralité de l'API vCenter Server. L'Explorateur d'API Orchestrator vous aide à trouver les objets dont vous avez besoin pour les ajouter aux scripts.

### 5 [Rédaction de scripts](#)

Le moteur de scripts Orchestrator vous aide à rédiger des scripts. L'insertion automatique de fonctions et l'écriture intuitive des lignes de scripts accélèrent le processus et réduisent les erreurs potentielles de rédaction.

### 6 [Ajouter des paramètres aux scripts](#)

Le moteur de scripts Orchestrator vous aide à importer les paramètres disponibles dans des scripts.

### 7 [Accès au système de fichiers Orchestrator à partir de JavaScript et des workflows](#)

Dans Orchestrator, l'accès au système de fichiers du serveur à partir de JavaScript et des workflows est limité à certains répertoires.

## 8 Accès aux catégories Java à partir de JavaScript

Par défaut, Orchestrator limite l'accès JavaScript à un ensemble limité de catégories Java. Si vous demandez un accès JavaScript à un éventail plus large de catégories Java, vous devez définir une propriété du système Orchestrator.

## 9 Accès aux commandes du système d'exploitation à partir de JavaScript

L'API d'Orchestrator propose une catégorie de scripts, **Command**, qui exécute les commandes dans le système d'exploitation hôte du serveur Orchestrator. Pour éviter tout accès non autorisé à l'hôte du serveur Orchestrator, les applications Orchestrator ne disposent, par défaut, d'aucune autorisation pour exécuter la catégorie **Command**.

## Accéder au moteur de scripts depuis l'éditeur de workflow

Le moteur de scripts Orchestrator s'appuie sur le moteur JavaScript Mozilla Rhino 1.7R4 afin de vous aider à rédiger des scripts pour les éléments en script des workflows. Vous pouvez accéder au moteur de scripts pour les éléments en script des workflows à partir de l'onglet **Script** de l'éditeur de workflow.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez avec le bouton droit sur un workflow dans la vue **Workflows** du client Orchestrator, puis sélectionnez **Modifier**.
- 3 Cliquez sur l'onglet **Schéma** dans l'éditeur de workflow.
- 4 Ajoutez un élément Tâche inscriptible ou un élément Décision personnalisée au schéma de workflow.
- 5 Cliquez sur l'onglet **Script** de l'élément inscriptible.

Vous avez accédé au moteur de scripts pour définir les fonctions en script des éléments de workflow. L'onglet **Script** vous permet de naviguer dans l'API, de consulter la documentation sur les objets, de rechercher des objets et de rédiger en JavaScript.

### Étape suivante

Effectuez des recherches dans l'API Orchestrator à l'aide de l'explorateur d'API.

## Accéder au moteur de scripts depuis l'éditeur d'actions ou de stratégies

Le moteur de scripts Orchestrator s'appuie sur le moteur JavaScript Mozilla Rhino pour vous aider à rédiger des scripts d'actions ou de stratégies. Vous pouvez accéder au moteur de scripts d'actions ou de stratégies à partir des onglets **Script** des éditeurs d'actions ou de stratégies.



## Procédure

- 1 Sélectionnez une option dans le menu déroulant du client Orchestrator en fonction du type d'élément dont vous souhaitez modifier le script.

Option	Description
<b>Concevoir</b>	Sélectionnez cette option pour modifier le script d'un élément d'action.
<b>Exécuter</b>	Sélectionnez cette option pour modifier le script d'une stratégie.

- 2 Cliquez avec le bouton droit sur une action ou une stratégie dans les vues **Actions** ou **Stratégies**, puis sélectionnez **Modifier**.
- 3 Cliquez sur l'onglet **Script** dans l'éditeur d'actions ou de stratégies.

Vous avez accédé au moteur de scripts pour définir les fonctions en script des éléments d'action ou de stratégie. L'onglet **Script** vous permet de naviguer dans l'API, de consulter la documentation sur les objets, de rechercher des objets et de rédiger en JavaScript.

### Étape suivante

Effectuez des recherches dans l'API Orchestrator à l'aide de l'explorateur d'API.

## Accéder à l'explorateur d'API Orchestrator

Orchestrator propose un explorateur d'API qui vous permet d'effectuer des recherches dans l'API Orchestrator et de consulter la documentation des objets JavaScript que vous pouvez utiliser dans les éléments en script.

Vous pouvez consulter une version en ligne de l'API de scripts pour le plug-in vCenter Server sur la page d'accueil de la documentation d'Orchestrator.

## Procédure

- 1 Connectez-vous au client Orchestrator.
- 2 Sélectionnez **Outils > Explorateur d'API**.

L'explorateur d'API s'affiche. Vous pouvez l'utiliser pour rechercher tous les objets et fonctions de l'API Orchestrator.

### Étape suivante

Utilisez l'explorateur d'API afin de rédiger les scripts des éléments inscriptibles.

## Servez-vous de l'Explorateur d'API Orchestrator pour trouver des objets.

L'API Orchestrator expose l'API de l'ensemble des technologies en plug-in, dont l'intégralité de l'API vCenter Server. L'Explorateur d'API Orchestrator vous aide à trouver les objets dont vous avez besoin pour les ajouter aux scripts.

## Conditions préalables

Ouvrez l'Explorateur d'API.

### Procédure

- 1 Saisissez le nom ou une partie du nom d'un objet dans la zone de texte **Recherche** de l'Explorateur d'API et cliquez sur **Rechercher**.

Pour limiter votre recherche à un type d'objet en particulier, cochez les cases **Classe de scripts**, **Attributs et méthodes** et **Types et énumérations**.

- 2 Double-cliquez sur l'élément figurant dans la liste proposée.

L'objet est en évidence dans la liste hiérarchique à gauche. Un volet de documentation sous la liste hiérarchique présente des informations sur l'objet.

### Étape suivante











Utilisez les objets que vous trouvez dans les scripts.

## Objets JavaScript dans l'Explorateur d'API

L'Explorateur d'API Orchestrator identifie et regroupe les différentes sortes d'objets JavaScript dans l'arborescence hiérarchique de la partie gauche de l'onglet **Script** ou dans la boîte de dialogue de l'Explorateur d'API. L'Explorateur d'API utilise des icônes pour vous aider à identifier les différentes sortes d'objet.

Le tableau qui suit décrit les objets de l'API Orchestrator et montre leur icône.

**Tableau 2-1. Objets JavaScript dans l'API Orchestrator**

Objet	Icône dans la liste hiérarchique	Description
Type		Types
Jeu de fonctions		Type interne contenant un jeu de méthodes statiques
Primitive		Types primitifs
Objet		Objets de scripts standard d'Orchestrator
Attribut		attributs JavaScript
Méthode		méthodes JavaScript
Constructeur		constructeurs JavaScript
Énumération		énumérations JavaScript
Jeu de chaînes		Jeu de chaînes, valeurs par défaut
Module		Un ensemble d'actions
Plug-in	Image que le plug-in définit	Les API que le plug-in expose à Orchestrator

## Rédaction de scripts

Le moteur de scripts Orchestrator vous aide à rédiger des scripts. L'insertion automatique de fonctions et l'écriture intuitive des lignes de scripts accélèrent le processus et réduisent les erreurs potentielles de rédaction.

### Conditions préalables

Ouvrez un élément en script à modifier, puis cliquez sur son onglet **Script**.

### Procédure

- 1 Naviguez dans la liste hiérarchique des objets à gauche de l'onglet **Script** ou utilisez la fonction de recherche de l'explorateur d'API pour sélectionner un type, une catégorie ou une méthode à ajouter au script.

- 2 Cliquez avec le bouton droit sur le type, la catégorie ou la méthode, puis sélectionnez **Copier**.

Si le moteur de scripts ne vous permet pas de copier l'élément que vous avez sélectionné, cela signifie que cet objet n'est pas utilisable dans le contexte du script.

- 3 Cliquez avec le bouton droit sur la zone de script, puis collez l'élément que vous avez copié à l'emplacement approprié dans le script.

Le moteur de scripts intègre l'élément au script avec le nom de son créateur et de son instance.

Par exemple, si vous avez copié l'objet `Date`, le moteur de scripts colle le code suivant dans le script.

```
var myDate = new Date();
```

- 4 Copiez/collez une méthode à ajouter au script.

Le moteur de scripts effectue l'appel de méthode en ajoutant les attributs demandés.

Par exemple, si vous avez copié la méthode `cloneVM()` à partir du module `com.vmware.library.vc.vm`, le moteur de scripts colle le code suivant dans le script.

```
System.getModule("com.vmware.library.vc.vm").cloneVM(vm, folder, name, spec)
```

Le moteur de scripts met en surbrillance les paramètres que vous avez déjà définis dans l'élément. Les paramètres non définis ne sont pas mis en surbrillance.

- 5 Placez le curseur à la fin d'un élément que vous avez collé dans le script, puis appuyez sur **Ctrl** + **Espace** pour effectuer une sélection dans une liste contextuelle des méthodes et attributs applicables que l'objet peut appeler.

---

**Note** La fonction d'écriture intuitive en est encore au stade de l'expérimentation.

---

Vous avez ajouté des objets et des fonctions au script.

### Étape suivante

Ajoutez des paramètres au script.

## Code couleur des mots-clés de script

Lorsque vous ajoutez des scripts à l'onglet **Scripts** de l'élément de workflow concerné, certains mots-clés apparaissent dans une couleur différente afin d'optimiser la lecture du code.

Sauf mention contraire, tous les scripts s'affichent dans une police noire standard.

**Tableau 2-2. Code couleur des mots-clés de script**

Type de mot-clé	Couleur du texte dans l'onglet de script
Les mots-clés JavaScript, par exemple <code>if</code> , <code>else</code> , <code>for</code> et <code>new</code>	Noir, gras
Déclarations de variables, notamment <code>var</code>	Vert
Agents modificateurs dans les boucles, par exemple <code>in</code>	Rouge
Valeurs de variables invalides	Violet
Valeurs de variables valides	Vert
Commentaires du code	Gris, italique
Types d'objets de plug-in Orchestrator, par exemple <code>VC:VirtualMachine</code> ou <code>VC:Host</code>	Vert
Texte de sortie	Vert
Attributs du workflow	Rose
Entrées du workflow	Rose
Sorties du workflow	Rose

## Ajouter des paramètres aux scripts

Le moteur de scripts Orchestrator vous aide à importer les paramètres disponibles dans des scripts.

Si vous avez déjà défini les paramètres de l'élément que vous modifiez, ils s'affichent en tant que liens dans la barre d'outils de l'onglet **Script**.

### Conditions préalables

Un élément en script est ouvert pour modification et son onglet **Script** est également ouvert.

### Procédure

- 1 Déplacez le curseur vers l'emplacement approprié du script, dans la zone de script de l'onglet **Script**.
- 2 Cliquez sur le lien du paramètre dans la barre d'outils de l'onglet **Script**.

Orchestrator insère le paramètre au niveau de la position du curseur.

- 3 Insérez un paramètre avec une valeur nulle dans le script.

Si vous transmettez des valeurs nulles vers des types primitifs tels que des entiers, des booléens et des chaînes, l'API de script Orchestrator définit automatiquement la valeur par défaut pour cet argument.

Vous avez ajouté des paramètres au script.

**Étape suivante**

Ajoutez un accès aux catégories Java dans les scripts.

## Accès au système de fichiers Orchestrator à partir de JavaScript et des workflows

Dans Orchestrator, l'accès au système de fichiers du serveur à partir de JavaScript et des workflows est limité à certains répertoires.

Les fonctions JavaScript et les workflows disposent uniquement d'une autorisation en lecture, écriture et exécution dans le répertoire permanent `c:\orchestrator`.

L'administrateur Orchestrator peut modifier les dossiers auxquels les fonctions JavaScript et les workflows disposent d'un accès en lecture, écriture et exécution en configurant une propriété système. Pour plus d'informations sur la configuration des propriétés système, reportez-vous à *Installation et configuration de VMware vRealize Orchestrator*.

Les fonctions JavaScript et les workflows disposent également d'une autorisation en lecture, écriture et exécution dans le dossier d'E/S temporaire par défaut du système du serveur. L'écriture dans le dossier d'E/S temporaire par défaut est le seul moyen indépendant, garanti et indépendant de la configuration d'accéder au système de fichiers avec l'ensemble des autorisations. Toutefois, les fichiers que vous écrivez dans le dossier d'E/S temporaire seront perdus au redémarrage du serveur.

Vous pouvez obtenir le dossier d'E/S temporaire par défaut à l'aide de la méthode `System.getTempDirectory` des fonctions JavaScript.

### Accéder au système de fichiers du serveur à l'aide de la méthode `System.getTempDirectory`

Outre le fait d'écrire dans les dossiers du système du serveur Orchestrator où l'administrateur a défini les autorisations adéquates, il est également possible d'écrire dans le dossier d'E/S temporaire par défaut.

En configuration de base, Orchestrator dispose de l'ensemble des droits de lecture, écriture et exécution dans le dossier d'E/S par défaut. Vous pouvez obtenir le dossier d'E/S temporaire par défaut à l'aide de la méthode `System.getTempDirectory` des fonctions JavaScript.

**Procédure**

- ◆ Insérez la ligne de code suivante dans les fonctions JavaScript pour accéder au dossier `java.io.temp-dir`.

```
var tempDir = System.getTempDirectory()
```

### Accès aux catégories Java à partir de JavaScript

Par défaut, Orchestrator limite l'accès JavaScript à un ensemble limité de catégories Java. Si vous demandez un accès JavaScript à un éventail plus large de catégories Java, vous devez définir une propriété du système Orchestrator.

Par défaut, le moteur JavaScript d'Orchestrator peut uniquement accéder aux catégories du module `java.util.*`.

L'administrateur d'Orchestrator peut autoriser un accès aux autres catégories Java à partir des fonctions JavaScript en configurant une propriété système. Pour plus d'informations sur la configuration des propriétés système, reportez-vous à *Installation et configuration de VMware vRealize Orchestrator*.

## Accès aux commandes du système d'exploitation à partir de JavaScript

L'API d'Orchestrator propose une catégorie de scripts, `Command`, qui exécute les commandes dans le système d'exploitation hôte du serveur Orchestrator. Pour éviter tout accès non autorisé à l'hôte du serveur Orchestrator, les applications Orchestrator ne disposent, par défaut, d'aucune autorisation pour exécuter la catégorie `Command`.

L'administrateur Orchestrator peut autoriser l'accès à la catégorie de script `Command` en configurant la propriété du système `com.vmware.js.allow-local-process=true`.

Pour plus d'informations sur la configuration des propriétés système, reportez-vous à *Installation et configuration de VMware vCenter Orchestrator*.

Pour plus d'informations sur la configuration des propriétés système, reportez-vous à *Installation et configuration de VMware vCenter Orchestrator*.

## Utilisation des expressions XPath avec le plug-in vCenter Server

Vous pouvez utiliser des méthodes de fonction de recherche dans le plug-in vCenter Server pour demander des objets d'inventaire vCenter Server. Vous pouvez utiliser des expressions XPath pour définir des paramètres de recherche.

Le plug-in vCenter Server inclut un ensemble de méthodes de fonction de recherche d'objets telles que `getAllDatastores()`, `getAllResourcePools()` et `findAllForType()`. Vous pouvez utiliser ces méthodes pour accéder aux inventaires des instances vCenter Server connectées à votre serveur Orchestrator et rechercher des objets par ID, nom ou d'autres propriétés.

À des fins de performances, les méthodes de recherche ne renvoient pas de propriétés pour les objets ayant été interrogés, sauf si vous spécifiez un ensemble de propriétés dans la requête de recherche.

Vous pouvez consulter une version en ligne de l'API de script pour le plug-in vCenter Server sur la page d'accueil de la documentation d'Orchestrator.

---

**Important** Les requêtes basées sur des expressions XPath peuvent avoir un impact sur les performances d'Orchestrator, car la méthode de fonction de recherche renvoie tous les objets d'un type donné du côté vCenter Server et les filtres de recherche sont appliqués du côté du plug-in vCenter Server.

---

## Utilisation des expressions XPath avec le plug-in vCenter Server

Lorsque vous appelez une méthode de fonction de recherche, vous pouvez utiliser des expressions basées sur le langage de requête XPath. La recherche renvoie tous les objets d'inventaire qui

correspondent aux expressions XPath. Si vous souhaitez interroger des propriétés, vous pouvez les inclure dans le script de recherche dans le formulaire d'un groupe de chaînes.

L'exemple Javascript suivant utilise l'objet de script `VcPlugin` et une expression XPath pour renvoyer les noms de tous les objets du magasin de données qui font partie des objets gérés de vCenter Server et qui contiennent la chaîne **ds** dans leur nom.

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

La même expression XPath peut être appelée à l'aide de l'objet de script `Server` et de la méthode de fonction de recherche `findAllForType`.

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

L'exemple de script suivant renvoie les noms de tous les objets du système hôte dont l'ID commence par le chiffre **1**.

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

Le script suivant renvoie les noms et les ID de tous les objets de centre de données qui contiennent la chaîne **DC**, en majuscules ou minuscules, dans leur nom. Le script récupère également la propriété de **balise**.

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

## Directives de gestion des exceptions

L'implémentation Orchestrator du moteur JavaScript Mozilla Rhino prend en charge la gestion des exceptions pour vous permettre de traiter les erreurs. Vous devez suivre les directives suivantes lorsque vous rédigez des gestionnaires d'exceptions en scripts.

- Utilisez les types d'erreurs de l'ECMA (European Computer Manufacturers Association). Utilisez `Error` en tant qu'exception générique renvoyée par les fonctions du plug-in et les types d'erreurs spécifiques suivants.
  - `TypeError`
  - `RangeError`

- EvalError
- ReferenceError
- URIError
- SyntaxError

L'exemple suivant illustre la définition d'un URIError.

```
try {
    ...
    throw new URIError("VirtualMachine with ID 'vm-0056'
        not found on 'vcenter-test-1'");
    ...
} catch ( e if e instanceof URIError ) {
}
}
```

- Toutes les exceptions non récupérées par les scripts doivent être des objets de chaîne simples du formulaire <type>:SPACE<human readable message>, comme illustré dans l'exemple suivant.

```
throw "ValidationError: The input parameter 'myParam' of type 'string' is too short."
```

- Rédigez des messages aussi clairs que possible et facilement lisibles.
- La vérification du type d'exception de chaîne simple doit s'appuyer sur le schéma suivant.

```
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 vérification du type d'exception de chaîne simple doit s'appuyer sur le schéma suivant dans les éléments en script des workflows.

```
if (typeof(errorCode)=="string"
    && errorCode.indexOf("VMwareNoSpaceLeftOnDatastore:")
    == 0) {
    // Do something useful here
}
}
```



# Exemples JavaScript d'Orchestrator

Vous pouvez couper, coller et adapter les exemples JavaScript d'Orchestrator pour vous aider à écrire du JavaScript pour les tâches d'orchestration courantes.

- [Exemples de scripts de base](#)

Les éléments, actions et stratégies en script du workflow exigent que les tâches courantes soient rédigées sous forme de scripts. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

- [Exemples de scripts d'e-mails](#)

Les éléments en script des workflows peuvent inclure des scripts des tâches courantes relatives aux e-mails. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

- [Exemples de scripts du système de fichiers](#)

Les éléments, actions et stratégies en scripts du workflow exigent que les scripts soient rédigés dans des tâches du système de fichiers commun. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

- [Exemples de scripts LDAP](#)

Les éléments, actions et stratégies en script du workflow nécessitent l'écriture de scripts pour réaliser des tâches LDAP courantes. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

- [Exemples de scripts de journalisation](#)

Les éléments, actions et stratégies en script du workflow nécessitent l'écriture de scripts pour réaliser des tâches courantes de journalisation. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

- [Exemples de scripts de mise en réseau](#)

Les éléments, actions et stratégies en script du workflow réclament d'écrire sous forme de script les tâches de mise en réseau courantes. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

- [Exemples de scripts de workflow](#)

Les éléments, actions et stratégies en script du workflow réclament d'écrire sous forme de script les exemples de tâches courantes. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

## Exemples de scripts de base

Les éléments, actions et stratégies en script du workflow exigent que les tâches courantes soient rédigées sous forme de scripts. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

## Accéder aux documents XML

L'exemple JavaScript suivant vous permet d'accéder aux documents XML à partir de JavaScript en utilisant l'implémentation ECMAScript for XML (E4X) dans l'API JavaScript Orchestrator.

**Note** Outre l'implémentation d'E4X dans l'API JavaScript, Orchestrator fournit également une implémentation XML DOM (Document Object Model) dans le plug-in XML. Pour en savoir plus sur le plug-in XML et ses modèles de workflows, consultez *Utilisation des plug-ins 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 + ")");
}
```

## Configuration et récupération des propriétés à partir d'une table de hachage

L'exemple JavaScript suivant permet de définir les propriétés dans une table de hachage et de récupérer ces propriétés à partir de la table de hachage. Dans l'exemple suivant, la clé est toujours une chaîne et la valeur est un objet, un nombre, un booléen ou une chaîne.

```
var table = new Properties() ;
table.put("myKey",new Date()) ;
// get the object back
var myDate= table.get("myKey") ;
System.log("Date is : "+myDate) ;
```

## Remplacer le contenu d'une chaîne

L'exemple JavaScript suivant permet de remplacer le contenu d'une chaîne avec un nouveau contenu.

```
var str1 = "'hello'" ;
var reg = new RegExp("'", "g");
var str2 = str1.replace(reg,"\\'") ;
System.log(""+str2) ; // result : \'hello\'
```

## Comparer les types

L'exemple JavaScript suivant permet de vérifier si un objet correspond à un type d'objet donné.

```
var path = 'myurl/test';
if(typeof(path, string)){
    throw("string");
} else {
    throw("other");
}
```

## Exécuter une commande dans le serveur Orchestrator

L'exemple JavaScript suivant vous permet d'exécuter une ligne de commande sur le serveur Orchestrator. Utilisez les mêmes informations d'identification que celles qui ont servi à démarrer le serveur.

---

**Note** L'accès au système de fichiers est limité par défaut.

---

```
var cmd = new Command("ls -al") ;
cmd.execute(true) ;
System.log(cmd.output) ;
```

## Exemples de scripts d'e-mails

Les éléments en script des workflows peuvent inclure des scripts des tâches courantes relatives aux e-mails. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

Lorsque vous exécutez un workflow d'e-mail, il utilise la même configuration de serveur de messagerie par défaut que celle définie dans le workflow Configurer le courrier. Vous pouvez remplacer les valeurs par défaut en utilisant des paramètres d'entrée ou en définissant des valeurs personnalisées dans les éléments en script des workflows.

## Obtenir une adresse e-mail

L'exemple JavaScript suivant permet d'obtenir l'adresse e-mail du propriétaire d'un script en cours d'exécution.

```
var emailAddress = Server.getRunningUser().emailAddress ;
```

## Envoyer un e-mail

L'exemple JavaScript suivant permet d'envoyer un e-mail au destinataire défini, avec le contenu défini, par le biais d'un serveur SMTP.

```
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() ;
```

## Récupérer des messages électroniques

L'exemple JavaScript suivant permet de récupérer les messages d'un compte de messagerie donné, sans les supprimer, à l'aide de l'API de script fournie par la catégorie 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();
}

```

## Exemples de scripts du système de fichiers

Les éléments, actions et stratégies en scripts du workflow exigent que les scripts soient rédigés dans des tâches du système de fichiers commun. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

### Ajouter du contenu à un fichier texte simple

L'exemple JavaScript suivant permet d'ajouter du contenu à un fichier texte.

```

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() ;

```

### Récupérer le contenu d'un fichier

L'exemple JavaScript suivant permet de récupérer le contenu d'un fichier auprès de la machine hébergeant le serveur Orchestrator.

```

var tempDir = System.getTempDirectory() ;
var fileReader = new FileReader(tempDir + "/readme.txt") ;
fileReader.open() ;
var fileContentAsString = fileReader.readAll();
fileReader.close() ;

```

## Exemples de scripts LDAP

Les éléments, actions et stratégies en script du workflow nécessitent l'écriture de scripts pour réaliser des tâches LDAP courantes. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

## Convertir un objet LDAP en objet Active Directory

L'exemple JavaScript suivant effectue la conversion d'éléments de groupe LDAP en objets Groupe d'utilisateurs Active Directory et la conversion inverse.

```
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) ;
```

## Exemples de scripts de journalisation

Les éléments, actions et stratégies en script du workflow nécessitent l'écriture de scripts pour réaliser des tâches courantes de journalisation. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

### Journalisation permanente

L'exemple JavaScript suivant crée des entrées de journal permanentes.

```
Server.log("This is a persistent message", "enter a long description here");
Server.warn("This is a persistent warning", "enter a long description here");
Server.error("This is a persistent error", "enter a long description here");
```

### Journalisation non permanente

L'exemple JavaScript suivant crée des entrées de journal non permanentes.

```
System.log("This is a non-persistent log message");
System.warn("This is a non-persistent log warning");
System.error("This is a non-persistent log error");
```

## Exemples de scripts de mise en réseau

Les éléments, actions et stratégies en script du workflow réclament d'écrire sous forme de script les tâches de mise en réseau courantes. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

### Obtenir un texte auprès d'une URL

L'exemple JavaScript suivant accède à une URL, obtient un texte, puis convertit celui-ci en chaîne.

```
var url = new URL("http://www.vmware.com") ;
var htmlContentAsString = url.getContent() ;
```

## Exemples de scripts de workflow

Les éléments, actions et stratégies en script du workflow réclament d'écrire sous forme de script les exemples de tâches courantes. Vous avez la possibilité de couper, coller et adapter ces exemples dans vos éléments en script.

## Renvoyer tous les workflows exécutés par l'utilisateur actuel

L'exemple JavaScript suivant récupère toutes les exécutions de workflow auprès du serveur et vérifie si elles appartiennent à l'utilisateur actuel.

```
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;
```

## Accéder au jeton du workflow en cours

Vous pouvez accéder au jeton du workflow en cours en utilisant la variable `workflow`. Il s'agit d'un objet de type `WorkflowToken` qui permet d'accéder à l'exécution de workflow actuelle. L'exemple JavaScript suivant obtient l'identifiant du jeton de workflow et sa date de début.

```
System.log("Current workflow run ID: " + workflow.id);
System.log("Current workflow run start date: "+workflow.startDate);
```

## Planifier un workflow

L'exemple JavaScript suivant démarre un workflow avec un jeu de propriétés donné, puis planifie son exécution une heure plus tard.

```
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);
```

## Exécuter un workflow sur une sélection d'objets en boucle

L'exemple JavaScript suivant prend le tableau des machines virtuelles et exécute un workflow sur chacune d'elles dans une boucle `For`. `VMs` et `workflowToRun` constituent des entrées du workflow.

```
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);
}
```



# Développement d'actions

Orchestrator propose des bibliothèques d'actions prédéfinies. Les actions représentent des fonctions individuelles que vous utilisez en tant que composants de base dans les workflows et dans les scripts.

Les actions sont des fonctions JavaScript. Elles prennent en charge plusieurs paramètres d'entrée pour une seule valeur de retour. Elles peuvent appeler n'importe quel objet de l'API d'Orchestrator, ou encore des objets de n'importe quelle API que vous importez dans Orchestrator via un plug-in.

Lorsqu'un workflow s'exécute, l'action récupère les paramètres d'entrée auprès des attributs du workflow. Ces attributs peuvent correspondre soit aux paramètres d'entrée initiaux du workflow, soit aux attributs que les autres éléments du workflow définissent au cours de leur exécution.

Ce chapitre contient les rubriques suivantes :

- [Réutilisation d'actions](#)
- [Accéder à la vue Actions](#)
- [Composants de la vue Actions](#)
- [Création d'actions](#)
- [Utiliser l'historique des versions des actions](#)
- [Restaurer des actions supprimées](#)

## Réutilisation d'actions

Lorsque vous définissez une fonction individuelle en tant qu'action au lieu de la coder directement au sein d'un élément de workflow Tâche inscriptible, vous l'exposez dans la bibliothèque. Lorsqu'une action est visible dans la bibliothèque, d'autres workflows peuvent l'utiliser.

Lorsque vous définissez des actions indépendamment des workflows qui y font appel, vous pouvez mettre à jour ou optimiser plus facilement les actions. La définition d'actions individuelles permet aussi à d'autres workflows de réutiliser les actions. Lorsqu'un workflow s'exécute, Orchestrator met en cache chaque action la première fois seulement que le workflow l'exécute. Orchestrator est alors en mesure de réutiliser l'action mise en cache. La mise en cache des actions est utile dans le cas d'appels récursifs dans un workflow ou de boucles rapides.

Vous avez la possibilité de dupliquer des actions, de les exporter vers d'autres workflows ou modules ou de les déplacer vers un autre module dans la liste hiérarchique des actions.

## Accéder à la vue Actions

L'interface du client Orchestrator présente une vue **Actions** qui fournit un accès aux bibliothèques d'actions du serveur Orchestrator.

La vue **Actions** de l'interface du client Orchestrator vous propose une liste hiérarchique de l'ensemble des actions disponibles sur le serveur Orchestrator.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Actions**.
- 3 Parcourez les bibliothèques d'actions en développant les nœuds de la liste hiérarchique d'actions.

Vous pouvez utiliser la vue **Actions** pour afficher des informations sur les actions des bibliothèques, ainsi que pour créer et modifier des actions.

## Composants de la vue Actions

Lorsque vous cliquez sur une action de la liste hiérarchique des actions, les informations sur cette action s'affichent dans le volet droit du client Orchestrator.

La vue **Actions** présente quatre onglets.

<b>Général</b>	Affiche les informations générales sur l'action, notamment son nom, son numéro de version, ses autorisations et une description.
<b>Scripts</b>	Indique les types de retour, les paramètres d'entrée et le code JavaScript qui définissent la fonction de cette action.
<b>Événements</b>	Affiche l'ensemble des événements que l'action a rencontré ou déclenché.
<b>Autorisations</b>	Affiche les utilisateurs et groupes d'utilisateurs autorisés à accéder à cette action.

## Création d'actions

Vous pouvez définir des fonctions individuelles, telles que des actions, que les autres éléments comme les workflows peuvent utiliser. Les actions sont des fonctions JavaScript avec des autorisations et des paramètres d'entrée et de sortie définis.

### ■ Créer une action

Lorsque vous définissez une fonction individuelle en tant qu'action, vous n'êtes pas obligé de la coder directement au sein d'un élément de workflow Tâche inscriptible ; vous pouvez l'exposer dans la bibliothèque afin que d'autres workflows puissent l'utiliser.

## ■ Rechercher des éléments qui implémentent une action

Si vous modifiez une action et que vous changez son comportement, vous risquez de rompre accidentellement un workflow ou une application qui implémente cette action. Orchestrator propose une fonction qui permet de rechercher l'ensemble des actions, des workflows ou des modules qui implémentent un élément donné. Vous pouvez vérifier si la modification de l'élément affecte le fonctionnement des autres éléments.

## ■ Directives de codage des actions

Lorsque vous créez des actions, vous devez suivre certaines directives de base de codage qui vous permettront d'optimiser les performances des workflows et de maximiser les chances de réutiliser les actions.

## Créer une action

Lorsque vous définissez une fonction individuelle en tant qu'action, vous n'êtes pas obligé de la coder directement au sein d'un élément de workflow Tâche inscriptible ; vous pouvez l'exposer dans la bibliothèque afin que d'autres workflows puissent l'utiliser.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Actions**.
- 3 Développez la racine de la liste hiérarchique des actions et naviguez jusqu'au module dans lequel vous souhaitez créer l'action.
- 4 Cliquez avec le bouton droit sur le module, puis sélectionnez **Ajouter une action**.
- 5 Entrez un nom pour l'action dans la zone de texte, puis cliquez sur **OK**.  
Votre action personnalisée est ajoutée à la bibliothèque d'actions.
- 6 Cliquez avec le bouton droit sur l'action, puis sélectionnez **Modifier**.
- 7 Cliquez sur l'onglet **Script**.
- 8 Pour modifier le type de renvoi par défaut, cliquez sur le lien **invalide**.
- 9 Ajoutez les paramètres d'entrée de l'action en cliquant sur l'icône en forme de flèche.
- 10 Rédigez le script de l'action.
- 11 Définissez les autorisations de l'action.
- 12 Cliquez sur **Enregistrer et fermer**.

Vous avez créé une action personnalisée et ajouté les paramètres d'entrée de l'action.

### Étape suivante

Vous pouvez utiliser la nouvelle action personnalisée dans un workflow.

## Rechercher des éléments qui implémentent une action

Si vous modifiez une action et que vous changez son comportement, vous risquez de rompre accidentellement un workflow ou une application qui implémente cette action. Orchestrator propose une fonction qui permet de rechercher l'ensemble des actions, des workflows ou des modules qui implémentent un élément donné. Vous pouvez vérifier si la modification de l'élément affecte le fonctionnement des autres éléments.

---

**Important** La fonction **Rechercher des éléments utilisant cet élément** vérifie l'ensemble des modules, des workflows et des stratégies mais elle ne contrôle pas les scripts. Ainsi, la modification d'une action est susceptible d'affecter un élément qui appelle cette action dans un script et que la fonction **Rechercher des éléments utilisant cet élément** n'a pas identifié.

---

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Actions**.
- 3 Développez les nœuds de la liste hiérarchique d'actions pour naviguer jusqu'à une action précise.
- 4 Cliquez avec le bouton droit sur l'action, puis sélectionnez **Rechercher des éléments utilisant cet élément**.

Une boîte de dialogue affiche tous les éléments, tels que les workflows ou modules, qui implémentent cette action.

- 5 Double-cliquez sur un élément dans la liste des résultats pour afficher cet élément dans le client Orchestrator.

Vous venez de trouver tous les éléments qui implémentent une action donnée.

### Étape suivante

Vous pouvez vérifier si la modification de cet élément affectera les autres éléments.

## Directives de codage des actions

Lorsque vous créez des actions, vous devez suivre certaines directives de base de codage qui vous permettront d'optimiser les performances des workflows et de maximiser les chances de réutiliser les actions.

### Directives de base pour les actions

Vous devez vous appuyer sur les directives de base pour créer des actions.

- Chaque action doit s'accompagner d'une description de son rôle et de sa fonction.
- Rédigez des actions courtes et élémentaires, puis associez-les à un workflow.
- Évitez de rédiger des actions qui effectuent plusieurs fonctions, car cela restreindrait la possibilité de les réutiliser.

- Évitez les actions qui s'exécutent sur de longues durées. Créez plutôt une boucle dans le workflow et insérez un événement d'attente ou un élément de minuterie d'attente après l'élément d'action.
- N'inscrivez aucun point de contrôle dans les actions. Les workflows définissent un point de contrôle au début et à la fin de chaque exécution d'élément.
- Évitez d'inscrire des boucles dans les actions. Créez plutôt des boucles dans le workflow. Si le serveur doit redémarrer, les workflows qui étaient en cours d'exécution reprennent au niveau de leur dernier point de contrôle, au début d'un élément. Si vous inscrivez une boucle dans une action et que le serveur redémarre alors que le workflow est en train d'exécuter cette action, le workflow reprend au niveau du point de contrôle du début de cette action et la boucle reprend au point de départ.

## Directives d'attribution de noms aux actions

Utilisez les directives de base lorsque vous nommez des actions.

- Rédigez les noms d'actions en anglais.
- Les noms d'actions doivent commencer par une lettre minuscule. Utilisez une majuscule au début de chaque mot collé dans le nom. Par exemple, `myAction`.
- Faites en sorte que les noms d'actions soient aussi explicites que possible afin que la fonction de l'action soit évidente. Par exemple, `backupAllVMsInPool`.
- Faites en sorte que les noms de modules soient aussi explicites que possible.
- Faites en sorte que les noms de module soient uniques.
- Utilisez le format d'adresse Internet inversée pour les noms de modules. Par exemple, `com.vmware.myactions.myAction`.

## Directives relatives aux paramètres des actions

Utilisez les directives de base lorsque vous rédigez les définitions des paramètres des actions.

- Rédigez les noms de paramètres en anglais.
- Les noms de paramètres doivent commencer par une lettre minuscule.
- Faites en sorte que les noms de paramètres soient aussi explicites que possible.
- Il est préférable de limiter les noms de paramètres à un seul mot. Si le nom doit contenir plusieurs mots, utilisez une majuscule au début de chaque mot collé dans le nom. Par exemple, `myParameter`.
- Utilisez le pluriel pour les paramètres qui représentent un groupe d'objets.
- Faites en sorte que les noms de variables ne soient pas ambigus, par exemple, `displayName`.
- Insérez une description pour chaque paramètre afin d'expliquer son rôle.
- N'utilisez pas trop de paramètres pour une seule action.

## Utiliser l'historique des versions des actions

Vous pouvez utiliser l'historique des versions pour restaurer une action à sa version précédente. Vous pouvez restaurer l'état de l'action à une version antérieure ou ultérieure. Vous pouvez également comparer les différences entre l'état actuel de l'action et une version enregistrée de cette action.

Orchestrator crée un nouvel élément d'historique des versions pour chaque action dès que vous enrichissez et enregistrez une version d'action. Les modifications apportées par la suite ne transforment pas la version enregistrée. Par exemple, lorsque vous créez la version 1.0.0 de l'action et que vous l'enregistrez, l'état de l'action est stocké dans la base de données. Si vous apportez des modifications à l'action, vous pourrez enregistrer l'état de l'action dans le client Orchestrator mais vous ne pourrez pas appliquer ces modifications à la version 1.0.0 de l'action. Pour stocker les modifications dans la base de données, vous devez créer une version d'action suivante et l'enregistrer. L'historique des versions est conservé dans la base de données aux côtés de l'action.

Lorsque vous supprimez une action, Orchestrator indique l'élément comme supprimé de la base de données, mais il ne supprime pas l'historique des versions de l'élément. Vous pouvez ainsi restaurer les actions supprimées. Reportez-vous à [Restaurer des actions supprimées](#).

### Conditions préalables

Ouvrir une action à modifier.

### Procédure

- 1 Cliquez sur l'onglet **Général** de l'éditeur d'actions.
- 2 Cliquez sur **Afficher l'historique des versions**.  
La fenêtre de l'historique des versions s'affiche.
- 3 Sélectionnez une version d'action, puis cliquez sur **Différences avec version actuelle** pour comparer les différences.  
Une fenêtre s'affiche avec les différences entre la version en cours et la version sélectionnée.
- 4 Sélectionnez une version d'action, puis cliquez sur **Restaurer** pour restaurer l'état de l'action.

---

**Attention** Si vous n'avez pas enregistré la version d'action en cours, elle est supprimée de l'historique des versions et vous ne pouvez pas la restaurer.

---

L'état de l'action est restauré à la version sélectionnée.

## Restaurer des actions supprimées

Vous pouvez restaurer des actions supprimées de la bibliothèque.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Actions**.

- 3 Naviguez jusqu'au dossier dans lequel vous souhaitez restaurer une ou plusieurs actions supprimées.
- 4 Cliquez avec le bouton droit sur le dossier, puis sélectionnez **Restaurer des actions supprimées**.
- 5 Sélectionnez l'action ou les actions que vous souhaitez restaurer, puis cliquez sur **Restaurer**.

Les actions concernées apparaissent dans le dossier sélectionné.

# Créer des éléments de ressources

# 4

Les workflows peuvent demander à ce que les objets créés en dehors d'Orchestrator soient utilisés en tant qu'attributs. Pour utiliser les objets externes en tant qu'attributs dans les workflows, importez-les dans le serveur Orchestrator en tant qu'éléments de ressources.

Les objets que les workflows peuvent utiliser en tant qu'éléments de ressources incluent les fichiers image, les scripts, les modèles XML, les fichiers HTML, etc. Tous les workflows s'exécutant sur le serveur Orchestrator peuvent utiliser n'importe quel élément de ressources que vous importez dans Orchestrator.

L'importation d'un objet en tant qu'élément de ressources dans Orchestrator vous permet d'apporter les modifications nécessaires à l'objet à un emplacement unique et de répercuter automatiquement ces modifications à l'ensemble des workflows utilisant cet élément de ressources.

Vous pouvez organiser les éléments de ressources en dossiers. La taille maximale d'un élément de ressources est de 16 Mo.

Ce chapitre contient les rubriques suivantes :

- [Afficher un élément de ressource](#)
- [Importer un objet externe pour utiliser un élément de ressource](#)
- [Modifier les informations et les droits d'accès d'un élément de ressource](#)
- [Enregistrer un élément de ressource dans un fichier](#)
- [Mettre à jour un élément de ressource](#)
- [Ajouter un élément de ressource à un workflow](#)

## Afficher un élément de ressource

Le client Orchestrator vous permet d'afficher des éléments de ressource, de consulter leur contenu et de découvrir les workflows qui les utilisent.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Ressources**.
- 3 Développer l'affichage de l'arborescence pour naviguer jusqu'à un élément de ressource.



- 4 Cliquez sur un élément de ressource pour afficher les informations le concernant dans le volet de droite.
- 5 Cliquez sur l'onglet **Afficheur** pour afficher le contenu de l'élément de ressource.
- 6 Cliquez avec le bouton droit sur l'élément de ressource, puis sélectionnez **Rechercher les éléments qui utilisent cet élément**.

Orchestrator répertorie tous les workflows qui utilisent cet élément de ressource.

#### Étape suivante

Importer et modifier un élément de ressource.

## Importer un objet externe pour utiliser un élément de ressource

Les workflows peuvent demander à ce que les objets créés en dehors d'Orchestrator soient utilisés en tant qu'attributs. Pour utiliser les objets externes en tant qu'attributs dans les workflows, vous devez les importer dans le serveur Orchestrator en tant qu'éléments de ressource.

#### Conditions préalables

Vérifiez que vous disposez d'un fichier image, d'un script, d'un modèle XML, d'un fichier HTML ou d'un autre type d'objet à importer.

#### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Ressources**.
- 3 Cliquez avec le bouton droit sur un dossier de ressource dans la liste hiérarchique ou à la racine, puis sélectionnez **Nouveau dossier** pour créer un dossier qui servira à stocker l'élément de ressource.
- 4 Cliquez avec le bouton droit sur le dossier dans lequel vous souhaitez importer l'élément de ressource, puis sélectionnez **Importer les ressources**.
- 5 Sélectionnez la ressource à importer, puis cliquez sur **Ouvrir**.

Orchestrator ajoute l'élément de ressource au dossier que vous avez sélectionné.

Vous avez importé un élément de ressource dans le serveur Orchestrator.

#### Étape suivante

Modifiez les informations générales sur l'élément de ressource et définissez les droits d'accès utilisateur.


## Modifier les informations et les droits d'accès d'un élément de ressource

Après avoir importé un objet dans le serveur Orchestrator en tant qu'élément de ressource, vous pouvez modifier les détails et les autorisations de cet élément de ressource.

## Conditions préalables

Vérifiez que vous avez importé une image, un script, un fichier XML ou HTML, ou tout autre type d'objet dans Orchestrator en tant qu'élément de ressource.

## Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Ressources**.
- 3 Cliquez avec le bouton droit sur l'élément de ressource, puis sélectionnez **Modifier**.
- 4 Cliquez sur l'onglet **Général** et définissez le nom, la version et la description de l'élément de ressource.
- 5 Cliquez sur l'onglet **Autorisations**, puis cliquez sur l'icône **Ajouter des droits d'accès** () pour définir les autorisations d'un groupe d'utilisateurs.
- 6 Tapez un nom de groupe d'utilisateurs dans la zone de texte **Filtrer**.
- 7 Sélectionnez un groupe d'utilisateurs, puis cliquez sur **OK**.
- 8 Cliquez avec le bouton droit sur le groupe d'utilisateurs, puis sélectionnez **Ajouter des droits d'accès**.
- 9 Cochez les cases appropriées pour définir le niveau des autorisations de ce groupe d'utilisateurs, puis cliquez sur **OK**.  
  
Les autorisations ne sont pas cumulables. Pour permettre à un utilisateur d'afficher l'élément de ressource, utilisez-le dans ses workflows et modifiez les autorisations. Vous devez cocher toutes les cases.
- 10 Cliquez sur **Enregistrer et fermer** pour quitter l'éditeur.

Vous avez modifié les informations générales sur l'élément de ressource et vous avez défini les droits d'accès de l'utilisateur.

## Étape suivante

Enregistrez l'élément de ressource dans un fichier pour le mettre à jour ou ajoutez-le à un workflow.

## Enregistrer un élément de ressource dans un fichier

Vous pouvez enregistrer un élément de ressource dans un fichier sur votre système local.

L'enregistrement d'un élément de ressource en tant que fichier vous donnera la possibilité de le modifier.

Vous ne pouvez pas modifier un élément de ressource dans le client Orchestrator. Par exemple, si l'élément de ressource est un fichier de configuration XML ou un script, vous devez l'enregistrer en local pour pouvoir le modifier.

### Conditions préalables

Vérifiez que le serveur Orchestrator comporte un élément de ressource que vous pouvez enregistrer dans un fichier.

#### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Ressources**.
- 3 Cliquez avec le bouton droit sur l'élément de ressource, puis sélectionnez **Enregistrer dans un fichier**.
- 4 Apportez les modifications nécessaires au fichier.

Vous avez enregistré un élément de ressource dans un fichier.

#### Étape suivante

Mettez à jour l'élément de ressource dans le serveur Orchestrator.

## Mettre à jour un élément de ressource

Si vous souhaitez mettre à jour un élément de ressource, vous devez l'exporter dans le système de fichiers, modifier le fichier exporté avec un outil adapté, puis mettre à jour l'élément de ressource en important le fichier modifié.

### Conditions préalables

Vérifiez que vous avez importé une image, un script, un fichier XML ou HTML, ou tout autre type d'objet dans Orchestrator en tant qu'élément de ressource.

#### Procédure

- 1 Modifiez le fichier source de l'élément de ressource dans votre système local.
- 2 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 3 Cliquez sur la vue **Ressources**.
- 4 Naviguez dans la liste hiérarchique jusqu'à l'élément de ressource que vous avez mis à jour.
- 5 Cliquez avec le bouton droit sur l'élément de ressource, puis sélectionnez **Mettre à jour la ressource**.
- 6 (Facultatif) Cliquez sur l'onglet **Afficheur** pour vérifier qu'Orchestrator a mis à jour l'élément de ressource.

Vous avez mis à jour un élément de ressource du serveur Orchestrator.

## Ajouter un élément de ressource à un workflow

Les éléments de ressource sont des objets externes que vous pouvez importer dans le serveur Orchestrator afin que les workflows puissent les utiliser en tant qu'attributs lors de leur exécution. Par exemple, un workflow en cours d'exécution peut utiliser un fichier XML importé qui définit une carte pour convertir un type de données en un autre, ou un script déterminant une fonction.

### Conditions préalables

Vérifiez que vous disposez des objets suivants dans votre serveur Orchestrator :

- une image, un script, un fichier XML ou HTML, ou tout autre type d'objet importé dans Orchestrator en tant qu'élément de ressource.
- Workflow nécessitant un élément de ressource en tant qu'attribut.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Conception**.
- 2 Cliquez sur la vue **Workflows**.
- 3 Développez l'affichage de l'arborescence pour naviguer jusqu'au workflow qui doit utiliser l'élément de ressource en tant qu'attribut.
- 4 Cliquez avec le bouton droit sur le workflow et sélectionnez **Modifier**.
- 5 Dans l'onglet **Général**, consultez le volet Attributs et cliquez sur l'icône **Ajouter un attribut** (A+).
- 6 Cliquez sur le nom de l'attribut et tapez un nouveau nom pour cet attribut.
- 7 Cliquez sur **Type** pour définir le type d'attribut.
- 8 Dans la boîte de dialogue **Sélectionnez un type**, tapez **ressource** dans la zone de texte **Filtre** pour rechercher un type d'objet.

Option	Action
Définir un élément de ressource unique en tant qu'attribut	Sélectionnez ResourceElement dans la liste.
Définir un dossier qui contient plusieurs éléments de ressource en tant qu'attribut	Sélectionnez ResourceElementCategory dans la liste.

- 9 Cliquez sur **Valeur** et tapez le nom de l'élément de ressource ou la catégorie d'éléments de ressource dans la zone de texte **Filtre**.
- 10 À partir de la liste proposée, sélectionnez l'élément de ressource ou un dossier comportant les éléments de ressource, puis cliquez sur **Sélectionner**.
- 11 Cliquez sur **Enregistrer et fermer** pour quitter l'éditeur.

Vous avez ajouté un élément de ressource ou un dossier d'éléments de ressource en tant qu'attribut dans le workflow.

## Création de modules

Les modules servent à distribuer le contenu d'un serveur Orchestrator vers un autre. Les modules peuvent contenir des workflows, des actions, des modèles de stratégies, des configurations ou des ressources.

Lorsque vous ajoutez un élément à un module, Orchestrator vérifie les dépendances et ajoute tous les éléments dépendants au module. Par exemple, si vous ajoutez un workflow qui utilise des actions ou d'autres workflows, Orchestrator ajoute ces actions et ces workflows au module.

Lorsque vous importez un module, le serveur compare les versions des différents éléments de son contenu aux éléments locaux correspondants. La comparaison indique les différences de versions entre les éléments locaux et importés. L'administrateur peut décider s'il souhaite importer le module ou sélectionner des éléments spécifiques.

Les modules s'appuient sur la gestion des droits numériques pour contrôler la façon dont le serveur récepteur peut utiliser leur contenu. Orchestrator signe les modules et les chiffre pour protéger les données. Les modules peuvent identifier les utilisateurs qui exportent et redistribuent les éléments grâce aux certificats X509.

Pour plus d'informations sur l'utilisation des modules, reportez-vous à *Utilisation de VMware vRealize Orchestrator Client*.

### ■ Créer un module

Vous pouvez exporter les workflows, les modèles de stratégies, les actions, les références de plug-ins, les ressources et les éléments de configuration dans des stratégies. Tous les éléments implémentés par un élément du module sont ajoutés automatiquement au module afin d'assurer la compatibilité entre les versions. Si vous ne souhaitez pas ajouter les éléments référencés, vous pouvez les supprimer dans l'éditeur de modules.

### ■ Définir les autorisations des utilisateurs sur un module

Vous définissez différents niveaux d'autorisation sur un module pour limiter l'accès que les différents utilisateurs ou groupes d'utilisateurs peuvent avoir au contenu de ce module.

## Créer un module

Vous pouvez exporter les workflows, les modèles de stratégies, les actions, les références de plug-ins, les ressources et les éléments de configuration dans des stratégies. Tous les éléments implémentés par un élément du module sont ajoutés automatiquement au module afin d'assurer la compatibilité entre les

versions. Si vous ne souhaitez pas ajouter les éléments référencés, vous pouvez les supprimer dans l'éditeur de modules.

### Conditions préalables

Vérifiez que le serveur Orchestrator contient des éléments tels que des workflows, des actions et des modèles de stratégies que vous pouvez ajouter à un module.

### Procédure

- 1 Dans le menu déroulant du client Orchestrator, sélectionnez **Administrer**.
- 2 Cliquez sur la vue **Paquets**.
- 3 Cliquez avec le bouton droit dans le volet de gauche et sélectionnez **Ajouter un module**.
- 4 Entrez le nom du nouveau module, puis cliquez sur **OK**.  
La syntaxe des noms de modules est *domain.votre\_société.dossier.nom\_du\_module*.  
Par exemple, *com.vmware.myfolder.mypackage*.
- 5 Cliquez avec le bouton droit sur le module, puis sélectionnez **Modifier**.  
L'éditeur de modules s'ouvre.
- 6 Dans l'onglet **Général** ajoutez une description du module.
- 7 Dans l'onglet **Workflows**, ajoutez des workflows au module.
  - Cliquez sur **Insérer des workflows (rechercher dans liste)** pour rechercher et sélectionner des workflows dans une boîte de dialogue de sélection.
  - Cliquez sur **Insérer des workflows (naviguer dans arborescence)** pour naviguer dans la liste hiérarchique et sélectionner des dossiers de workflows.
- 8 Dans les onglets **Modèles de stratégies**, **Actions**, **Configurations**, **Ressources** et **Plug-ins utilisés**, ajoutez des modèles de stratégie, des actions, des éléments de configuration, des éléments de ressource et des plug-ins au module.
- 9 Cliquez sur **Enregistrer et fermer** pour quitter l'éditeur.

Vous avez créé un module et vous y avez ajouté des éléments.

### Étape suivante

Définissez les autorisations utilisateur pour ce module.

## Définir les autorisations des utilisateurs sur un module

Vous définissez différents niveaux d'autorisation sur un module pour limiter l'accès que les différents utilisateurs ou groupes d'utilisateurs peuvent avoir au contenu de ce module.


Vous pouvez sélectionner les différents utilisateurs ou groupes d'utilisateurs pour qui définir des autorisations parmi les utilisateurs ou groupes d'utilisateurs du serveur LDAP ou vCenter Single Sign-On d'Orchestrator. Orchestrator définit les niveaux d'autorisation que vous pouvez appliquer aux utilisateurs ou aux groupes.

<b>Afficher</b>	L'utilisateur a la possibilité de visualiser les éléments se trouvant dans le module, sans en voir les schémas, ni les scripts.
<b>Inspecter</b>	L'utilisateur a la possibilité de visualiser les éléments se trouvant dans le module, schémas et scripts compris.
<b>Modifier</b>	L'utilisateur a la possibilité de modifier les éléments se trouvant dans le module.
<b>Administrateur</b>	L'utilisateur a la possibilité de définir des autorisations sur les éléments se trouvant dans le module.

### Conditions préalables

Créez un module, ouvrez-le pour modification dans l'éditeur de modules et ajoutez-y les éléments nécessaires.

### Procédure

- 1 Cliquez sur l'onglet **Autorisations** de l'éditeur de modules.
- 2 Cliquez sur l'icône **Ajouter des droits d'accès** () pour définir les autorisations d'un nouvel utilisateur ou groupe d'utilisateurs.
- 3 Recherchez un utilisateur ou groupe d'utilisateurs.  
Les résultats de la recherche montrent l'ensemble des utilisateurs et groupes d'utilisateurs répondant à la recherche.
- 4 Sélectionnez un utilisateur ou groupe d'utilisateurs.
- 5 Cochez les cases qui conviennent pour définir le niveau d'autorisation de cet utilisateur, puis cliquez sur **Sélectionner**.  
Pour autoriser un utilisateur à visualiser les éléments, inspecter le schéma et les scripts, exécuter et modifier les éléments et changer les autorisations, vous devez cocher la totalité des cases.
- 6 Cliquez sur **Enregistrer et fermer** pour quitter l'éditeur.

Vous avez donc créé un module et défini les autorisations appropriées pour les utilisateurs.

# Développement de plug-ins

Orchestrator permet une intégration à des solutions de gestion et d'administration par le biais de son architecture ouverte de plug-ins. Vous utilisez le client Orchestrator pour exécuter et créer des workflows de plug-ins, et accéder à l'API des plug-ins.

Ce chapitre contient les rubriques suivantes :

- [Présentation des plug-ins](#)
- [Contenu et structure d'un plug-in](#)
- [Guide de référence de l'API de plug-in Orchestrator](#)
- [Éléments du fichier de définition du plug-in vso.xml](#)
- [Recommandations pour le développement de plug-ins Orchestrator](#)

## Présentation des plug-ins

Les plug-ins Orchestrator doivent inclure un jeu standard de composants et se conformer à une architecture standard. Le respect de ces pratiques vous aidera à créer des plug-ins pour le plus large éventail possible de technologies externes.

- [Structure d'un plug-in Orchestrator](#)

Les plug-ins Orchestrator possèdent une structure commune composée de différents types de couches servant à implémenter des fonctionnalités spécifiques.
- [Exposer une API externe à Orchestrator](#)

Vous exposez une API provenant d'un produit externe à la plateforme Orchestrator en créant un plug-in Orchestrator. Il vous est possible de créer un plug-in pour toute technologie qui expose une API pouvant être mappée dans les objets JavaScript utilisables par Orchestrator.
- [Composants d'un plug-in](#)

Les plug-ins sont composés d'un jeu standard de composants qui exposent les objets au sein de la technologie en plug-in à la plate-forme Orchestrator
- [Rôle du fichier vso.xml](#)

Vous utilisez le fichier `vso.xml` pour mapper les objets, les classes, les méthodes et les attributs de la technologie en plug-in vers les objets d'inventaire, les types de scripts, les classes de scripts, les méthodes de scripts et les attributs. Le fichier `vso.xml` définit également la configuration et le comportement de départ du plug-in.



- **Rôles de l'adaptateur de plug-in**

L'adaptateur de plug-in constitue le point d'entrée du plug-in dans le serveur Orchestrator. L'adaptateur de plug-in sert de banque de données à la technologie en plug-in du serveur Orchestrator, crée la fabrique de plug-in et gère les événements qui surviennent au sein de la technologie en plug-in.

- **Rôles de la fabrique de plug-in**

La fabrique de plug-in définit la façon dont Orchestrator recherche les objets au sein de la technologie en plug-in et réalise des opérations sur les objets.

- **Rôle des objets de l'outil de recherche**

Les objets de l'outil de recherche identifient et localisent des instances précises des types d'objets gérés de la technologie en plug-in. Orchestrator peut modifier les objets et interagir avec eux dès qu'il les trouve dans la technologie en plug-in. Pour ce faire, il exécute des workflows dans les objets de l'outil de recherche.

- **Rôle des objets de scripts**

Les objets de scripts sont des représentations JavaScript des objets provenant de la technologie en plug-in. Les objets de script des plug-ins apparaissent dans l'API JavaScript d'Orchestrator et vous pouvez les utiliser dans les éléments en script des workflows et des actions.

- **Rôle des gestionnaires d'événements**

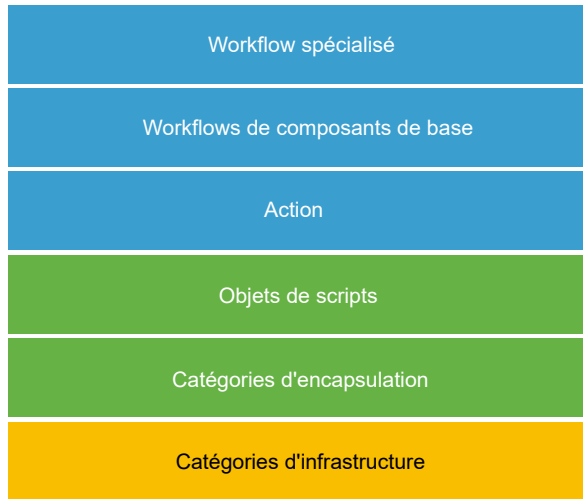
Les événements sont des modifications apportées aux états ou aux attributs des objets qu'Orchestrator trouve dans la technologie en plug-in. Orchestrator surveille les événements en implémentant des gestionnaires d'événement.

## Structure d'un plug-in Orchestrator

Les plug-ins Orchestrator possèdent une structure commune composée de différents types de couches servant à implémenter des fonctionnalités spécifiques.

Les trois couches inférieures d'un plug-in Orchestrator sont les classes d'infrastructure, les catégories d'encapsulation et les objets de script. Elles connectent la technologie en plug-in à Orchestrator.

Dans un plug-in Orchestrator, les parties visibles par l'utilisateur sont les trois couches supérieures, soit les actions, les composants de base et les workflows de spécialisés.

**Figure 6-1. Structure d'un plug-in Orchestrator**

<b>Classes d'infrastructure</b>	Ensemble de classes qui connecte la technologie en plug-in à Orchestrator. Les classes d'infrastructure incluent les classes à implémenter en fonction de la définition du plug-in ; par exemple, la fabrique du plug-in, l'adaptateur du plug-in, etc. Les classes d'infrastructure incluent également les classes qui fournissent des fonctionnalités pour les tâches et les objets courants tels que les systèmes d'aide, la mise en cache, l'inventaire, etc.
<b>Classes d'encapsulation</b>	Ensemble de classes qui adaptent le modèle d'objet de la technologie en plug-in au modèle d'objet que vous souhaitez exposer dans Orchestrator.
<b>Objets de scripts</b>	Types d'objets JavaScript conférant un accès aux classes d'encapsulation, aux méthodes et aux attributs de la technologie en plug-in. Le fichier <code>vso.xml</code> vous permet de définir les classes d'encapsulation, les attributs et les méthodes de la technologie en plug-in qui seront exposés à Orchestrator.
<b>Actions</b>	Ensemble de fonctions JavaScript que vous pouvez utiliser directement dans les workflows et les tâches d'écriture de scripts. Les actions peuvent présenter plusieurs paramètres d'entrée et une seule valeur de retour.
<b>Workflows de composants de base</b>	Ensemble de workflows qui couvrent toutes les fonctionnalités génériques que vous souhaitez transmettre avec le plug-in. Généralement, un workflow de composants de base représente une opération dans l'interface utilisateur de la technologie orchestrée. Les workflows de composants de base peuvent être utilisés directement ou intégrés aux workflows spécialisés.
<b>Workflows spécialisés</b>	Ensemble de workflows qui couvrent une fonctionnalité spécifique du plug-in. Vous pouvez fournir des workflows spécialisés pour répondre à des exigences précises ou pour illustrer des exemples complexes d'utilisation des plug-ins.

## Exposer une API externe à Orchestrator

Vous exposez une API provenant d'un produit externe à la plateforme Orchestrator en créant un plug-in Orchestrator. Il vous est possible de créer un plug-in pour toute technologie qui expose une API pouvant être mappée dans les objets JavaScript utilisables par Orchestrator.

Les plug-ins mappent les objets et méthodes Java vers les objets JavaScript qu'ils ajoutent à l'API de script d'Orchestrator. Si une technologie externe expose une API Java, vous pouvez mapper l'API directement dans JavaScript pour qu'Orchestrator l'utilise dans les workflows et les actions.

Vous pouvez créer des plug-ins pour les applications qui exposent une API dans une langue autre que Java à l'aide de WSDL (Web Service Definition Language), REST (Representational State Transfer) ou un service de messagerie afin d'intégrer l'API exposée aux objets Java. Vous mappez ensuite les objets Java intégrés dans JavaScript pour qu'Orchestrator puisse les utiliser.

La technologie en plug-in est indépendante d'Orchestrator. Vous pouvez créer des plug-ins Orchestrator pour les produits externes même si vous ne pouvez accéder qu'au code binaire et non au code source : par exemple dans les archives Java (fichiers JAR).

## Composants d'un plug-in

Les plug-ins sont composés d'un jeu standard de composants qui exposent les objets au sein de la technologie en plug-in à la plate-forme Orchestrator

Les principaux composants d'un plug-in sont l'adaptateur de plug-in, la fabrique et les implémentations d'événement. Vous mappez les objets et opérations définis dans l'adaptateur, la fabrique et les implémentations d'événement avec les objets Orchestrator au sein d'un fichier de définition XML nommé `vso.xml`. Le fichier `vso.xml` mappe les objets et les fonctions de la technologie en plug-in avec les objets de script JavaScript qui apparaissent dans l'API JavaScript d'Orchestrator. Le fichier `vso.xml` mappe également les types d'objet issus de la technologie en plug-in avec les fonctions de recherche, qui apparaissent dans l'onglet **Inventaire**

Les plug-ins sont constitués des composants suivants.

<b>Le module plug-in</b>	C'est le plug-in en lui-même, tel que défini par un jeu de classes Java, un fichier <code>vso.xml</code> et des modules de workflows et d'actions qui interagissent avec les objets auxquels vous accédez au travers du plug-in. Le module plug-in est obligatoire.
<b>L'adaptateur de plug-in</b>	Il définit l'interface entre la technologie en plug-in et le serveur Orchestrator. L'adaptateur constitue le point d'entrée du plug-in dans la plate-forme Orchestrator. L'adaptateur crée la fabrique de plug-in, gère le chargement et le déchargement du plug-in et gère les événements qui arrivent aux objets de la technologie en plug-in. L'adaptateur de plug-in est obligatoire.
<b>La fabrique de plug-in</b>	Elle définit la façon dont Orchestrator trouve les objets de la technologie en plug-in et effectue des opérations sur ces derniers. L'adaptateur crée une fabrique pour la session client qui s'ouvre entre Orchestrator et une technologie en plug-in. La fabrique vous permet soit de partager une

session entre toutes les connexions client, soit d'ouvrir une session par connexion client. La fabrique de plug-in est obligatoire.

## Configuration

Orchestrator ne définit pas de moyen standard pour le plug-in de stocker sa configuration. Vous pouvez stocker les informations de configuration à l'aide de registres Windows, de fichiers de configuration statique, en conservant les informations dans une base de données ou dans des fichiers XML. Les plug-ins Orchestrator peuvent être configurés par l'exécution de workflows de configuration au sein du client Orchestrator.

## Fonctions de recherche

Les règles d'interaction définissent la façon dont Orchestrator localise et représente les objets de la technologie en plug-in. Les fonctions de recherche retrouvent les objets dans la série d'objets que la technologie en plug-in expose à Orchestrator. Dans le fichier `vso.xml`, vous définissez les relations entre les objets pour pouvoir naviguer dans le réseau d'objets. Orchestrator représente l'objet modèle de la technologie en plug-in dans l'onglet **Inventaire**. Les fonctions de recherche sont obligatoires si vous désirez exposer les objets de la technologie en plug-in à Orchestrator.

## Objets de scripts

Types d'objet JavaScript donnant accès aux objets, attributs et opérations de la technologie en plug-in. Les objets de script définissent la façon dont Orchestrator accède à l'objet modèle de la technologie en plug-in grâce au JavaScript. Vous mappez les classes et les méthodes de la technologie en plug-in aux objets JavaScript dans le fichier `vso.xml`. Vous pouvez accéder aux objets JavaScript dans l'API des scripts Orchestrator et les intégrer aux tâches, actions et workflows en script d'Orchestrator. Les objets de scripts sont obligatoires si vous désirez ajouter des types, classes et méthodes de script à l'API JavaScript d'Orchestrator.

## Inventaire

Les instances d'objets de la technologie en plug-in qu'Orchestrator localise à l'aide des fonctions de recherche apparaissent dans la vue **Inventaire** du client Orchestrator. Vous pouvez réaliser des opérations sur les objets de l'inventaire en exécutant des workflows sur eux. L'inventaire est facultatif. Vous avez la possibilité de créer un plug-in qui ne fait qu'ajouter des types et classes de scripts à l'API JavaScript d'Orchestrator sans exposer la moindre instance d'objet de l'inventaire.

## Événements

Changements dans l'état d'un objet de la technologie en plug-in. Orchestrator est en mesure de guetter de façon passive la survenue d'événements au sein de la technologie en plug-in. Il peut aussi déclencher de façon active des événements au sein de la technologie en plug-in. Les événements sont facultatifs.

## Rôle du fichier `vso.xml`

Vous utilisez le fichier `vso.xml` pour mapper les objets, les classes, les méthodes et les attributs de la technologie en plug-in vers les objets d'inventaire, les types de scripts, les classes de scripts, les

méthodes de scripts et les attributs. Le fichier `vso.xml` définit également la configuration et le comportement de départ du plug-in.

Le fichier `vso.xml` joue les rôles clés suivants.

<b>Comportement de départ et de configuration</b>	Définit la manière selon laquelle le plug-in démarre et localise les implémentations de configuration qu'il doit définir. Charge l'adaptateur de plug-ins.
<b>Objets d'inventaire</b>	Définit les types d'objets auxquels le plug-in accède dans la technologie en plug-in. Les méthodes de recherche de la fabrique de plug-ins localisent les instances de ces objets et les affichent dans l'inventaire Orchestrator.
<b>Types de scripts</b>	Ajoute des types de scripts à l'API JavaScript d'Orchestrator afin de permettre la représentation des différents types d'objets dans l'inventaire. Vous pouvez utiliser ces types de scripts en tant que paramètres d'entrée des workflows.
<b>Classes de scripts</b>	Ajoute des classes à l'API JavaScript d'Orchestrator que vous pouvez utiliser dans les éléments en scripts des workflows, actions, stratégies, etc.
<b>Méthodes de scripts</b>	Ajoute des méthodes à l'API JavaScript d'Orchestrator que vous pouvez utiliser dans les éléments en scripts des workflows, actions, stratégies, etc.
<b>Attributs de scripts</b>	Ajoute les attributs des objets de la technologie en plug-in à l'API JavaScript d'Orchestrator. Vous pouvez utiliser ces attributs dans les éléments en scripts des workflows, actions, stratégies, etc.

## Rôles de l'adaptateur de plug-in

L'adaptateur de plug-in constitue le point d'entrée du plug-in dans le serveur Orchestrator. L'adaptateur de plug-in sert de banque de données à la technologie en plug-in du serveur Orchestrator, crée la fabrique de plug-in et gère les événements qui surviennent au sein de la technologie en plug-in.

Pour créer un adaptateur de plug-in, commencez par créer une classe Java qui met en œuvre l'interface `IPluginAdaptor`.

La classe Adaptateur de plug-in que vous créez gère la fabrique, les événements et les déclencheurs du plug-in au sein de la technologie en plug-in. L'interface `IPluginAdaptor` fournit les méthodes dont vous vous servez pour réaliser ces tâches.

L'adaptateur de plug-in remplit les rôles principaux ci-dessous.

<b>Il crée une fabrique</b>	Le rôle le plus important de l'adaptateur de plug-in est de charger et décharger une instance de la fabrique de plug-in pour chaque connexion d'Orchestrator à la technologie en plug-in. La classe Adaptateur de plug-in
-----------------------------	---

appelle la méthode `IPluginAdaptor.createPluginFactory()` pour créer une instance d'une classe qui met en œuvre l'interface `IPluginFactory`.

## Il gère les événements

L'adaptateur de plug-in constitue l'interface entre le serveur Orchestrator et la technologie en plug-in. Il gère les événements qu'Orchestrator suscite ou guette sur les objets se trouvant dans la technologie en plug-in.

L'adaptateur gère les événements au travers des modules de publication d'événement. Les modules de publication d'événement sont des instances de l'interface `IPluginEventPublisher` que crée l'adaptateur en appelant la méthode `IPluginAdaptor.registerEventPublisher()`. Les modules de publication d'événement définissent des déclencheurs et des indicateurs de stratégie sur les objets se trouvant dans la technologie en plug-in pour permettre à Orchestrator de lancer des actions définies en cas de survenue de certains événements dans l'objet ou si les valeurs de l'objet franchissent certains seuils. De la même façon, vous pouvez définir des instances `PluginTrigger` et `PluginWatcher` qui définissent les événements que les éléments de workflows au long cours En attente d'événement attendent.

## Il définit le nom du plug-in

Indiquez un nom pour le plug-in dans le fichier `vso.xml`. L'adaptateur de plug-in obtient ce nom auprès du fichier `vso.xml` et le publie sur la vue **Inventaire** du client Orchestrator.

## Il installe les licences

Vous pouvez appeler des méthodes pour installer tout fichier de licence requis par la technologie en plug-in pour la mise en œuvre de l'adaptateur.

Pour plus de précisions sur l'interface `IPluginAdaptor`, l'ensemble de ses méthodes et l'ensemble des autres classes de l'API du plug-in, consultez [Guide de référence de l'API de plug-in Orchestrator](#).

## Rôles de la fabrique de plug-in

La fabrique de plug-in définit la façon dont Orchestrator recherche les objets au sein de la technologie en plug-in et réalise des opérations sur les objets.

Pour créer la fabrique de plug-in, il vous faut mettre en œuvre et étendre l'interface `IPluginFactory` depuis l'API de plug-in Orchestrator. La classe Fabrique de plug-in que vous créez définit les fonctions de recherche qu'Orchestrator utilise pour accéder aux objets au sein de la technologie en plug-in. La fabrique permet au serveur Orchestrator de rechercher les objets par leur ID, leur relation à d'autres objets ou de les rechercher par l'intermédiaire d'une chaîne de requête.

La fabrique de plug-in accomplit les tâches principales ci-dessous.

### Elle recherche les objets

Vous pouvez créer des fonctions qui recherchent les objets en fonction de leur nom et de leur type. Vous rechercherez les objets par nom et par type à l'aide de la méthode `IPluginFactory.find()`.

### Elle recherche les objets apparentés à d'autres objets

Vous pouvez créer des fonctions qui recherchent les objets qui sont apparentés à un objet donné par le biais d'un type de relation donné. Vous définirez ces relations dans le fichier `vso.xml`. Vous pouvez également

créer des fonctions de recherche pour les objets enfants dépendants qui sont apparentés à tous les parents par le biais d'un type de relation donné. Pour rechercher tout objet apparenté à un objet parent donné par le biais d'un type de relation donné, mettez en œuvre la méthode `IPluginFactory.findRelation()`. Pour découvrir si au moins un objet enfant existe pour une instance parent, mettez en œuvre la méthode `IPluginFactory.hasChildrenInRelation()`.

### Définir des requêtes pour rechercher des objets selon vos propres critères

Vous pouvez créer des fonctions de recherche d'objets qui mettent en œuvre les règles de requête que vous définissez. Mettez en œuvre la méthode `IPluginFactory.findAll()` pour rechercher tous les objets qui satisfont aux règles de requête que vous définissez lorsque la fabrique appelle cette méthode. Vous obtiendrez les résultats d'une méthode `findAll()` dans un objet de `QueryResult` qui contient une liste de l'ensemble des objets répondant aux règles de requête que vous définissez.

Pour plus d'informations, sur l'interface `IPluginFactory`, l'ensemble de ses méthodes et l'ensemble des autres classes de l'API de plug-in, consultez [Guide de référence de l'API de plug-in Orchestrator](#).

## Rôle des objets de l'outil de recherche

Les objets de l'outil de recherche identifient et localisent des instances précises des types d'objets gérés de la technologie en plug-in. Orchestrator peut modifier les objets et interagir avec eux dès qu'il les trouve dans la technologie en plug-in. Pour ce faire, il exécute des workflows dans les objets de l'outil de recherche.

Chaque instance d'un type d'objet géré donné de la technologie en plug-in doit posséder un identifiant unique afin que les objets de l'outil de recherche Orchestrator puisse la retrouver. La technologie en plug-in qui fournit des identifiants uniques sous forme de chaînes aux instances des objets. Lorsqu'un workflow s'exécute, Orchestrator définit les identifiants uniques des objets qu'il trouve en tant que valeurs d'attribut du workflow. Les workflows nécessitant un objet d'un type précis en tant que paramètre d'entrée s'exécutent sur une instance spécifique de ce type d'objet.

Les objets de l'outil de recherche que les plug-ins ajoutent à l'API JavaScript d'Orchestrator se voient affecter le nom du plug-in en tant que préfixe. Par exemple, le type d'objet géré `VirtualMachine` de l'API vCenter Server apparaît dans Orchestrator sous le type JavaScript `VC:VirtualMachine`.

Par exemple, Orchestrator accède à une instance `VC:VirtualMachine` précise via le plug-in vCenter Server en implémentant un objet de l'outil de recherche qui utilise l'attribut `id` de la machine virtuelle en tant qu'identifiant unique. Vous pouvez transmettre cette instance d'objet aux éléments de workflow sous forme de valeurs d'attribut.

Un plug-in Orchestrator mappe les objets de la technologie en plug-in vers des objets équivalents de l'outil de recherche Orchestrator dans les éléments `<finder>` du fichier `vso.xml`. Les éléments `<finder>` identifient la méthode ou la fonction de la technologie en plug-in qui obtient l'identifiant unique d'une instance ou d'un objet spécifique. Les éléments `<finder>` déterminent également les relations entre les objets, afin de permettre de retrouver des objets en fonction de la manière dont ils sont liés aux autres objets.

Les objets de l'outil de recherche apparaissent dans l'onglet **Inventaire** d'Orchestrator, sous leur plug-in.

## Rôle des objets de scripts

Les objets de scripts sont des représentations JavaScript des objets provenant de la technologie en plug-in. Les objets de script des plug-ins apparaissent dans l'API JavaScript d'Orchestrator et vous pouvez les utiliser dans les éléments en script des workflows et des actions.

Les objets de script des plug-ins apparaissent dans l'API JavaScript d'Orchestrator sous forme de modules, de types et de catégories JavaScript. La plupart des objets de l'outil de recherche sont représentés par un objet de script. Les classes JavaScript peuvent ajouter des méthodes et des attributs à l'API JavaScript d'Orchestrator ; ces méthodes représentent les méthodes et attributs des objets de l'API de la technologie en plug-in. La technologie en plug-in fournit les implémentations des objets, des types, des classes et des méthodes indépendamment d'Orchestrator. Par exemple, le plug-in vCenter Server représente tous les objets de l'API vCenter Server sous forme d'objets JavaScript dans l'API JavaScript d'Orchestrator, avec les représentations de l'ensemble des classes, des méthodes et des attributs définis par l'API vCenter Server. Vous pouvez utiliser les classes de script vCenter Server, ainsi que les méthodes et attributs qu'ils définissent dans les fonctions en script d'Orchestrator.

Par exemple, le type d'objet géré `VirtualMachine` de l'API vCenter Server est retrouvé par l'outil de recherche `VC:VirtualMachine` et apparaît dans l'API JavaScript d'Orchestrator en tant que classe JavaScript `VcVirtualMachine`. La classe JavaScript `VcVirtualMachine` de l'API JavaScript d'Orchestrator définit des méthodes et attributs identiques aux objets gérés `VirtualMachine` de l'API vCenter Server.

Un plug-in Orchestrator mappe les objets, les types, les classes, les attributs et les méthodes de la technologie en plug-in vers des objets, des types, des classes, des attributs et des méthodes JavaScript Orchestrator équivalents dans l'élément `<scripting-objects>` du fichier `vso.xml`.

## Rôle des gestionnaires d'événements

Les événements sont des modifications apportées aux états ou aux attributs des objets qu'Orchestrator trouve dans la technologie en plug-in. Orchestrator surveille les événements en implémentant des gestionnaires d'événement.

Avec les plug-ins Orchestrator, vous pouvez surveiller les événements d'une technologie en plug-in de plusieurs manières. L'API des plug-ins Orchestrator vous permet de créer les types de gestionnaires d'événements suivants pour surveiller les événements d'une technologie en plug-in.

### Écouteurs

Surveillent de manière passive les modifications apportées à l'état des objets de la technologie en plug-in. La technologie en plug-in ou l'implémentation des plug-ins détermine les événements surveillés par les



écouteurs. Les écouteurs ne lancent pas d'événements, mais ils avertissent Orchestrator lorsqu'un événement se produit. Les écouteurs détectent les événements soit en interrogeant la technologie en plug-in, soit en recevant des notifications de la technologie en plug-in. Lorsqu'un événement se produit, les stratégies ou workflows Orchestrator qui attendent l'événement peuvent réagir en lançant des opérations dans le serveur Orchestrator. Les composants des écouteurs sont facultatifs.

### **Stratégies**

Surveillent certains événements dans la technologie en plug-in et lancent des opérations dans le serveur Orchestrator si un événement se produit. Les stratégies peuvent surveiller les déclencheurs de stratégies et les indicateurs de stratégies. Les déclencheurs de stratégie définissent un événement de la technologie en plug-in qui, lorsqu'il se produit, provoque le lancement d'une opération dans le serveur Orchestrator par une stratégie en cours d'exécution. Il peut par exemple s'agir de lancer un workflow. Les indicateurs de stratégies définissent des gammes de valeurs pour les attributs d'un objet de la technologie en plug-in. Lorsque ces valeurs sont dépassées, Orchestrator lance une opération. Les stratégies sont facultatives.

### **Déclencheurs de workflow**

Si un workflow en cours d'exécution comporte un élément En attente d'événement et qu'il atteint cet élément, il interrompt son exécution et attend que l'événement se produise dans la technologie en plug-in. Les déclencheurs de workflows définissent les événements de la technologie en plug-in que les éléments En attente d'événement des workflows doivent attendre. Les déclencheurs de workflows s'enregistrent avec les observateurs. Les déclencheurs de workflows sont facultatifs.

### **Observateurs**

Observent les déclencheurs de workflows sur certains événements de la technologie en plug-in pour le compte de l'élément En attente d'événement d'un workflow. Lorsque l'événement se produit, les observateurs notifient tous les workflows qui attendaient cet événement. Les observateurs sont facultatifs.

## **Contenu et structure d'un plug-in**

Les plug-ins Orchestrator doivent contenir un ensemble normalisé de composants et respecter une structure de fichier normalisée. Pour que le plug-in soit conforme à la structure de fichier normalisée, il doit inclure des dossiers et des fichiers spécifiques.

Pour créer un plug-in Orchestrator, vous devez définir la façon dont Orchestrator accède aux objets et interagit avec eux dans la technologie en plug-in. De même, vous devez mapper tous les objets et toutes les fonctions de la technologie en plug-in vers les objets Orchestrator correspondants dans le fichier `vso.xml`.

Le fichier `vso.xml` doit inclure une référence à chaque type d'objet ou d'opération à exposer à Orchestrator. Chaque objet détecté par le plug-in dans la technologie en plug-in doit disposer d'un identifiant unique que vous indiquez. Vous définissez les noms d'objet dans les éléments `finder` et dans les éléments d'objet du fichier `vso.xml`.

Un plug-in peut prendre la forme d'un fichier d'archive Java standard (JAR) ou d'un fichier ZIP mais, dans tous les cas, le fichier doit être renommé avec une extension `.dar`.

---

**Note** Vous pouvez utiliser le Centre de contrôle Orchestrator pour importer un fichier DAR sur le serveur Orchestrator.

---

- **Définition du mappage des applications dans le fichier `vso.xml`**

Les objets que vous insérez dans le fichier `vso.xml` apparaissent en tant qu'objets de scripts dans l'API de script Orchestrator, ou en tant qu'objets de la fonction de recherche dans l'onglet **Inventaire** d'Orchestrator.

- **Format du fichier de définition du plug-in `vso.xml`**

Le fichier `vso.xml` définit la façon dont le serveur Orchestrator interagit avec la technologie en plug-in. Le fichier `vso.xml` doit inclure une référence à chaque type d'objet ou d'opération à exposer à Orchestrator.

- **Attribution de noms aux objets de plug-in**

Vous devez fournir un identifiant unique pour chaque objet que le plug-in trouve dans la technologie en plug-in. Les noms des objets sont à définir dans les éléments `<finder>` et dans les éléments `<object>` du fichier `vso.xml`.

- **Conventions de nommage des objets des plug-ins**

Vous devez respecter les conventions de nommage des classes Java lorsque vous nommez les objets des plug-ins.

- **Structure de fichiers du plug-in**

Un plug-in doit respecter une structure de fichiers standard et comprendre certains dossiers et fichiers précis. Le plug-in est livré sous la forme d'un fichier archive Java (JAR) ou ZIP standard que vous renommerez avec l'extension `.dar`.

## Définition du mappage des applications dans le fichier `vso.xml`

Les objets que vous insérez dans le fichier `vso.xml` apparaissent en tant qu'objets de scripts dans l'API de script Orchestrator, ou en tant qu'objets de la fonction de recherche dans l'onglet **Inventaire** d'Orchestrator.

Le fichier `vso.xml` fournit les informations suivantes au serveur Orchestrator :

- Version, nom et description du plug-in :
- Se réfère aux catégories de la technologie en plug-in et à l'adaptateur de plug-in associé
- Initialise le plug-in au démarrage du serveur Orchestrator
- Types de scripts afin de permettre la représentation des types d'objets dans la technologie en plug-in

- Les relations entre les types d'objets afin de définir le mode d'affichage des objets dans l'inventaire Orchestrator
- Les catégories de scripts qui mappent les objets et les opérations de la technologie en plug-in vers des types d'objets et de fonctions de l'API JavaScript d'Orchestrator.
- Des énumérations pour définir une liste de valeurs des constantes qui s'appliquent à tous les objets d'un certain type
- Les événements qu'Orchestrator surveille dans la technologie en plug-in

Le fichier `vso.xml` doit être conforme à la définition des schémas XML des plug-ins Orchestrator. Vous pouvez accéder à la définition des schémas sur le site de support de VMware.

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

Pour consulter les descriptions de tous les éléments du fichier `vso.xml`, reportez-vous à [Éléments du fichier de définition du plug-in vso.xml](#).

## Format du fichier de définition du plug-in vso.xml

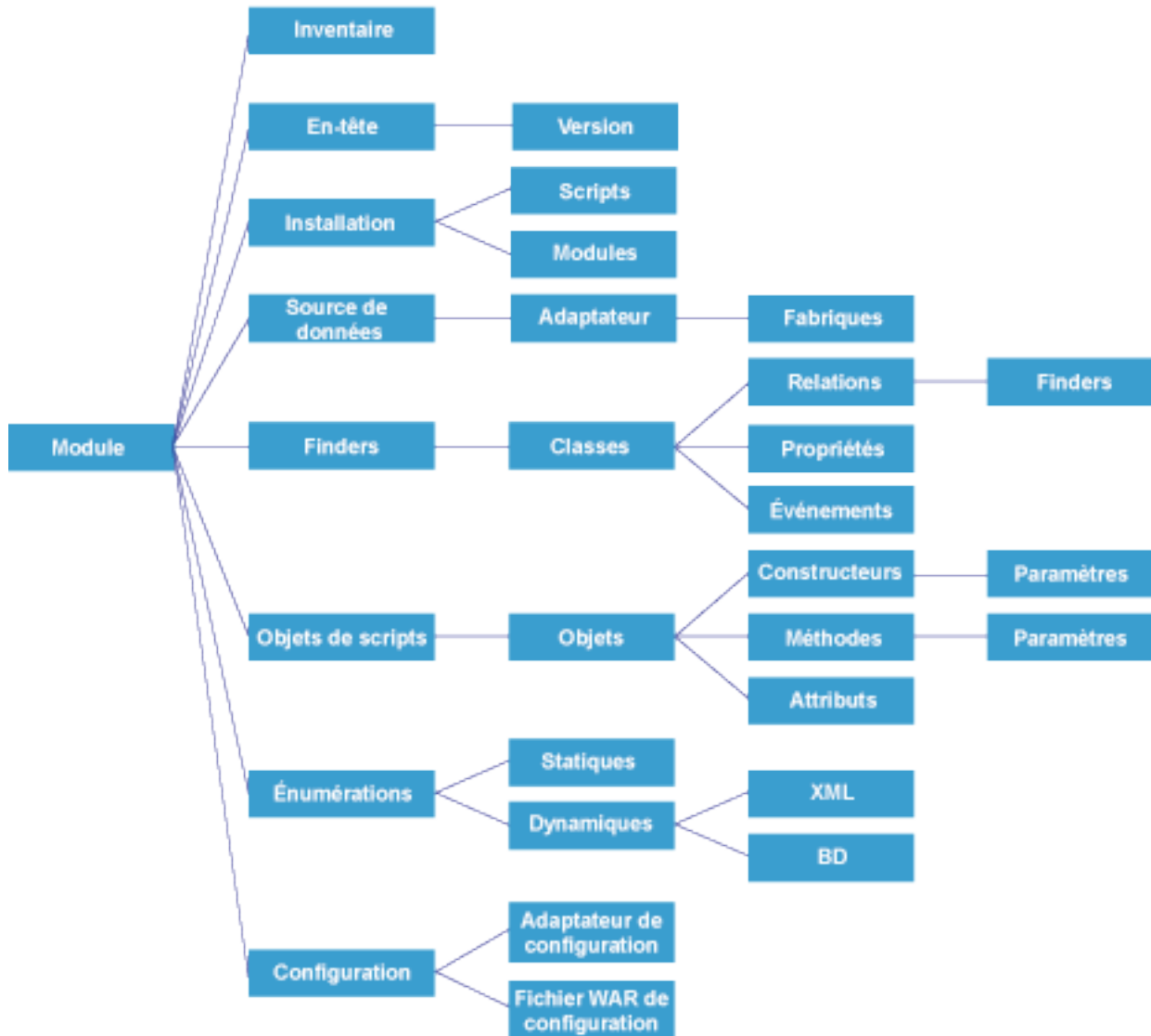
Le fichier `vso.xml` définit la façon dont le serveur Orchestrator interagit avec la technologie en plug-in. Le fichier `vso.xml` doit inclure une référence à chaque type d'objet ou d'opération à exposer à Orchestrator.

Les objets que vous insérez dans le fichier `vso.xml` apparaissent en tant qu'objets de scripts dans l'API de script Orchestrator, ou en tant qu'objets de la fonction de recherche dans l'onglet **Inventaire** d'Orchestrator.

S'inscrivant dans l'architecture ouverte et l'implémentation normalisée des plug-ins, le fichier `vso.xml` doit se conformer à un format standard.

Le diagramme suivant illustre le format du fichier de définition du plug-in `vso.xml` et la manière dont les éléments s'imbriquent les uns avec les autres.

Figure 6-2. Format du fichier de définition du plug-in vso.xml



## Attribution de noms aux objets de plug-in

Vous devez fournir un identifiant unique pour chaque objet que le plug-in trouve dans la technologie en plug-in. Les noms des objets sont à définir dans les éléments `<finder>` et dans les éléments `<object>` du fichier `vso.xml`.

Les opérations de la fonction de recherche que vous définissez dans l'implémentation de la fabrique recherchent les objets dans la technologie en plug-in. Lorsque le plug-in trouve des objets, vous avez la possibilité de les utiliser dans les workflows d'Orchestrator et de les passer d'un élément du workflow à un autre. Les identifiants uniques que vous fournissez pour les objets permettent de passer ces derniers d'un élément à un autre du workflow.

Le serveur Orchestrator ne stocke que le type et l'identifiant de chaque objet qu'il traite et ne stocke aucune information sur l'origine de l'objet ou la façon dont Orchestrator l'a obtenu. Vous devez nommer les objets de façon cohérente dans l'implémentation du plug-in de sorte que vous puissiez suivre à la trace les objets provenant des plug-ins.

Si le serveur Orchestrator s'arrête durant l'exécution des workflows, lorsque vous redémarrerez le serveur, les workflows reprendront à l'élément qui s'exécutait au moment de l'arrêt du serveur. Le workflow utilise les identifiants pour récupérer les objets que l'élément traitait lorsque le serveur s'est arrêté.

## Conventions de nommage des objets des plug-ins

Vous devez respecter les conventions de nommage des classes Java lorsque vous nommez les objets des plug-ins.

---

**Important** Compte tenu de la façon dont le moteur de workflows effectue la sérialisation des données, n'utilisez pas les séquences des chaînes suivantes dans les noms d'objets. L'utilisation de ces séquences de caractères dans les identifiants d'objets ne permet pas au moteur d'analyser correctement les workflows, ce qui peut entraîner un comportement imprévu lors de l'exécution de ces derniers.

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

Suivez les directives suivantes pour nommer les objets et le plug-ins.

- Chaque mot du nom doit commencer par une majuscule.
- N'utilisez pas d'espaces pour séparer les mots.
- Pour les lettres, utilisez uniquement des caractères standard de A à Z et de a à z.
- N'utilisez pas de caractères spéciaux tels que des accents.
- Le premier caractère d'un nom ne doit pas être un chiffre.
- Si possible, ne dépassez pas 10 caractères.

[Tableau 6-1. Règles de nommage des objets des plug-ins](#) indique les règles qui s'appliquent à chaque type d'objets.

**Tableau 6-1. Règles de nommage des objets des plug-ins**

Type d'objet	Règles de nommage
Plug-in	<ul style="list-style-type: none"> <li>■ Définies dans l'élément <code>&lt;module&gt;</code> du fichier <code>vso.xml</code>.</li> <li>■ Doit suivre les conventions de nommage des classes Java.</li> <li>■ Doivent être uniques. Vous ne pouvez pas exécuter deux plug-ins portant le même nom dans un serveur Orchestrator.</li> </ul>
Objet de l'outil de recherche	<ul style="list-style-type: none"> <li>■ Défini dans les éléments <code>&lt;finder&gt;</code> du fichier <code>vso.xml</code>.</li> <li>■ Doit suivre les conventions de nommage des classes Java.</li> <li>■ Doit être unique dans le plug-in.</li> </ul> <p>Orchestrator ajoute le nom du plug-in avec deux points aux noms des objets de l'outil de recherche dans les types d'objets de l'outil de recherche de l'API de script Orchestrator. Par exemple, le type d'objet <code>VirtualMachine</code> du plug-in <code>vCenter Server</code> apparaît dans l'API de script Orchestrator en tant que <code>VC:VirtualMachine</code>.</p>
Objet de script	<ul style="list-style-type: none"> <li>■ Défini dans les éléments <code>&lt;scripting-object&gt;</code> du fichier <code>vso.xml</code>.</li> <li>■ Doit suivre les conventions de nommage des classes Java.</li> <li>■ Doit être unique dans le serveur Orchestrator.</li> <li>■ Pour éviter toute confusion entre les objets de script, les objets de l'outil de recherche ou les objets de script d'autres plug-ins portant le même nom, n'oubliez jamais de placer un préfixe avec le nom du plug-in devant le nom de l'objet de script sans ajouter les deux points. Par exemple, la classe <code>VirtualMachine</code> du plug-in <code>vCenter Server</code> apparaît dans l'API de script Orchestrator en tant que classe <code>VcVirtualMachine</code>.</li> </ul>

## Structure de fichiers du plug-in

Un plug-in doit respecter une structure de fichiers standard et comprendre certains dossiers et fichiers précis. Le plug-in est livré sous la forme d'un fichier archive Java (JAR) ou ZIP standard que vous renommerez avec l'extension `.dar`.

Le contenu du fichier archive DAR doit suivre la structure de dossiers et les conventions de nommage suivantes.

**Tableau 6-2. Structure du fichier archive DAR**

Dossiers	Description
<i>nom_du_plug-in</i> \VS0-INF\	Contient le fichier <code>vso.xml</code> qui définit le mappage des objets présents dans la technologie en plug-in avec les objets Orchestrator.  Le dossier VS0-INF et le fichier <code>vso.xml</code> sont obligatoires.
<i>nom_du_plug-in</i> \lib\	Contient les fichiers JAR qui renferment les binaires de la technologie en plug-in. Contient aussi les fichiers JAR qui renferment les implémentations de l'adaptateur, de la fabrique, des gestionnaires de notifications et d'autres interfaces au sein du plug-in.  Le dossier lib et les fichiers JAR sont obligatoires.
<i>nom_du_plug-in</i> \resources\	Contient les fichiers de ressources que nécessite le plug-in. Le dossier resources peut comprendre les types d'élément suivants : <ul style="list-style-type: none"> <li>■ Des fichiers image, pour représenter les objets du plug-in dans l'onglet <b>Inventaire</b> d'Orchestrator.</li> <li>■ Des scripts, pour définir le comportement à l'initialisation lorsque le plug-in se lance.</li> <li>■ Des modules Orchestrator, qui peuvent contenir des workflows personnalisés et d'autres ressources pour interagir avec les objets auxquels vous accédez à l'aide du plug-in.</li> </ul> <p>Vous pouvez organiser les ressources dans des sous-dossiers. Par exemple, <code>resources\images\</code>, <code>resources\scripts\</code> ou <code>resources\packages\</code>.</p> <p>Le dossier resources est facultatif.</p>

Vous pouvez utiliser le Centre de contrôle Orchestrator pour importer un fichier DAR sur le serveur Orchestrator.

## Guide de référence de l'API de plug-in Orchestrator

L'API de plug-in Orchestrator définit les interfaces et classes à mettre en œuvre et à étendre lorsque vous développez les implémentations de `IPluginAdaptor` et de `IPluginFactory` pour créer un plug-in.

Sauf indication contraire, la totalité des classes est contenue dans le module `ch.dunes.vso.sdk.api`.

### Interface IAop

L'interface `IAop` propose des méthodes pour obtenir et définir des propriétés sur les objets de la technologie en plug-in.

```
public interface IAop
```

L'interface `IAop` définit les méthodes suivantes :

Méthode	Renvois	Description
<code>get(java.lang.String propertyName, java.lang.Object object, java.lang.Object sdkObject)</code>	<code>java.lang.Object</code>	Obtient une propriété à partir d'un objet donné dans le plug-in.
<code>set(java.lang.String propertyName, java.lang.String propertyValue, java.lang.Object object)</code>	<code>Void</code>	Définit une propriété sur un objet donné dans le plug-in.

## Interface IDynamicFinder

L'interface `IDynamicFinder` renvoie l'ID et les propriétés d'une fonction de recherche par la programmation au lieu de définir l'ID et les propriétés dans le fichier `vso.xml`.

L'interface `IDynamicFinder` définit les méthodes suivantes.

Méthode	Renvois	Description
<code>getIdAccessor(java.lang.String type)</code>	<code>java.lang.String</code>	Fournit une expression OGNL pour obtenir l'ID d'un objet par programmation.
<code>getProperties(java.lang.String type)</code>	<code>java.util.List&lt;SDKFinderProperty&gt;</code>	Fournit une liste des propriétés des objets par programmation.

## Interface IPluginAdaptor

L'interface `IPluginAdaptor` est mise en œuvre pour gérer les fabriques, événements et observateurs de plug-in. L'interface `IPluginAdaptor` définit un adaptateur entre un plug-in et le serveur Orchestrator.

Les instances d'`IPluginAdaptor` sont en charge de la gestion des sessions. L'interface `IPluginAdaptor` définit les méthodes suivantes.

Méthode	Renvois	Description
<code>addWatcher(PluginWatcher watcher)</code>	<code>Void</code>	Ajoute un observateur pour surveiller la survenue d'un événement précis.
<code>createPluginFactory(java.lang.String sessionId, java.lang.String username, java.lang.String password, IPluginNotificationHandler notificationHandler)</code>	<code>IPluginFactory</code>	<p>Crée une instance d'<code>IPluginFactory</code>. Le serveur Orchestrator se sert de la fabrique pour obtenir les objets auprès de la technologie en plug-in par leur ID, leur relation avec d'autres objets et ainsi de suite.</p> <p>L'ID de session vous permet d'identifier une session en cours d'exécution. Par exemple, un utilisateur peut très bien se connecter à deux clients Orchestrator différents et exécuter deux sessions simultanément.</p> <p>De la même façon, le démarrage d'un workflow crée une session indépendante du client dans lequel le workflow a démarré. Un workflow continue de s'exécuter même si vous refermez le client Orchestrator.</p>



Méthode	Renvois	Description
<code>installLicenses(PluginLicense[] licenses)</code>	Void	Installe les informations de licence des plug-ins standard fournis par VMware
<code>registerEventPublisher(java.lang.String type, java.lang.String id, IPluginEventPublisher publisher)</code>	Void	Définit des déclencheurs et des indicateurs de stratégie sur un élément de l'inventaire
<code>removeWatcher(java.lang.String watcherId)</code>	Void	Supprime un observateur
<code>setPluginName(java.lang.String pluginName)</code>	Void	Récupère le nom du plug-in dans le fichier <code>vso.xml</code>
<code>setPluginPublisher(IPluginPublisher pluginPublisher)</code>	Void	Définit l'éditeur du plug-in
<code>uninstallPluginFactory(IPluginFactory plugin)</code>	Void	Désinstalle une fabrique de plug-ins.
<code>unregisterEventPublisher(java.lang.String type, java.lang.String id, IPluginEventPublisher publisher)</code>	Void	Supprime les déclencheurs et les indicateurs de stratégie d'un élément de l'inventaire

## Interface IPluginEventPublisher

L'interface `IPluginEventPublisher` publie des déclencheurs et des indicateurs de stratégie sur un bus de notification d'événements pour une surveillance par les stratégies d'Orchestrator.

Vous avez la possibilité de créer des instances d'`IPluginEventPublisher` directement dans l'implémentation de l'adaptateur du plug-in ou les créer dans des classes de générateur d'événement distinctes.

Vous pouvez implémenter l'interface `IPluginEventPublisher` pour publier des événements dans la technologie en plug-in pour le moteur de stratégies Orchestrator. Vous créez des méthodes pour définir des déclencheurs et des indicateurs de stratégies sur des objets dans la technologie en plug-in et des écouteurs d'événements pour écouter les événements de ces objets.

Les stratégies permettent d'implémenter des indicateurs ou des déclencheurs pour surveiller les objets de la technologie en plug-in. Les indicateurs de stratégie surveillent les attributs des objets et insèrent un événement dans le serveur Orchestrator si les valeurs des objets dépassent certaines limites. Les déclencheurs de stratégie surveillent les objets et insèrent un événement dans le serveur Orchestrator si un événement prédéfini se produit sur l'objet. Vous enregistrez les indicateurs et déclencheurs de stratégies avec les instances de `IPluginEventPublisher` afin que les stratégies Orchestrator puissent les surveiller.

L'interface `IPluginEventPublisher` définit les méthodes suivantes.

Type	Renvoie	Description
<code>pushGauge(java.lang.String type, java.lang.String id, java.lang.String gaugeName, java.lang.String deviceName, java.lang.Double gaugeValue)</code>	Void	Publier un indicateur de stratégie pour une surveillance par les stratégies. Adopte les paramètres suivants : <ul style="list-style-type: none"> <li>■ <code>type</code> : type de l'objet à surveiller.</li> <li>■ <code>id</code> : identifiant de l'objet à surveiller.</li> <li>■ <code>gaugeName</code> : nom de cet indicateur de stratégie.</li> <li>■ <code>deviceName</code> : nom du type d'attribut que l'indicateur de stratégie surveille.</li> <li>■ <code>gaugeValue</code> : valeur en attente de laquelle l'indicateur de stratégie surveille l'objet.</li> </ul>
<code>pushTrigger(java.lang.String type, java.lang.String id, java.lang.String triggerName, java.util.Properties additionalProperties)</code>	Void	Publier un déclencheur pour une surveillance par les stratégies. Adopte les paramètres suivants : <ul style="list-style-type: none"> <li>■ <code>type</code> : type de l'objet à surveiller.</li> <li>■ <code>id</code> : identifiant de l'objet à surveiller.</li> <li>■ <code>triggerName</code> : nom de ce déclencheur.</li> <li>■ <code>additionalProperties</code> : toute propriété complémentaire pour la surveillance par le déclencheur.</li> </ul>

## Interface IPluginFactory

L'API IPluginAdaptor renvoie des instances de IPluginFactory. Les instances de IPluginFactory exécutent des commandes dans l'application en plug-in et trouve les objets sur lesquels réaliser les opérations d'Orchestrator.

L'interface IPluginFactory définit le champ suivant :

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

L'interface IPluginFactory définit les méthodes suivantes.

Méthode	Renvoie	Description
<code>executePluginCommand(java.lang.String cmd)</code>	Void	Utiliser le plug-in pour exécuter une commande. VMware recommande de ne pas utiliser cette méthode.
<code>find(java.lang.String type, java.lang.String id)</code>	java.lang.Object	Utiliser le plug-in pour rechercher un objet. Identifier l'objet par son ID et son type.

Méthode	Renvoie	Description
<code>findAll(java.lang.String type, java.lang.String query)</code>	<code>QueryResult</code>	Utiliser le plug-in pour rechercher des objets d'un certain type et répondant à une chaîne de requête. La syntaxe de la requête se définit dans l'implémentation d' <code>IPluginFactory</code> du plug-in. Si vous ne définissez pas la syntaxe de la requête, <code>findAll()</code> renverra tous les objets du type indiqué.
<code>findRelation(java.lang.String parentType, java.lang.String parentId, java.lang.String relationName)</code>	<code>java.util.List</code>	Détermine si un objet possède des enfants.
<code>hasChildrenInRelation(java.lang.String parentType, java.lang.String parentId, java.lang.String relationName)</code>	<code>HasChildrenResult</code>	Recherche tous les enfants d'un parent donné au travers d'une certaine relation.
<code>invalidate(java.lang.String type, java.lang.String id)</code>	<code>Void</code>	Invalider les objets par type et ID.
<code>void invalidateAll()</code>	<code>Void</code>	Invalider tous les objets se trouvant dans le cache.

## Interface IPluginNotificationHandler

L'API `IPluginNotificationHandler` définit des méthodes pour signaler à Orchestrator la survenue de différents types d'événements sur les objets auxquels il accède grâce au plug-in.

L'interface `IPluginNotificationHandler` définit les méthodes suivantes.

Méthode	Renvoie	Description
<code>getSessionID()</code>	<code>java.lang.String</code>	Renvoie l'ID de la session en cours
<code>notifyElementDeleted(java.lang.String type, java.lang.String id)</code>	<code>Void</code>	Signale au système qu'un objet du type et de l'ID donnés a été supprimé
<code>notifyElementInvalidate(java.lang.String type, java.lang.String id)</code>	<code>Void</code>	Signale au système que les relations d'un objet ont changé. Vous pouvez utiliser la méthode <code>notifyElementInvalidate()</code> pour signaler à Orchestrator tous les changements intervenus dans les relations entre objets et pas seulement ceux qui invalident un objet. Par exemple, l'ajout d'un objet enfant à un parent représente un changement dans la relation entre les deux objets.

Méthode	Renvoi	Description
<code>notifyElementUpdated(java.lang.String type, java.lang.String id)</code>	Void	Signale au système que les attributs d'un objet ont été modifiés
<code>notifyMessage(ch.dunes.vso.sdk.api.ErrorLevel severity, java.lang.String type, java.lang.String id, java.lang.String message)</code>	Void	Publie un message d'erreur relatif au module actuel

## Interface IPluginPublisher

L'interface `IPluginPublisher` publie un événement observateur sur un bus de notification d'événements pour la surveillance des éléments de workflow au long cours En attente d'événement.

Lorsqu'un déclencheur de workflows lance un événement dans la technologie en plug-in, l'observateur de plug-ins surveillant ce déclencheur et enregistré auprès d'une instance de `IPluginPublisher` notifie tous les workflows en attente que l'événement a eu lieu.

L'interface `IPluginPublisher` définit la méthode suivante.

Type	Valeur	Description
<code>pushWatcherEvent(java.lang.String id, java.util.Properties properties)</code>	Void	Publier un événement observateur sur un bus de notification d'événements

## Interface WebConfigurationAdaptor

L'interface `WebConfigurationAdaptor` implémente `IConfigurationAdaptor` et définit des méthodes pour localiser et installer une application Web dans l'onglet de configuration d'un plug-in.

**Note** L'interface `WebConfigurationAdaptor` est obsolète depuis Orchestrator 4.1. Pour ajouter une application Web à la configuration, implémentez `IConfigurationAdaptor` et utilisez l'attribut `configuration-war` du fichier `vso.xml` pour identifier l'application Web.

L'interface `WebConfigurationAdaptor` définit les méthodes suivantes.

Méthode	Renvois	Description
<code>getWebAppContext()</code>	String	Localise le fichier WAR de l'application Web pour l'onglet de configuration. Fournit le nom et le chemin d'accès au fichier WAR sous forme de chaîne à partir du répertoire <code>/webapps</code> du fichier DAR.
<code>setWebConfiguration(boolean webConfiguration)</code>	Booléen	Détermine si le contenu de l'onglet de configuration est défini par une application Web.

## Classe PluginTrigger

La classe `PluginTrigger` crée un module déclencheur qui obtient les informations relatives aux objets et événements à surveiller au sein de la technologie en plug-in pour le compte de l'élément de workflow En attente d'événement.

La classe `PluginTrigger` définit des méthodes pour obtenir ou définir le type et le nom de l'objet à surveiller, la nature de l'événement et le délai d'expiration.

Vous créez des implémentations de la classe `PluginTrigger` qui seront exclusivement utilisées par les éléments Attendre événement des workflows. Vous définissez les déclencheurs des stratégies Orchestrator dans les classes qui définissent les événements et implémentent la méthode `IPluginEventPublisher.pushTrigger()`.

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

La classe `PluginTrigger` définit les méthodes suivantes :

Méthode	Renvoie	Description
<code>getModuleName()</code>	<code>java.lang.String</code>	Obtient le nom du module déclencheur.
<code>getProperties()</code>	<code>java.util.Properties</code>	Obtient la liste des propriétés pour le déclencheur.
<code>getSdkId()</code>	<code>java.lang.String</code>	Obtient l'ID de l'objet à surveiller dans la technologie en plug-in.
<code>getSdkType()</code>	<code>java.lang.String</code>	Obtient le type de l'objet à surveiller dans la technologie en plug-in.
<code>getTimeout()</code>	<code>Long</code>	Obtient le délai d'expiration du déclencheur.
<code>setModuleName(java.lang.String moduleName)</code>	<code>Void</code>	Définit le nom du module déclencheur.
<code>setProperties(java.util.Properties properties)</code>	<code>Void</code>	Définit la liste des propriétés pour le déclencheur.
<code>setSdkId(java.lang.String sdkId)</code>	<code>Void</code>	Définit l'ID de l'objet à surveiller dans la technologie en plug-in.
<code>setSdkType(java.lang.String sdkType)</code>	<code>Void</code>	Définit le type de l'objet à surveiller dans la technologie en plug-in.
<code>setTimeout(long timeout)</code>	<code>Void</code>	Définit un délai d'expiration en secondes. Une valeur négative désactivera le délai d'expiration.

## Constructeurs

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

## Classe PluginWatcher

La classe PluginWatcher guette un événement défini de la technologie en plug-in dans le module déclencheur pour le compte de l'élément de workflow au long cours En attente d'événement.

La classe PluginWatcher définit un outil de création que vous pouvez utiliser pour créer des instances d'observation des plug-ins. La classe PluginWatcher définit des méthodes pour obtenir ou définir le nom du déclencheur de workflows à observer et un délai d'expiration.

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

La classe PluginWatcher définit les méthodes suivantes :

Méthode	Renvoie	Description
getId()	java.lang.String	Obtient l'ID du déclencheur
getModuleName()	java.lang.String	Obtient le nom du module déclencheur
getTimeoutDate()	Long	Obtient la date d'expiration du déclencheur
getTrigger()	Void	Obtient un déclencheur
setId(java.lang.String id)	Void	Définit l'ID du déclencheur
setTimeoutDate()	Void	Définit la date d'expiration du déclencheur

## Constructeur

PluginWatcher(PluginTrigger trigger)

## Classe QueryResult

La classe QueryResult contient les résultats d'une requête find effectuée sur les objets auxquels Orchestrator accède grâce aux plug-ins.

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

La valeur de totalCount peut être supérieure au nombre d'éléments que le QueryResult renvoie si le nombre total de résultats trouvés dépasse le nombre de résultats que la requête renvoie. Le nombre de résultats que la requête renvoie est défini dans la syntaxe de requête présente dans le fichier vso.xml.

La classe QueryResult définit les méthodes suivantes :

Méthode	Renvoie	Description
addElement(java.lang.Object element)	Void	Ajoute un élément au QueryResult
addElements(java.util.List elements)	Void	Ajoute une liste d'éléments au QueryResult

Méthode	Renvoie	Description
getElements()	java.util.List	Obtient les éléments auprès de l'application en plug-in
getTotalCount()	Long	Obtient un comptage de tous les éléments disponibles au sein de la technologie en plug-in
isPartialResult()	Booléen	Détermine si le résultat obtenu est final
removeElement(java.lang.Object element)	Void	Supprime un élément de la technologie en plug-in
setElements(java.util.List elements)	Void	Définit les éléments dans la technologie en plug-in
setTotalCount(long totalCount)	Void	Définit le nombre total d'éléments disponibles dans la technologie en plug-in

## Constructeurs

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

## Classe SDKFinderProperty

La classe SDKFinderProperty définit des méthodes pour obtenir et définir les propriétés dans un objet trouvé dans la technologie en plug-in par les objets fonction de recherche d'Orchestrator. La méthode IDynamicFinder.getProperties renvoie des objets SDKFinderProperty.

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

La classe SDKFinderProperty définit les méthodes suivantes :

Méthode	Renvoie	Description
getAttributeName()	java.lang.String	Obtient un nom d'attribut pour l'objet
getBeanProperty()	java.lang.String	Obtient les propriétés auprès d'un Java bean
getDescription()	java.lang.String	Obtient une description pour l'objet
getDisplayName()	java.lang.String	Obtient un nom complet pour l'objet
getPossibleResultType()	java.lang.String	Obtient les types possibles de résultat que la fonction de recherche renvoie
getPropertyAccessor()	java.lang.String	Obtient un accesseur de propriété pour l'objet
getPropertyAccessorTree()	java.lang.Object	Obtient une arborescence d'accesseurs de propriétés pour l'objet
isHidden()	Booléen	Affiche ou masque l'objet

Méthode	Renvoie	Description
<code>isShowInColumn()</code>	Booléen	Affiche ou masque l'objet dans la colonne Base de données
<code>isShowInDescription()</code>	Booléen	Affiche ou masque la description de l'objet
<code>setAttributeName(java.lang.String attributeName)</code>	Void	Définit un nom d'attribut pour l'objet
<code>setBeanProperty(java.lang.String beanProperty)</code>	Void	Définit les propriétés auprès d'un Java bean
<code>setDescription(java.lang.String description)</code>	Void	Définit une description pour l'objet
<code>setDisplayName(java.lang.String displayName)</code>	Void	Définit un nom complet pour l'objet
<code>setHidden(boolean hidden)</code>	Void	Affichent ou masquent l'objet
<code>setPossibleResultType(java.lang.String possibleResultType)</code>	Void	Définit les types possibles de résultat que la fonction de recherche renvoie
<code>setPropertyAccessor(java.lang.String propertyAccessor)</code>	Void	Définit un accesseur de propriété pour l'objet
<code>setPropertyAccessorTree(java.lang.Object propertyAccessorTree)</code>	Void	Définit une arborescence d'accesseurs de propriétés pour l'objet
<code>setShowInColumn(boolean showInTable)</code>	Void	Affichent ou masquent l'objet dans la colonne Base de données
<code>setShowInDescription(boolean showInDescription)</code>	Void	Affichent ou masquent la description de l'objet

## Constructeur

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

## Classe PluginExecutionException

La classe `PluginExecutionException` renvoie un message d'erreur si le plug-in tombe sur une exception lorsqu'il exécute une opération.

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

La classe `PluginExecutionException` hérite des méthodes suivantes de class `java.lang.Throwable` :

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

## Constructeur

`PluginExecutionException(java.lang.String message)`



## Classe PluginOperationException

La classe `PluginOperationException` traite les erreurs rencontrées au cours du fonctionnement du plugin.

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

La classe `PluginOperationException` hérite des méthodes suivantes de class `java.lang.Throwable` :

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

### Constructeur

`PluginOperationException(java.lang.String message)`

## Énumération HasChildrenResult

L'énumération `HasChildrenResult` déclare si un parent donné possède des enfants. La méthode `IPluginFactory.hasChildrenInRelation` renvoie des objets `HasChildrenResult`.

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

L'énumération `HasChildrenResult` définit les constantes suivantes :

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

L'énumération `HasChildrenResult` définit les méthodes suivantes :

Méthode	Renvois	Description
getValue()	int	Renvoie l'une des valeurs suivantes : <div> <div>1</div> <div>Le parent possède des enfants</div> </div> <div> <div>-1</div> <div>Le parent ne possède pas d'enfant</div> </div> <div> <div>0</div> <div>Paramètre inconnu ou invalide</div> </div>
valueOf(java.lang.String name)	static HasChildrenResult	Renvoie une constante d'énumération de ce type avec le nom indiqué. La Chaîne doit correspondre exactement à l'identifiant utilisé pour déclarer une énumération de ce type. Ne pas utiliser d'espaces dans le nom de l'énumération.
values()	static HasChildrenResult[]	Renvoie un tableau comprenant les constantes de ce type d'énumération, dans l'ordre déclaré. Cette méthode peut couvrir les constantes de la manière suivante : <div> <pre>for (HasChildrenResult c : HasChildrenResult.values()) System.out.println(c);</pre> </div>

L'énumération `HasChildrenResult` hérite des méthodes suivantes de class `java.lang.Enum` :

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

## Type d'annotation `ScriptingAttribute`

Le type d'annotation `ScriptingAttribute` annote un attribut issu d'un objet de la technologie en plug-in pour l'utiliser comme propriété dans l'écriture de scripts.

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

Le type d'annotation `ScriptingAttribute` possède la valeur suivante :

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

## Type d'annotation ScriptingFunction

Le type d'annotation `ScriptingFunction` annote une méthode pour l'utiliser comme propriété dans l'écriture de scripts.

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

Le type d'annotation `ScriptingFunction` possède la valeur suivante :

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

## Type d'annotation ScriptingParameter

Le type d'annotation `ScriptingParameter` annote un paramètre pour l'utiliser comme propriété dans l'écriture de scripts.

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

Le type d'annotation `ScriptingParameter` possède la valeur suivante :

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

## Éléments du fichier de définition du plug-in vso.xml

Le fichier `vso.xml` comporte un ensemble de workflows standard. Certains éléments sont obligatoires tandis que d'autres sont facultatifs. Chaque élément présente des attributs qui définissent les valeurs des objets et des opérations que vous mappez vers des objets et opérations Orchestrator.

En outre, les éléments peuvent présenter des éléments enfants ou en être dépourvus. Un élément enfant permet de mieux définir l'élément parent. Le même élément enfant peut apparaître dans plusieurs éléments parents. Par exemple, l'élément `description` est dépourvu d'éléments enfants mais apparaît lui-même en tant qu'élément enfant dans de nombreux éléments parents : `module`, `example`, `trigger`, `gauge`, `finder`, `constructor`, `method`, `object` et `enumeration`.

Chaque définition d'élément qui suit répertorie ses attributs, ses parents et ses enfants.

### Élément module

Un module désigne un jeu d'objets de plug-in à mettre à disposition d'Orchestrator.

Le module contient des informations sur la façon dont les données issues de la technologie en plug-in sont mappées avec les classes Java, sur le contrôle de version, sur la façon de déployer le module et sur la façon dont le plug-in apparaîtra dans l'inventaire d'Orchestrator.

L'élément `<module>` est facultatif. L'élément `<module>` possède les attributs suivants :

Attributs	Valeur	Description
name	Chaîne	Définit le type de tous les éléments <finder> au sein du plug-in. Attribut obligatoire.
version	Numéro	Numéro de version du plug-in, à utiliser au moment de recharger les modules dans une nouvelle version du plug-in. Attribut obligatoire.
build-number	Numéro	Numéro de build du plug-in, à utiliser au moment de recharger les modules dans une nouvelle version du plug-in. Attribut obligatoire.
image	Fichier image	Icône à afficher dans l'inventaire d'Orchestrator. Attribut obligatoire.
display-name	Chaîne	Nom apparaissant dans l'inventaire d'Orchestrator. Attribut facultatif.
interface-mapping-allowed	true ou false	VMware décourage fortement de mapper les interfaces. Attribut facultatif.

**Tableau 6-3. Hiérarchie des éléments**

Élément parent	Éléments enfants
Aucun	<ul style="list-style-type: none"> <li>■ &lt;description&gt;</li> <li>■ &lt;installation&gt;</li> <li>■ &lt;configuration&gt;</li> <li>■ &lt;finder-datasources&gt;</li> <li>■ &lt;inventory&gt;</li> <li>■ &lt;finders&gt;</li> <li>■ &lt;scripting-objects&gt;</li> <li>■ &lt;enumerations&gt;</li> </ul>

## Élément obsolète

Les éléments <description> fournissent des descriptions des éléments de plug-in apparaissant dans la documentation de l'explorateur de l'API.

Vous ajoutez le texte qui apparaît dans la documentation de l'explorateur de l'API entre les balises <description> et </description>.

L'élément <description> est facultatif. L'élément <description> ne dispose d'aucun attribut.

**Tableau 6-4. Hiérarchie des éléments**

Éléments parents	Éléments enfants
■ <module>	Aucun
■ <example>	
■ <trigger>	
■ <gauge>	
■ <finder>	
■ <constructor>	
■ <method>	
■ <object>	
■ <enumeration>	

## Élément obsolète

L'élément <deprecated> indique les objets et les méthodes obsolètes dans la documentation de l'explorateur de l'API.

Vous ajoutez le texte qui apparaît dans la documentation de l'explorateur de l'API entre les balises <deprecated> et </deprecated>.

L'élément <deprecated> est facultatif. L'élément <deprecated> ne dispose d'aucun attribut.

**Tableau 6-5. Hiérarchie des éléments**

Éléments parents	Éléments enfants
■ <method>	Aucun
■ <object>	

## Élément url

L'élément <url> fournit une URL qui pointe sur la documentation externe d'un objet ou d'une énumération.

Vous indiquerez l'URL entre les balises <url> et </url>.

L'élément <url> est facultatif. L'élément <url> ne possède aucun attribut.

**Tableau 6-6. Hiérarchie des éléments**

Éléments parents	Éléments enfants
■ <enumeration>	Aucun
■ <object>	

## Élément installation

L'élément <installation> vous permet d'installer un module ou d'exécuter un script au démarrage du serveur.

L'élément <installation> est facultatif. L'élément <installation> possède les attributs suivants :

Attributs	Valeur	Description
mode	always, never ou version	La définition de la valeur mode occasionne le comportement suivant lorsque le serveur Orchestrator se lance : <ul style="list-style-type: none"> <li>■ L'action s'exécute always</li> <li>■ L'action ne s'exécute never</li> <li>■ L'action s'exécute lorsque le serveur détecte une toute nouvelle version du plug-in</li> </ul> Attribut obligatoire.

Tableau 6-7. Hiérarchie des éléments

Élément parent	Élément enfant
<module>	<action>

## Élément d'action

L'élément <action> indique l'action qui s'exécute au démarrage du serveur Orchestrator.

Les attributs de l'élément <action> fournissent un chemin d'accès au module ou au script Orchestrator qui définit le comportement du plug-in à son démarrage.

L'élément <action> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <action>.

L'élément <action> présente les attributs suivants.

Attributs	Valeur	Description
resource	Chaîne	Chemin d'accès au module ou au script Java à partir de la racine du fichier dar. Attribut obligatoire.
type	install-package ou execute-script	Installe le module Orchestrator indiqué sur le serveur Orchestrator ou exécute le script indiqué. Attribut obligatoire.

Tableau 6-8. Hiérarchie des éléments

Élément parent	Éléments enfants
<installation>	Aucun

## Élément finder-datasource

L'élément <finder-datasources> constitue le conteneur des éléments <finder-datasource>.

L'élément <finder-datasources> est facultatif. L'élément <finder-datasources> ne dispose d'aucun attribut.

Tableau 6-9. Hiérarchie des éléments

Élément parent	Éléments enfants
<module>	<finder-datasource>

## Élément finder-datasource

L'élément `<finder-datasource>` redirige vers le fichier de classe Java de l'implémentation `IPluginAdaptor` que vous créez pour le plug-in.

Vous définissez comment Orchestrator accède aux objets de la technologie en plug-in dans l'élément `<finder-datasource>`. L'élément `<finder-datasource>` identifie la classe Java de l'adaptateur de plug-ins que vous créez. La classe de l'adaptateur de plug-ins instancie la fabrique de plug-ins que vous créez. La fabrique de plug-ins définit les méthodes qui recherchent les objets dans la technologie en plug-in. Vous pouvez définir des délais d'expiration dans l'élément `<finder-datasource>` pour les appels à la méthode de recherche réalisés par la fabrique. Les délais d'expiration varient selon les méthodes de recherche dans l'interface `IPluginFactory`.

L'élément `<finder-datasource>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<finder-datasources>`. L'élément `<finder-datasource>` présente les attributs suivants.

Attributs	Valeur	Description
<code>name</code>	Chaîne	Identifie la source de données dans les attributs <code>datasource</code> de l'élément <code>&lt;finder&gt;</code> . Identique à un <code>id</code> XML. Attribut obligatoire.
<code>adaptor-class</code>	Classe Java	Redirige vers l'implémentation <code>IPluginAdaptor</code> que vous définissez pour créer l'adaptateur de plug-ins, par exemple <code>com.vmware.plugins.sample.Adaptor</code> . Attribut obligatoire.
<code>concurrent-call</code>	<code>true</code> (par défaut) ou <code>false</code>	Permet à plusieurs utilisateurs d'accéder simultanément à l'adaptateur. Vous devez définir <code>concurrent-call</code> sur <code>false</code> si le plug-in ne prend pas en charge les appels simultanés. Attribut facultatif.
<code>invoker-mode</code>	<code>direct</code> (par défaut) ou <code>timeout</code>	Définit un délai d'expiration pour la fonction de recherche. Si elle est définie sur <code>direct</code> , les appels vers la fonction de recherche n'expirent jamais. Si elle est définie sur <code>timeout</code> , le serveur Orchestrator applique le délai d'expiration qui correspond à la méthode de la fonction de recherche. Attribut facultatif.
<code>anonymous-login-mode</code>	<code>never</code> (par défaut) ou <code>always</code>	Transmet ou ne transmet pas le nom d'utilisateur et le mot de passe de l'utilisateur au plug-in. Attribut facultatif.
<code>timeout-fetch-relation</code>	Numéro ; 30 secondes par défaut	S'applique aux appels de <code>findRelation()</code> . Attribut facultatif.
<code>timeout-find-all</code>	Numéro ; 60 secondes par défaut	S'applique aux appels de <code>findAll()</code> . Attribut facultatif.
<code>timeout-find</code>	Numéro ; 60 secondes par défaut	S'applique aux appels de <code>find()</code> . Attribut facultatif.

Attributs	Valeur	Description
timeout-has-children-in-relation	Numéro ; 2 secondes par défaut	S'applique aux appels de <code>findChildrenInRelation()</code> . Attribut facultatif.
timeout-execute-plugin-command	Numéro ; 30 secondes par défaut	S'applique aux appels de <code>executePluginCommand()</code> . Attribut facultatif.

Tableau 6-10. Hiérarchie des éléments

Élément parent	Élément enfant
<finder-datasources>	Aucun

## Élément inventaire

L'élément <inventory> définit la racine de la liste hiérarchique pour le plug-in qui apparaît sur la vue **Inventaire** et les boîtes de dialogue de sélection d'objets du client Orchestrator.

L'élément <inventory> ne représente pas un objet au sein de l'application en plug-in, mais plutôt le plug-in lui-même en tant qu'objet de l'API de scripts d'Orchestrator.

L'élément <inventory> est facultatif. L'élément <inventory> possède l'attribut suivant.

Attributs	Valeur	Description
type	Un type d'objet Orchestrator	Type de l'élément <finder> qui représente la racine de la hiérarchie des objets. Attribut obligatoire.

Tableau 6-11. Hiérarchie des éléments

Élément parent	Éléments enfants
<module>	Aucun

## Élément Fonctions de recherche

L'élément <finders> constitue le conteneur des éléments <finder>.

L'élément <finders> est facultatif. L'élément <finders> ne dispose d'aucun attribut.

Tableau 6-12. Hiérarchie des éléments

Élément parent	Élément enfant
<module>	<finder>

## Élément Fonction de recherche

Dans le client Orchestrator, l'élément <finder> représente un type d'objet trouvé dans le plug-in.



L'élément `<finder>` identifie la catégorie Java qui définit les représentations de l'objet et de la fonction de recherche de l'objet. L'élément `<finder>` définit l'apparence de l'objet dans l'interface du client Orchestrator. Il identifie également l'objet de script que l'API de script Orchestrator définit pour représenter cet objet.

Les fonctions de recherche servent d'interface entre les formats d'objets utilisés par les différents types de technologies en plug-in.

L'élément `<finder>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<finder>`.

L'élément `<finder>` définit les attributs suivant :

Attributs	Valeur	Description
type	Un type d'objet Orchestrator	Type d'objet représenté par la fonction de recherche. Attribut obligatoire.
datasource	Attribut <code>&lt;finder-datasource name&gt;</code>	Identifie la catégorie Java qui définit l'objet en s'appuyant sur la source de données refid. Attribut obligatoire.
dynamic-finder	Méthode Java	Détermine une méthode personnalisée que vous pouvez implémenter dans une instance <code>IDynamicFinder</code> pour renvoyer l'ID et les propriétés d'une fonction de recherche par la programmation au lieu de la définir dans le fichier <code>vso.xml</code> . Attribut facultatif.
hidden	true ou false (par défaut)	Si true, masque la fonction de recherche dans le client Orchestrator. Attribut facultatif.
image	Chemin d'accès à un fichier graphique.	Icône en 16 x 16 pour représenter la fonction de recherche dans les listes hiérarchiques du client Orchestrator. Attribut facultatif.
java-class	Nom de la catégorie Java.	Catégorie Java qui définit l'objet que la fonction de recherche a trouvé et mappé vers un objet de script. Attribut facultatif.
script-object	Attribut <code>&lt;scripting-object type&gt;</code>	Le type <code>&lt;scripting-object&gt;</code> , le cas échéant, vers lequel cette fonction de recherche doit être mappée. Attribut facultatif.

**Tableau 6-13. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Élément propriétés

L'élément <properties> est le conteneur pour les éléments <finder><property>.

L'élément <properties> est facultatif. L'élément <properties> ne dispose d'aucun attribut.

**Tableau 6-14. Hiérarchie des éléments**

Élément parent	Élément enfant
<finder>	<property>

## Élément propriété

L'élément <property> mappe les propriétés de l'objet trouvé avec des propriétés Java ou des appels de méthode.

Vous pouvez faire appel aux méthodes de la classe SDKFinderProperty lorsque vous mettez en œuvre la fabrique de plug-in pour obtenir les propriétés de l'implémentation de fabrique de plug-in à traiter.

Vous pouvez afficher ou masquer les propriétés de l'objet dans les vues du client Orchestrator. Vous pouvez aussi utiliser des énumérations pour définir les propriétés de l'objet.

L'élément <property> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <property>.

L'élément <property> présente les attributs suivants.

Attributs	Valeur	Description
name	Nom de la fonction de recherche	Nom que FinderResult emploie pour stocker l'élément. Attribut obligatoire.
display-name	Nom de la fonction de recherche	Nom complet de la propriété. Attribut facultatif.

Attributs	Valeur	Description
bean-property	Nom de la propriété	Vous vous servirez de l'attribut bean-property pour identifier une propriété à obtenir au moyen des opérations get et set. Si vous identifiez une propriété nommée MyProperty, le plug-in définira les opérations getMyProperty et setMyProperty.  Vous définirez bean-property ou property-accessor, mais pas les deux. Attribut facultatif.
property-accessor	Méthode qui obtient une valeur de propriété auprès d'un objet	L'attribut property-accessor vous permet de définir une expression OGNL pour valider les propriétés d'un objet.  Vous définirez bean-property ou property-accessor, mais pas les deux. Attribut facultatif.
show-in-column	true (par défaut) ou false	Si la valeur est true, cette propriété apparaîtra dans le tableau des résultats du client Orchestrator. Attribut facultatif.
show-in-description	true (par défaut) ou false	Si la valeur est true, cette propriété apparaîtra dans la description de l'objet. Attribut facultatif.
hidden	true ou false (par défaut)	Si la valeur est true, cette propriété sera masquée dans tous les cas. Attribut facultatif.
linked-enumeration	Nom de l'énumération	Lie une propriété fonction de recherche à une énumération. Attribut facultatif.

Tableau 6-15. Hiérarchie des éléments

Élément parent	Éléments enfants
<properties>	Éléments enfants

## Élément relations

L'élément <relations> constitue le conteneur des éléments <finder><relation>.

L'élément <relations> est facultatif. L'élément <relations> ne dispose d'aucun attribut.

Tableau 6-16. Hiérarchie des éléments

Élément parent	Élément enfant
<finder>	<relation>

## Élément relation

L'élément <relation> définit le mode de relation des objets entre eux

C'est dans l'élément <relation> que vous définissez le nom de la relation.

L'élément `<relation>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<relation>`. L'élément `<relation>` présente les attributs suivants.

Attributs	Valeur	Description
name	Nom de la relation	Nom de cette relation. Attribut obligatoire.
type	Type d'objet Orchestrator	Type d'objet apparenté à un autre objet du fait de la relation. Attribut obligatoire.
cardinality	to-one ou to-many	Définit la relation entre les objets comme une relation de un à un ou de un à plusieurs. Attribut facultatif.

Tableau 6-17. Hiérarchie des éléments

Élément parent	Éléments enfants
<code>&lt;relations&gt;</code>	Aucun

## Élément id

L'élément `<id>` définit une méthode d'obtention d'un ID unique pour l'objet que la fonction de recherche identifie.

L'élément `<id>` est facultatif. L'élément `<id>` possède les attributs suivants.

Attributs	Valeur	Description
accessor	Nom de la méthode	L'attribut accessor vous permet de définir une expression OGNL pour valider les propriétés d'un objet. Attribut obligatoire.

Tableau 6-18. Hiérarchie des éléments

Élément parent	Éléments enfants
<code>&lt;finder&gt;</code>	Aucun

## Élément inventaire-enfants

L'élément `<inventory-children>` définit la hiérarchie des listes qui figurent les objets sur la vue **Inventaire** et les boîtes de dialogue de sélection d'objets du client Orchestrator.

L'élément `<inventory-children>` est facultatif. L'élément `<inventory-children>` ne possède aucun attribut.

Tableau 6-19. Hiérarchie des éléments

Élément parent	Élément enfant
<code>&lt;finder&gt;</code>	<code>&lt;relation-link&gt;</code>

## Élément lien-relation

L'élément `<relation-link>` définit les hiérarchies entre objets parent et enfant dans l'onglet **Inventaire**.

L'élément `<relation-link>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<relation-link>`. L'élément `<relation-link>` présente l'attribut suivant.

Type	Valeur	Description
name	Nom de la relation	refid à un nom de relation. Attribut obligatoire.

**Tableau 6-20. Hiérarchie des éléments**

Élément parent	Éléments enfants
<code>&lt;inventory-children&gt;</code>	Aucun

## Élément Événements

L'élément `<events>` constitue le conteneur des éléments `<trigger>` et `<gauge>`.

L'élément `<events>` peut inclure un nombre illimité de déclencheurs ou d'indicateurs.

L'élément `<events>` est facultatif. L'élément `<events>` ne dispose d'aucun attribut.

**Tableau 6-21. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Élément trigger

L'élément `<trigger>` déclare les déclencheurs que vous pouvez utiliser pour cet outil de recherche. Vous devez implémenter les méthodes `registerEventPublisher()` et `unregisterEventPublisher()` de `IPluginAdaptor` pour définir les déclencheurs.

L'élément `<trigger>` est facultatif. L'élément `<trigger>` présente l'attribut suivant.

Type	Valeur	Description
name	Nom du déclencheur	Nom pour ce déclencheur. Attribut obligatoire.

**Tableau 6-22. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Élément trigger-properties

L'élément `<trigger-properties>` constitue le conteneur des éléments `<trigger-property>`.

L'élément `<trigger-properties>` est facultatif. L'élément `<trigger-properties>` ne possède aucun attribut.

**Tableau 6-23. Hiérarchie des éléments**

Élément parent	Élément enfant
<trigger>	<trigger-property>

## Élément trigger-property

L'élément <trigger-property> définit les propriétés qui identifient un objet déclencheur.

L'élément <trigger-property> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <trigger-property>. L'élément <trigger-property> présente les attributs suivants.

Type	Valeur	Description
name	Nom du déclencheur	Nom attribué au déclencheur. Attribut facultatif.
display-name	Nom du déclencheur	Nom qui s'affiche dans le client Orchestrator. Attribut facultatif.
type	Type de déclencheur	Type d'objet définissant le déclencheur. Attribut obligatoire.

**Tableau 6-24. Hiérarchie des éléments**

Élément parent	Éléments enfants
<trigger-properties>	Aucun

## Élément Indicateur

L'élément <gauge> définit les indicateurs que vous pouvez utiliser pour cette fonction de recherche. Vous devez implémenter les méthodes `registerEventPublisher()` et `unregisterEventPublisher()` de `IPluginAdaptor` pour définir les indicateurs.

L'élément <gauge> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <gauge>.

L'élément <gauge> présente les attributs suivants.

Type	Valeur	Description
name	Nom de l'indicateur	Un nom pour l'indicateur. Attribut obligatoire.
min-value	Nombre	Seuil minimal. Attribut facultatif.
max-value	Nombre	Seuil maximal. Attribut facultatif.
unit	Type d'objet	Type d'objet définissant l'indicateur. Attribut obligatoire.
format	Chaîne	Format de la valeur surveillée. Attribut facultatif.

**Tableau 6-25. Hiérarchie des éléments**

Élément parent	Élément enfant
<events>	<description>

## Élément scripting-objects

L'élément <scripting-objects> constitue le conteneur des éléments <object>.

L'élément <scripting-objects> est facultatif. L'élément <scripting-objects> ne possède aucun attribut.

**Tableau 6-26. Hiérarchie des éléments**

Élément parent	Élément enfant
<module>	<object>

## Élément objet

L'élément <object> mappe les constructeurs, attributs et méthodes de la technologie en plug-in aux types d'objet JavaScript que l'API de scripts d'Orchestrator expose.

Pour les conventions de nommage des objets, consultez [Attribution de noms aux objets de plug-in](#).

L'élément <object> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <object>.

L'élément <object> présente les attributs suivants.

Type	Valeur	Description
script-name	Nom JavaScript	Nom de la classe de scripts. Doivent être uniques au niveau global. Attribut obligatoire.
java-class	Classe Java	Classe Java encapsulée par cette classe JavaScript. Attribut obligatoire.
create	true (par défaut) ou false	Si true, vous pouvez créer une nouvelle instance de cette classe. Attribut facultatif.
strict	true ou false (par défaut)	Si true, vous ne pouvez appeler que les méthodes que vous annotez ou déclarez dans le fichier vso.xml. Attribut facultatif.
is-deprecated	true ou false (par défaut)	Si true, l'objet est mappé avec une classe obsolète. Attribut facultatif.
since-version	String	Version depuis l'obsolescence de la classe Java. Attribut facultatif.

**Tableau 6-27. Hiérarchie des éléments**

Élément parent	Éléments enfants
<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>

## Élément constructeurs

L'élément <constructors> constitue le conteneur des éléments <object><constructor>.

L'élément <constructors> est facultatif. L'élément <constructors> ne dispose d'aucun attribut.

**Tableau 6-28. Hiérarchie des éléments**

Élément parent	Élément enfant
<object>	<constructor>

## Élément constructeur

L'élément <constructor> définit une méthode constructeur. La méthode <constructor> produit de la documentation dans l'explorateur d'API.

L'élément <constructor> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <constructor>. L'élément <constructor> ne dispose d'aucun attribut.

**Tableau 6-29. Hiérarchie des éléments**

Élément parent	Éléments enfants
<constructors>	<ul style="list-style-type: none"> <li>■ &lt;description&gt;</li> <li>■ &lt;parameters&gt;</li> </ul>

## Élément Paramètres du constructeur

L'élément <parameters> constitue le conteneur des éléments <constructor><parameter>.

L'élément <parameters> est facultatif. L'élément <parameters> ne dispose d'aucun attribut.

**Tableau 6-30. Hiérarchie des éléments**

Élément parent	Élément enfant
<constructor>	<parameter>

## Élément Paramètre du constructeur

L'élément <parameter> définit les paramètres du constructeur.



L'élément `<parameter>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<parameter>`. L'élément `<parameter>` présente les attributs suivants.

Type	Valeur	Description
name	String	Nom du paramètre à utiliser dans la documentation sur l'API. Attribut obligatoire.
type	Type de paramètre Orchestrator	Type de paramètre à utiliser dans la documentation sur l'API. Attribut obligatoire.
is-optional	true ou false	Si true, la valeur peut être Null. Attribut facultatif.
since-version	String	Version de la méthode. Attribut facultatif.

**Tableau 6-31. Hiérarchie des éléments**

Élément parent	Éléments enfants
<code>&lt;parameters&gt;</code>	Aucun

## Élément attributs

L'élément `<attributes>` constitue le conteneur des éléments `<object><attribute>`.

L'élément `<attributes>` est facultatif. L'élément `<attributes>` ne dispose d'aucun attribut.

**Tableau 6-32. Hiérarchie des éléments**

Élément parent	Élément enfant
<code>&lt;object&gt;</code>	<code>&lt;attribute&gt;</code>

## Élément attribut

L'élément `<attribute>` mappe les attributs d'une catégorie Java issue de la technologie en plug-in aux attributs JavaScript que le moteur JavaScript d'Orchestrator met à disposition.

L'élément `<attribute>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<attribute>`. L'élément `<attribute>` présente les attributs suivants.

Type	Valeur	Description
java-name	Attribut Java	Nom de l'attribut Java. Attribut obligatoire.
script-name	Objet JavaScript	Nom de l'objet JavaScript correspondant. Attribut obligatoire.

Type	Valeur	Description
return-type	String	Type d'objet que cet attribut renvoie. Indiqué dans la documentation de l'explorateur d'API. Attribut facultatif.  <b>Note</b> Si le type de retour JavaScript est Properties, les implémentations Java sous-jacentes prises en charge seront java.util.HashMap et java.util.Hashtable.
read-only	true OU false	Si true, vous ne pouvez pas modifier cet attribut. Attribut facultatif.
is-optional	true OU false	Si true, ce champ peut être nul. Attribut facultatif.
show-in-api	true OU false	Si false, cet attribut ne figure pas dans la documentation de l'API. Attribut facultatif.
is-deprecated	true OU false	Si true, l'objet mappe un attribut obsolète. Attribut facultatif.
since-version	Numéro	Version à laquelle l'attribut est devenu obsolète. Attribut facultatif.

**Tableau 6-33. Hiérarchie des éléments**

Élément parent	Éléments enfants
<attributes>	Aucun

## Élément méthodes

L'élément <methods> constitue le conteneur des éléments <object><method>.

L'élément <methods> est facultatif. L'élément <methods> ne dispose d'aucun attribut.

**Tableau 6-34. Hiérarchie des éléments**

Élément parent	Élément enfant
<object>	<method>

## Élément méthode

L'élément <method> mappe une méthode Java issue de la technologie en plug-in à une méthode JavaScript que le moteur JavaScript d'Orchestrator expose.

L'élément <method> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <method>.

L'élément <method> présente les attributs suivants.

Type	Valeur	Description
java-name	méthode Java	Nom de la signature de méthode Java avec les types d'argument entre parenthèses, par exemple : <code>getVms(DataStore)</code> . Attribut obligatoire.
script-name	méthode JavaScript	Nom de la méthode JavaScript correspondante. Attribut obligatoire.
return-type	Type d'objet en Java	Le type que cette méthode obtient. Attribut facultatif.  <b>Note</b> Si le type de retour JavaScript est <code>Properties</code> , les implémentations Java sous-jacentes prises en charge seront <code>java.util.HashMap</code> et <code>java.util.Hashtable</code> .
static	true OU false	Si true, cette méthode est statique. Attribut facultatif.
show-in-api	true OU false	Si false, cette méthode ne figure pas dans la documentation des API. Attribut facultatif.
is-deprecated	true OU false	Si true, l'objet mappe une méthode obsolète. Attribut facultatif.
since-version	Numéro	Version à laquelle la méthode est devenue obsolète. Attribut facultatif.

Tableau 6-35. Hiérarchie des éléments

Élément parent	Éléments enfants
<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>

## Exemple d'élément

L'élément <example> vous permet d'ajouter des exemples de codes aux méthodes JavaScript qui apparaissent dans la documentation de l'explorateur de l'API.

L'élément <example> est facultatif. L'élément <example> ne dispose d'aucun attribut.

Tableau 6-36. Hiérarchie des éléments

Élément parent	Éléments enfants
<method>	<ul style="list-style-type: none"> <li>■ &lt;code&gt;</li> <li>■ &lt;description&gt;</li> </ul>

## Élément de code

L'élément `<code>` fournit un exemple de code qui apparaît dans la documentation de l'explorateur de l'API.

Vous indiquez l'exemple de code entre les balises `<code>` et `</code>`. L'élément `<code>` est facultatif. L'élément `<code>` ne dispose d'aucun attribut.

**Tableau 6-37. Hiérarchie des éléments**

Élément parent	Éléments enfants
<code>&lt;example&gt;</code>	Aucun

## Élément paramètres de la méthode

L'élément `<parameters>` constitue le conteneur des éléments `<method>``<parameter>`.

L'élément `<parameters>` est facultatif. L'élément `<parameters>` ne dispose d'aucun attribut.

**Tableau 6-38.**

Élément parent	Élément enfant
<code>&lt;method&gt;</code>	<code>&lt;parameter&gt;</code>

## Élément paramètre de la méthode

L'élément `<parameter>` définit les paramètres d'entrée de la méthode.

L'élément `<parameter>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<parameter>`. L'élément `<parameter>` présente les attributs suivants.

Type	Valeur	Description
name	String	Nom du paramètre. Attribut obligatoire.
type	Type de paramètre Orchestrator	Type de paramètre. Attribut obligatoire.
is-optional	true ou false	Si true, la valeur peut être Null. Attribut facultatif.
since-version	String	Version de la méthode. Attribut facultatif.

**Tableau 6-39. Hiérarchie des éléments**

Élément parent	Élément enfant
<code>&lt;parameters&gt;</code>	Aucun

## Élément singleton

L'élément `<singleton>` crée un objet de script JavaScript en tant qu'instance singleton.

Les objets singleton se comportent de la même façon que les catégories Java statiques. Les objets singleton définissent les objets génériques que le plug-in doit utiliser plutôt que des instances spécifiques d'objets auxquels Orchestrator accède dans la technologie en plug-in. Par exemple, vous pouvez utiliser un objet singleton pour établir la connexion à la technologie en plug-in.

L'élément `<singleton>` est facultatif. L'élément `<singleton>` présente les attributs suivants.

Type	Valeur	Description
script-name	Objet JavaScript	Nom de l'objet JavaScript correspondant. Attribut obligatoire.
datasource	Objet Java	Objet Java source pour cet objet JavaScript. Attribut obligatoire.

**Tableau 6-40. Hiérarchie des éléments**

Élément parent	Élément enfant
<code>&lt;object&gt;</code>	Aucun

## Élément Énumérations

L'élément `<enumerations>` constitue le conteneur des éléments `<enumeration>`.

L'élément `<enumerations>` est facultatif. L'élément `<enumerations>` ne dispose d'aucun attribut.

**Tableau 6-41. Hiérarchie des éléments**

Élément parent	Élément enfant
<code>&lt;module&gt;</code>	<code>&lt;enumeration&gt;</code>

## Élément Énumération

L'élément `<enumeration>` définit les valeurs communes qui s'appliquent à tous les objets d'un certain type.

Si tous les objets d'un certain type ont besoin d'un certain attribut et que la plage de valeur de cet attribut est limitée, vous pouvez définir les valeurs en tant qu'entrées d'énumération. Par exemple, si un type d'objet nécessite un attribut `color` et que les seules couleurs disponibles sont le rouge, le bleu et le vert, vous pouvez définir trois entrées d'énumération pour déterminer ces trois valeurs de couleur. Vous devez définir ces entrées en tant qu'éléments enfants de l'élément Énumération.

L'élément `<enumeration>` est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments `<enumeration>`. L'élément `<enumeration>` présente l'attribut suivant.

Type	Valeur	Description
type	Type d'objet Orchestrator	Type d'énumération. Attribut obligatoire.

Tableau 6-42. Hiérarchie des éléments

Élément parent	Éléments enfants
<enumerations>	<ul style="list-style-type: none"> <li>■ &lt;url&gt;</li> <li>■ &lt;description&gt;</li> <li>■ &lt;entries&gt;</li> </ul>

## Élément Entrées

L'élément <entries> constitue le conteneur des éléments <enumeration><entry>.

L'élément <entries> est facultatif. L'élément <entries> ne dispose d'aucun attribut.

Tableau 6-43. Hiérarchie des éléments

Élément parent	Élément enfant
<enumeration>	<entry>

## Élément Entrée

L'élément <entry> fournit les valeurs des attributs d'énumération.

L'élément <entry> est facultatif. Un plug-in peut disposer d'un nombre illimité d'éléments <entry>.

L'élément <entry> présente les attributs suivants.

Type	Valeur	Description
id	Texte	Identifiant utilisé par les objets pour définir l'entrée d'énumération en tant qu'Attribut. Attribut obligatoire.
name	Texte	Nom de l'entrée. Attribut obligatoire.

Tableau 6-44. Hiérarchie des éléments

Élément parent	Éléments enfants
<entries>	Aucun

## Recommandations pour le développement de plug-ins Orchestrator

Vous pouvez améliorer certains aspects des plug-ins Orchestrator que vous développez si vous comprenez la structure et le contenu des plug-ins, et que vous savez comment éviter des problèmes spécifiques.

### ■ Méthodes de création de plug-ins Orchestrator

Vous pouvez vous appuyer sur différentes méthodes pour créer vos plug-ins Orchestrator. Vous pouvez commencer par créer un plug-in couche par couche ou vous pouvez créer toutes les couches du plug-in en même temps.

## ■ Types des plug-ins Orchestrator

L'utilisation de plug-ins vous permet d'intégrer à Orchestrator des bibliothèques généralistes, des utilitaires tels que XML ou SSH, ou des systèmes complets comme vCloud Director. Selon la technologie que vous intégrez à Orchestrator, les plug-ins peuvent être classés en plug-ins pour services, plug-ins généralistes et plug-ins pour systèmes.

## ■ Implémentation des plug-ins

Vous pouvez utiliser certaines pratiques et techniques utiles pour structurer vos plug-ins, implémenter les catégories Java et les objets JavaScript demandés, développer les workflows et les actions des plug-ins et rédiger la présentation d'un workflow.

## ■ Recommandations pour le développement de plug-ins Orchestrator

Lors du développement des différents composants de vos plug-ins Orchestrator, le respect de certaines pratiques vous permettra d'améliorer la qualité de ces plug-ins.

## ■ Documentation des chaînes et des API de l'interface utilisateur du plug-in

Veillez à suivre les règles validées de style et de format lorsque vous écrivez des chaînes d'interface utilisateur pour les plug-ins Orchestrator et la documentation de l'API associée.

# Méthodes de création de plug-ins Orchestrator

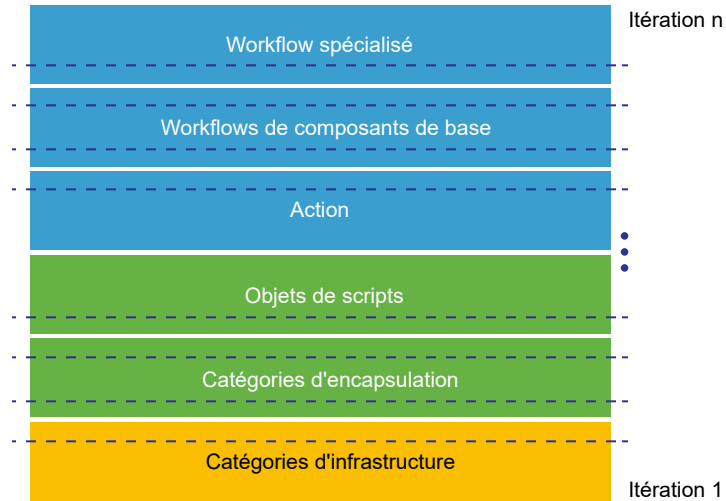
Vous pouvez vous appuyer sur différentes méthodes pour créer vos plug-ins Orchestrator. Vous pouvez commencer par créer un plug-in couche par couche ou vous pouvez créer toutes les couches du plug-in en même temps.

Pour des informations détaillées sur les couches de plug-ins, reportez-vous à [Structure d'un plug-in Orchestrator](#).

## Développement de plug-ins ascendant

Un plug-in peut être développé couche par couche dans le cadre d'une approche ascendante.

L'approche de développement ascendant permet de créer le plug-in couche par couche en commençant par les couches inférieures et en poursuivant par les couches supérieures. Lorsque cette approche est combinée à une approche interactive et itérative, une partie ou l'ensemble de chaque couche est fournie pour chaque itération. Au bout de N itérations, le plug-in est complètement terminé.

**Figure 6-3. Développement de plug-ins ascendant**

L'avantage du développement de plug-ins ascendant est qu'il est concentré sur chaque couche, l'une après l'autre.

Veuillez tenir compte des inconvénients suivants de l'approche de développement ascendant des plug-ins.

- La progression du développement de plug-ins est difficile à évaluer jusqu'à ce que certaines insertions soient terminées.
- Il n'est pas parfaitement compatible avec les pratiques de développement Agile.

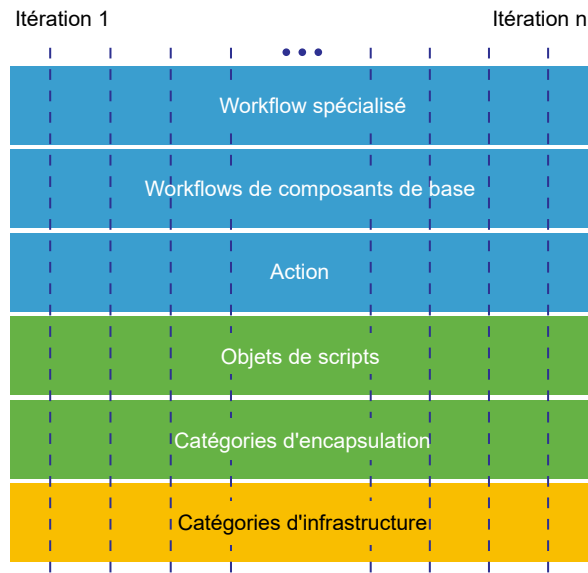
Le processus de développement ascendant est considéré comme satisfaisant pour les petits plug-ins, avec un ensemble inexistant ou limité de catégories d'encapsulation, d'objets de script, d'actions ou de workflows.

## Développement de plug-ins de haut en bas

Un plug-in peut être créé par un fractionnement de haut en bas de fonctionnalités, dans une approche de développement de haut en bas.

Lorsque l'approche de haut en bas est associée à un processus de développement Agile, les nouvelles fonctionnalités sont incluses dans chaque itération. Ainsi, au bout de l'itération N, le plug-in est complètement implémenté.



**Figure 6-4. Développement de plug-ins de haut en bas**

L'approche de développement de plug-ins de haut en bas présente les avantages suivants.

- La progression du développement du plug-in est facile à observer dès la première itération, car les nouvelles fonctionnalités sont implémentées pour chaque itération et le plug-in peut être lancé et utilisé après chaque itération.
- Cette approche par fractions verticales des fonctionnalités permet d'observer clairement le degré de réussite et la définition de ce qui a été accompli, et facilite la communication entre les développeurs, le service de gestion des produits et les ingénieurs de l'assurance qualité (QA).
- Permet aux ingénieurs QA de lancer les tests et l'automatisation dès le début du processus de développement. Une telle approche permet la formulation de commentaires pertinents et réduit le temps de mise à exécution du projet.

L'inconvénient de l'approche du développement de plug-ins de haut en bas est que le développement a lieu sur plusieurs couches différentes en même temps.

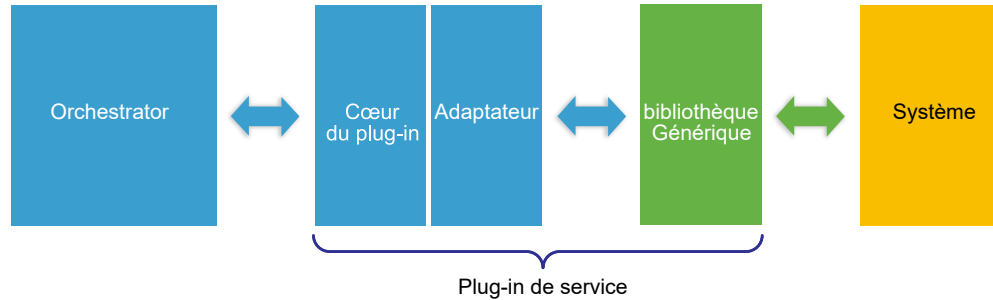
Il est conseillé d'appliquer le processus de développement de plug-ins de haut en bas pour la plupart des plug-ins. Ce processus convient aux plug-ins présentant des exigences de dynamique.

## Types des plug-ins Orchestrator

L'utilisation de plug-ins vous permet d'intégrer à Orchestrator des bibliothèques généralistes, des utilitaires tels que XML ou SSH, ou des systèmes complets comme vCloud Director. Selon la technologie que vous intégrez à Orchestrator, les plug-ins peuvent être classés en plug-ins pour services, plug-ins généralistes et plug-ins pour systèmes.

### Plug-ins pour services

Les plug-ins pour services, encore appelés plug-ins généralistes, confèrent des fonctionnalités qui peuvent être considérées comme des services dans Orchestrator.

**Figure 6-5. Architecture des plug-ins pour services**

Les plug-ins pour services exposent des bibliothèques ou utilitaires génériques à Orchestrator comme XML, SSH ou SOAP. Par exemple, les plug-ins suivants disponibles dans Orchestrator sont des plug-ins pour services.

<b>Plug-in JDBC</b>	Vous permet d'utiliser n'importe quelle base de données d'un workflow.
<b>Plug-in Mail</b>	Vous permet d'envoyer des e-mails dans un workflow.
<b>Plug-in SSH</b>	Vous permet d'ouvrir les connexions SSH et d'exécuter les commandes dans un workflow.
<b>Plug-in XML</b>	Vous permet de gérer les documents XML dans un workflow.

Les plug-ins pour services présentent les caractéristiques suivantes.

<b>Complexité</b>	Les plug-ins pour services présentent des niveaux de complexité de faibles à intermédiaires. Les plug-ins pour services exposent une bibliothèque spécifique ou une partie de bibliothèque dans Orchestrator afin de mettre à disposition une fonctionnalité précise. Par exemple, le plug-in XML ajoute l'implémentation d'un analyseur XML de modèle d'objet de documents (DOM) à l'API JavaScript d'Orchestrator.
<b>Taille</b>	Les plug-ins pour services sont relativement petits. Ils nécessitent le même ensemble de classes de base que les autres plug-ins, ainsi que d'autres classes qui proposent de nouveaux objets de script permettant d'ajouter de nouvelles fonctionnalités.
<b>Inventaire</b>	Les plug-ins pour services peuvent fonctionner avec un inventaire restreint d'objets ou sans inventaire du tout. Les plug-ins pour services présentent un modèle d'objet générique et peu volumineux ; ils ne sont donc pas obligés de faire apparaître ce modèle dans l'inventaire Orchestrator.

## Plug-ins pour systèmes

Les plug-ins pour systèmes connectent le moteur de workflows d'Orchestrator à un système externe afin que vous puissiez orchestrer ce système externe.

Exemples de plug-ins pour systèmes.

<b>Plug-in vCenter Server</b>	Vous permet de gérer les instances vCenter Server à l'aide de workflows.
<b>Plug-in vCloud Director</b>	Vous permet d'interagir avec l'installation vCloud Director au sein d'un workflow.
<b>Plug-in Cisco UCSM</b>	Vous permet d'interagir avec les entités Cisco au sein d'un workflow.

Principales caractéristiques des plug-ins pour systèmes.

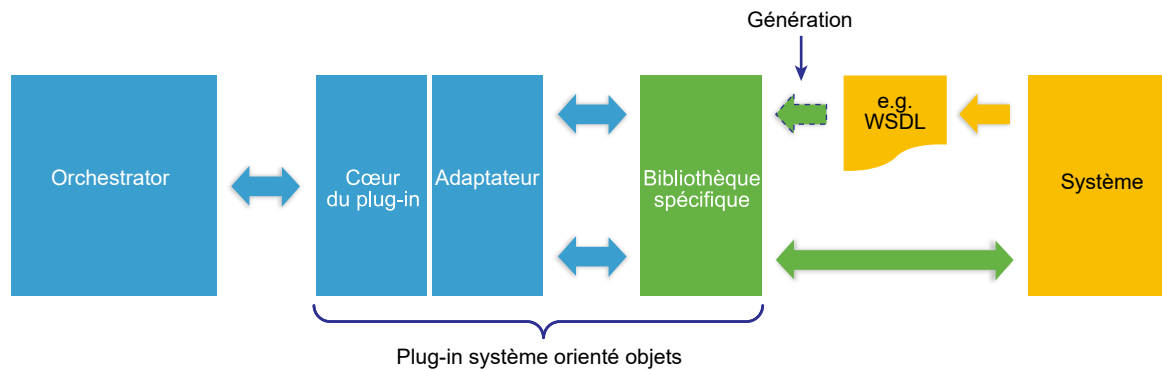
<b>Complexité</b>	Les plug-ins pour systèmes présentent un degré de complexité supérieur à celui des plug-ins généralistes, car les technologies qu'ils exposent sont relativement complexes. Les plug-ins pour systèmes doivent représenter tous les éléments du système externe dans Orchestrator afin de pouvoir interagir et mettre leurs fonctionnalités à disposition dans Orchestrator. Si le système externe fournit un mécanisme d'intégration, vous pouvez l'utiliser afin d'exposer plus facilement les fonctionnalités du système dans Orchestrator. Cependant, outre leur fonction de représentation des éléments du système externe dans Orchestrator, les plug-ins pour systèmes peuvent également servir à améliorer l'évolutivité, à fournir un mécanisme de mise en cache, à gérer les événements et notifications, etc.
<b>Taille</b>	Les plug-ins pour systèmes sont de taille intermédiaire à importante. Les plug-ins pour systèmes demandent de nombreuses classes en plus de l'ensemble classique de classes, car ils proposent généralement un grand nombre d'objets de script. Les plug-ins pour systèmes peuvent nécessiter des systèmes d'aide et des classes auxiliaires supplémentaires pour interagir.
<b>Inventaire</b>	En règle générale, les plug-ins pour systèmes présentent un grand nombre d'objets que vous devez exposer correctement dans l'inventaire pour pouvoir facilement les localiser et les exploiter dans Orchestrator. Étant donné le grand nombre d'objets que les plug-ins pour systèmes doivent exposer, il est conseillé de créer un outil ou un processus auxiliaire pour générer automatiquement autant de code que possible pour chaque plug-in. Par exemple, le plug-in vCenter Server propose un outil de ce type.

## Plug-ins pour les systèmes orientés objets

Les systèmes orientés objets proposent un mécanisme d'interaction basés sur les objets et RPC.

Le modèle de système orienté objets le plus utilisé est le modèle de service Web s'appuyant sur SOAP. Les objets de ce modèle possèdent un ensemble d'attributs relatifs à l'état des objets et proposent des méthodes distantes que le système cible peut invoquer.

### Figure 6-6. Plug-ins pour les systèmes orientés objets



Il est conseillé de tenir compte des points suivants lors de l'implémentation de plug-ins pour les systèmes orientés objets.

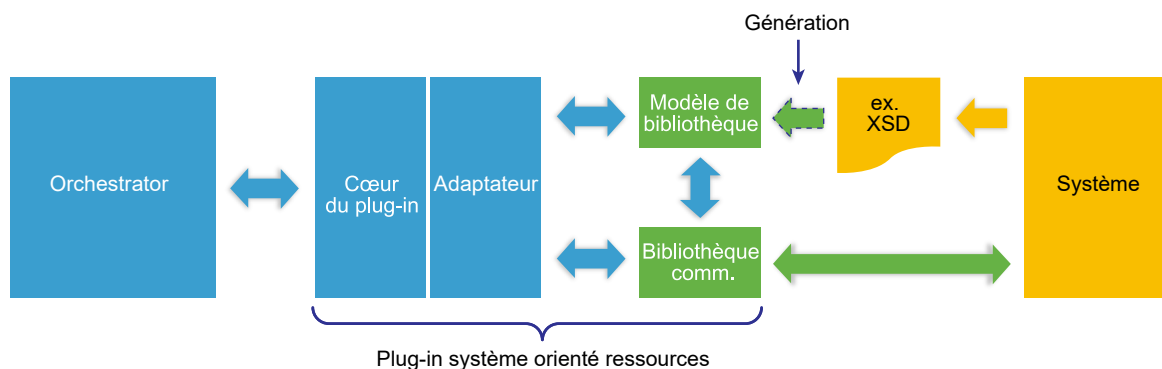
- Si vous utilisez SOAP, vous pouvez utiliser le fichier WSDL pour générer un ensemble de catégories qui associent le modèle d'objet au mécanisme de communication.
- Ce modèle d'objet constitue quasiment le seul élément à exposer dans Orchestrator.

## Plug-ins pour les systèmes orientés ressources

Les systèmes orientés ressources confèrent un mécanisme d'interaction basé sur les ressources et les opérations simples qui utilisent les méthodes HTTP.

Le modèle le plus représentatif de système orienté ressources est le modèle REST associé, par exemple, à XML. Les objets de ce modèle possèdent un ensemble d'attributs relatifs à leur état. Pour invoquer des méthodes sur le système cible (mécanisme de communication), vous devez utiliser des méthodes HTTP standard telles que GET, POST, PUT, etc., et suivre certaines conventions.

**Figure 6-7. Plug-ins pour les systèmes orientés ressources**



Il est conseillé de tenir compte des points suivants lors du développement de plug-ins pour les systèmes orientés ressources.

- Si vous utilisez REST ou uniquement HTTP avec XML, vous obtenez un ou plusieurs fichiers de schéma pour pouvoir lire et écrire des messages. À partir de ces schémas, vous pouvez générer un

ensemble de classes qui définissent le modèle d'objet. Cet ensemble de classes définit uniquement l'état des objets, car les opérations sont définies de façon implicite avec les méthodes HTTP (par exemple, telles qu'elles sont définies dans le plug-in vCloud Director) ou de façon explicite avec certains messages XML (par exemple, le plug-in Cisco UCSM).

- Vous devez implémenter le mécanisme de communication dans un autre ensemble de catégories. Cet ensemble de classes définit un nouveau modèle d'objet qui interagit avec le modèle d'objet d'origine. Le modèle d'objet du mécanisme de communication se compose uniquement d'objets et de méthodes.
- Vous pouvez exposer aussi bien le modèle d'objet d'origine que le modèle d'objet pour le mécanisme de communication dans Orchestrator. Cette opération risque d'ajouter de la complexité ; cela dépend de la façon dont les deux modèles d'objet sont exposés et de si vous décidez de fusionner les objets liés des deux côtés (afin de simuler un système orienté objet) ou de les maintenir séparés.

## Implémentation des plug-ins

Vous pouvez utiliser certaines pratiques et techniques utiles pour structurer vos plug-ins, implémenter les catégories Java et les objets JavaScript demandés, développer les workflows et les actions des plug-ins et rédiger la présentation d'un workflow.

- **Structure des projets**

Vous pouvez appliquer une structure standard aux projets de vos plug-ins Orchestrator.

- **Opérations internes des projets**

Vous pouvez appliquer certaines approches lorsque vous implémentez votre plug-in ; par exemple, en mettant des objets en cache, en plaçant des objets en arrière-plan, en clonant des objets, etc. Ces approches sont susceptibles d'améliorer les performances de vos plug-ins, d'éviter les problèmes de concurrence et d'améliorer la réactivité du client Orchestrator.

- **Opérations internes aux workflows**

Vous pouvez implémenter un workflow pour surveiller les opérations à long terme réalisées par votre plug-in Orchestrator.

- **Workflows et actions**

Vous pouvez appliquer certaines recommandations pour faciliter le développement et l'utilisation des workflows.

- **Présentation d'un workflow**

Lorsque vous créez la présentation d'un workflow, vous devez appliquer une certaine structure et certaines règles.

## Structure des projets

Vous pouvez appliquer une structure standard aux projets de vos plug-ins Orchestrator.

Vous pouvez utiliser une structure Maven standard avec les modules de vos projets de plug-ins afin de clarifier l'emplacement de chaque élément fonctionnel.

**Tableau 6-45. Structure d'un projet de plug-in**

Module	Description
/myAwesomePlugin-plugin	Racine du projet du plug-in.
/o11nplugin-myAwesomePlugin	Module qui compose le fichier final DAR du plug-in.
/o11nplugin-myAwesomePlugin-config	Module qui comporte l'application Web de configuration des plug-ins. Il génère un fichier WAR standard.
/o11nplugin-myAwesomePlugin-core	Module comportant l'ensemble des classes qui implémentent les interfaces des plug-ins Orchestrator et leurs classes annexes. Il génère un fichier JAR standard.
/o11nplugin-myAwesomePlugin-model	Module contenant l'ensemble des classes qui vous aident à intégrer la technologie tierce à Orchestrator via le plug-in. Les classes ne doivent pas contenir de référence directe aux API des plug-ins Orchestrator standard.
/o11nplugin-myAwesomePlugin-package	Module important un fichier de module Orchestrator externe avec des actions et des workflows à intégrer au fichier DAR final du plug-in. Ce module est facultatif.

## Opérations internes des projets

Vous pouvez appliquer certaines approches lorsque vous implémentez votre plug-in ; par exemple, en mettant des objets en cache, en plaçant des objets en arrière-plan, en clonant des objets, etc. Ces approches sont susceptibles d'améliorer les performances de vos plug-ins, d'éviter les problèmes de concurrence et d'améliorer la réactivité du client Orchestrator.

### Mettre des objets en cache

Votre plug-in peut interagir avec un service distant, et cette interaction est rendue possible par les objets locaux qui représentent les objets distants du côté du service. Pour optimiser les performances du plug-in et la réactivité de l'UI Orchestrator, vous pouvez mettre les objets locaux en cache au lieu de les récupérer à chaque fois auprès du service distant. Vous pouvez tenir compte de la portée du cache : par exemple, un cache pour l'ensemble des plug-ins clients, un cache par utilisateur du plug-in et un cache par utilisateur du service tiers. Une fois implémenté, votre mécanisme de mise en cache est intégré à l'interface du plug-in pour rechercher et invalider des objets.

### Placer des objets en arrière-plan

Si vous devez faire apparaître des listes importantes d'objets dans l'inventaire du plug-in et que vous ne disposez d'aucune manière pour récupérer ces objets rapidement, vous pouvez les placer en arrière-plan. Vous pouvez placer les objets en arrière-plan avec, par exemple, des objets dans deux états : `fake` et `loaded`. Supposons que les objets `fake` soient extrêmement faciles à créer et fournissent un minimum d'informations à faire apparaître dans l'inventaire, comme le nom et l'ID. Il serait donc possible de toujours renvoyer les objets `fake` et, lorsque toutes les informations (l'objet réel) sont demandées, l'entité utilisatrice ou le plug-in pourrait appeler automatiquement une méthode `Load` pour obtenir l'objet réel. Vous pouvez même configurer le processus de chargement des objets de manière qu'il démarre automatiquement après le renvoi des objets factices afin d'anticiper les actions de l'entité utilisatrice.

## Cloner des objets afin d'éviter les problèmes de concurrence

Si vous utilisez un cache pour votre plug-in, vous devez cloner des objets. L'utilisation d'un cache qui renvoie toujours la même instance d'un objet à chaque entité qui le demande peut provoquer des effets indésirables. Par exemple, l'entité A demande l'objet O et l'entité consulte l'objet dans l'inventaire avec l'ensemble de ses attributs. Au même moment, l'entité B demande également l'objet O, mais l'entité A exécute un workflow qui commence à modifier les attributs de l'objet O. À la fin de cette exécution, le workflow appelle la méthode `update` de l'objet pour mettre ce dernier à jour du côté du serveur. Si l'entité A et l'entité B obtiennent la même instance de l'objet O, l'entité A peut voir dans l'inventaire toutes les modifications que l'entité B réalise, avant même que les modifications soient envoyées du côté du serveur. Cela ne pose aucun problème si l'exécution se déroule normalement. Par contre, si l'exécution échoue, les attributs de l'objet O pour l'entité A ne sont pas rétablis. Dans ce cas, si le cache (les opérations `find` du plug-in) renvoie un clone de l'objet au lieu de la même instance à chaque fois, les entités utilisatrices consultent et modifient leur propre copie et les problèmes de concurrence sont évités, au moins dans Orchestrator.

## Notifier les modifications aux autres intervenants

Des problèmes peuvent survenir lorsque vous utilisez simultanément un objet en cache et un objet cloné. Le problème le plus important est que l'objet qui apparaît à l'entité risque de ne pas s'afficher dans sa dernière version disponible. Par exemple, si une entité affiche l'inventaire, les objets sont chargés une seule fois au même moment. Donc si une autre entité modifie certains objets, la première entité ne voit pas ces modifications. Pour éviter ce problème, vous pouvez utiliser les méthodes `PluginWatcher` et `IPluginPublisher` de l'API du plug-in Orchestrator afin de notifier les modifications apportées et que ces dernières apparaissent dans les autres instances des clients Orchestrator. Cette solution s'applique également à une instance unique du client Orchestrator lorsque les modifications d'un objet de l'inventaire affectent les autres objets. Ces modifications doivent également être notifiées. Les opérations sujettes aux notifications sont l'ajout, la mise à jour et la suppression d'objets lorsque ces objets ou certaines de leurs propriétés apparaissent dans l'inventaire.

## Activer la recherche d'un objet à tout moment

Vous devez implémenter la méthode `find` de l'interface `IPluginFactory` pour rechercher des objets uniquement par type et par ID. La méthode `find` peut être appelée directement après le redémarrage d'Orchestrator et la reprise d'un workflow.

## Simuler un service de requête si vous n'en possédez pas

Dans certains cas, le client Orchestrator peut demander l'utilisation de requêtes pour certains objets, ou leur affichage en liste ou en tableau plutôt qu'en arborescence, par exemple. Cela signifie que votre plug-in doit être en mesure d'envoyer des requêtes à tout moment pour certains ensembles d'objets. Si la technologie tierce propose un service de requête, vous devez adapter et utiliser ce service. Si ce n'est pas le cas, vous devez pouvoir simuler un service de requête malgré la complexité accrue et les performances réduites de la solution.

## Les méthodes de recherche ne doivent pas renvoyer d'exceptions d'exécution

Les méthodes de l'interface `IPluginFactory` qui implémentent les recherches dans le plug-in ne doivent pas générer d'exceptions d'exécution contrôlées ou non contrôlées. Cela risque de causer des échecs de type *erreur de validation* étonnants au cours de l'exécution d'un workflow. Par exemple, entre les deux nœuds d'un workflow, la méthode `find` est appelée si une sortie du premier nœud est une entrée du deuxième nœud. À ce moment, si l'objet est introuvable à cause d'une exception d'exécution, vous risquez d'obtenir uniquement une *erreur de validation* sans information complémentaire dans le client Orchestrator. La quantité d'informations apparaissant dans les fichiers journaux dépend de la façon dont le plug-in enregistre les exceptions.

## Opérations internes aux workflows

Vous pouvez implémenter un workflow pour surveiller les opérations à long terme réalisées par votre plug-in Orchestrator.

Vous pouvez implémenter un workflow pour la surveillance des opérations de longue durée telles que la surveillance des tâches. Ce workflow peut être basé sur des déclencheurs Orchestrator et des événements en attente. Vous devez considérer qu'un workflow bloqué en attente d'une tâche peut reprendre dès que le serveur Orchestrator démarre. Le plug-in doit être en mesure d'obtenir l'ensemble des informations requises pour reprendre correctement le processus de surveillance.

Le workflow de surveillance ou la tâche qu'il peut utiliser en interne doit fournir un mécanisme pour indiquer le taux d'interrogation et un délai d'expiration éventuel.

Le processus de débogage d'un code de script au sein d'un workflow n'est pas simple, notamment si le code n'appelle aucun code Java. C'est la raison pour laquelle, parfois, la seule option est d'utiliser les méthodes de journalisation proposées par les objets de script Orchestrator par défaut.

## Workflows et actions

Vous pouvez appliquer certaines recommandations pour faciliter le développement et l'utilisation des workflows.

### Commencer à développer des workflows en tant que blocs constitutifs

Un bloc constitutif peut être un workflow élémentaire qui nécessite seulement quelques paramètres d'entrée et renvoie une sortie simple. Si vous disposez d'un ensemble important de blocs constitutifs, vous pouvez créer facilement des workflows spécialisés et proposer un ensemble d'outils amélioré pour la création de workflows complexes.

### Créer des workflows spécialisés basés sur des composants plus petits

Si vous devez développer un workflow complexe avec plusieurs entrées et étapes internes, vous pouvez le fractionner en workflows et actions plus petits et plus simples de blocs constitutifs.

### Créer des actions dès que possible

Vous pouvez créer des actions pour ajouter de la flexibilité au développement des workflows.

- Pour créer facilement des objets ou des paramètres complexes pour les méthodes de script



- Pour éviter la répétition permanente des parties de code identiques
- Pour réaliser les validations de l'IU

### Les workflows doivent appeler des actions autant que possible

Les actions peuvent être appelées directement en tant que nœuds dans le schéma de workflow. Cette fonction évite de complexifier le schéma de workflow, car vous ne devez pas ajouter des composants de code de script pour appeler chaque action.

### Renseigner les informations demandées

Renseignez les informations de chaque élément d'un workflow ou d'une action.

- Fournissez une description du workflow ou de l'action.
- Fournissez une description des paramètres d'entrée.
- Fournissez une description des sorties.
- Fournissez une description des attributs des workflows.

### Mettre à jour régulièrement les informations relatives à la version

Lorsque vous versionnez des plug-ins, ajoutez des commentaires pertinents avec des informations telles que les mises à jour principales du plug-in, les détails d'implémentation importants, etc.

### Présentation d'un workflow

Lorsque vous créez la présentation d'un workflow, vous devez appliquer une certaine structure et certaines règles.

Utilisez les propriétés suivantes pour les entrées de workflow de la présentation du workflow.

**Tableau 6-46. Propriétés des entrées de workflow**

Propriétés	Utilisation
Show in Inventory	Utilisez cette propriété pour aider l'utilisateur à exécuter un workflow à partir de la vue Inventaire.
Specify a root object to be shown in the chooser	Utilisez cette propriété pour aider l'utilisateur à sélectionner des entrées. Si l'objet root peut être actualisé dans la présentation, qu'il représente un attribut ou qu'il est récupéré par une méthode d'objet, vous devez créer ou définir une action appropriée pour actualiser l'objet dans la présentation.
Maximum string length	Utilisez cette propriété pour les chaînes longues telles que les noms, les descriptions, les chemins d'accès des fichiers, etc.
Minimum string length	Utilisez cette propriété pour éviter les chaînes vides des outils de test.
Custom validation	Implémentez des validations autres que des validations simples avec des actions.

Organisez les entrées avec des étapes et affichez le groupe. Une telle organisation aide l'utilisateur à identifier et à distinguer l'ensemble des paramètres d'entrée d'un workflow.

## Recommandations pour le développement de plug-ins Orchestrator

Lors du développement des différents composants de vos plug-ins Orchestrator, le respect de certaines pratiques vous permettra d'améliorer la qualité de ces plug-ins.

**Tableau 6-47. Pratiques utiles pour l'implémentation des plug-ins**

Composant	Élément	Description
Général	Accéder à une API tierce	Dès que cela est possible, les plug-ins doivent fournir des méthodes simplifiées pour accéder à l'API tierce.
	Interface	Les plug-ins doivent fournir une interface cohérente et normalisée aux utilisateurs, même lorsque l'API ne respecte pas ce principe.
Action	Objets de scripts	Vous devez créer des actions pour chaque création, modification et suppression, ainsi que pour toutes les autres méthodes disponibles pour un objet de scripts.
	Description	La description d'une action doit décrire ce que fait une action et non comment elle fonctionne.
	Scripts	Lorsque vous utilisez les scripts pour obtenir les propriétés ou les méthodes d'un objet, vous pouvez vérifier si la valeur de l'objet est différente de null ou de undefined.
	Obsolescence	Si une action est obsolète, le comment ou la déclaration throw doit indiquer l'action de remplacement, ou l'action doit appeler une nouvelle action de remplacement pour que les solutions créées sur la version obsolète ne rencontrent pas de problème.
Workflow	Opérations de l'interface utilisateur dans la technologie orchestrée	Vous devez créer un workflow pour chaque opération disponible dans l'interface utilisateur de la technologie orchestrée.
	Description	La description d'un workflow doit décrire ce que fait ce workflow et non comment il fonctionne.
	Propriété de présentation mandatory input	Vous devez définir la propriété mandatory input pour toutes les entrées de workflow obligatoires.
	Propriété de présentation default value	Si vous développez un workflow qui configure une entité, la présentation du workflow doit charger les valeurs de configuration par défaut de cette entité. Par exemple, si vous développez un workflow nommé Configuration d'hôte, la présentation du workflow doit charger les valeurs par défaut de la configuration d'hôte.
	Propriété de présentation Show in inventory	Vous devez définir la propriété Show in inventory de manière à obtenir des workflows contextuels sur les objets d'inventaire.
	Propriété de présentation specify a root parameter	Vous devez utiliser cette propriété dans les workflows lorsqu'il n'est pas nécessaire de parcourir l'inventaire à partir de la racine de l'arborescence.
	Validation des workflows	Vous devez valider les workflows et corriger toutes les erreurs.
	Création d'objets	Tous les workflows qui créent un nouvel objet doivent renvoyer le nouvel objet en tant que paramètre de sortie.

**Tableau 6-47. Pratiques utiles pour l'implémentation des plug-ins (suite)**

Composant	Élément	Description
	Obsolescence	Si un workflow est obsolète, le comment ou la déclaration <code>throw</code> doit indiquer le workflow de remplacement, ou le workflow obsolète doit appeler un nouveau workflow de remplacement pour que les solutions créées sur les anciennes versions ne rencontrent pas de problème.
Inventaire	Déconnexion de l'hôte	Si votre inventaire comporte une connexion à un hôte et que cet hôte devient indisponible, vous devez indiquer que l'hôte est déconnecté. Vous pouvez effectuer cette opération soit en renommant l'objet root avec l'ajout de <code>- disconnected</code> , soit en supprimant l'arborescence d'objets sous cet objet, de la même manière que procède le plug-in vCloud Director.
	Propriété <code>Select value as list</code>	Un objet d'inventaire doit être sélectionnable en tant que <code>treeview</code> ou <code>list</code> .
	Gestionnaire d'hôtes	Si le plug-in implémente un objet <code>host</code> pour le système cible, alors un objet root <code>hostmanager</code> parent doit exister avec des propriétés d'ajout, de suppression ou de modification des propriétés des hôtes.
	Obtention ou mise à jour d'objets	Si un service de requête s'exécute sur la technologie orchestrée, vous devez l'utiliser pour obtenir les différents objets.
	Découverte des objets enfants	Si vous devez récupérer des objets enfants séparément, le processus de récupération doit être parallélisé et ne doit pas se bloquer à chaque erreur.
	Modification d'objets Orchestrator	Tous les workflows pouvant modifier l'état d'un élément dans l'inventaire doivent mettre l'inventaire à jour pour éviter que des objets restent à l'écart de la synchronisation.
	Modification d'objets externes	Vous pouvez utiliser un mécanisme de notification pour indiquer les modifications apportées à la technologie orchestrée suite à des opérations ayant eu lieu en dehors d'Orchestrator. Si de telles opérations ont entraîné la suppression d'objets de la technologie orchestrée, vous devez actualiser l'inventaire de manière appropriée afin d'éviter tout échec ou toute perte de données. Par exemple, si une machine virtuelle est supprimée de vCenter Server, le plug-in vCenter Server met à jour l'inventaire pour supprimer l'objet de la machine virtuelle supprimée.
	Objet de l'outil de recherche	Les objets du système de recherche présentent des propriétés qu'il est possible d'utiliser pour différencier les objets. Il s'agit généralement des propriétés qui apparaissent dans l'interface utilisateur.
Objet de script	Implémentation	La méthode <code>equals</code> doit être implémentée pour que l'opération <code>==</code> fonctionne sur le même objet. En effet, l'objet peut parfois présenter deux instances.
	Propriétés des objets des plug-ins	Les objets présentant des objets parents doivent implémenter une propriété <code>parent</code> .
	Propriétés des objets des plug-ins	Les objets possédant des objets enfants doivent implémenter des méthodes <code>GET</code> qui renvoient des tableaux d'objets enfants.

**Tableau 6-47. Pratiques utiles pour l'implémentation des plug-ins (suite)**

Composant	Élément	Description
	Objets d'inventaire	Les objets d'inventaire doivent pouvoir être recherchés avec <code>Server.find</code> .  Tous les objets d'inventaire doivent pouvoir être mis en série afin d'être utilisés en tant qu'attributs d'entrée ou de sortie dans un workflow.
	Constructeur et méthodes	Dans la plupart des cas, les objets inscriptibles possèdent un constructeur ou doivent être renvoyés par d'autres attributs ou méthodes d'objets.
	ID objet	Les objets possédant un ID créé par un système externe doivent utiliser un ID interne afin de garantir qu'aucun doublon d'ID ne se produit lorsque vous orchestrez plusieurs serveurs.
	Rechercher des objets	Les méthodes <code>search</code> ou <code>find</code> doivent implémenter un filtre pour que le nom ou l'ID indiqué puisse être recherché. Cela évite d'effectuer une recherche dans tous les objets. Par exemple, le serveur Orchestrator possède une méthode <code>Server.FindForId</code> qui permet de rechercher un objet de plug-in par son ID. Afin de parvenir à cela, la méthode doit être implémentée pour chaque objet pouvant être recherché dans le plug-in.
	Déclencher	Si possible, des déclencheurs doivent être disponibles pour les objets qui subissent des modifications afin qu'Orchestrator puisse déclencher ses stratégies sur plusieurs événements. Par exemple, Orchestrator peut surveiller un déclencheur ou un événement dans le plug-in vCenter de l'objet Datacenter afin de pouvoir déterminer le moment où une nouvelle machine virtuelle est ajoutée, mise en service, désactivée, etc.
	Propriétés des objets	Les objets hébergés dans les autres plug-ins doivent posséder des propriétés qui leur permettent d'être facilement convertis d'un objet de plug-in en un autre. Par exemple, les objets des machines virtuelles doivent disposer d'un <code>moref</code> (ID de référence d'objet géré).
	Gestionnaire de session	Si vous vous connectez à un serveur distant susceptible de présenter une session différente, le plug-in doit implémenter une session partagée et une session par utilisateur.
Déclencher	Déclencher	Toutes les opérations et méthodes de blocage de longue durée doivent pouvoir démarrer de manière asynchrone dès qu'une tâche est renvoyée, et générer un événement de déclenchement une fois terminées.
Énumérations	Énumérations	Les énumérations d'un type donné doivent présenter un objet d'inventaire qui permet la sélection de différentes valeurs.
Journalisation	Journaux	Les méthodes doivent implémenter différents niveaux de journaux.
Contrôle de version	Version de plug-in	La version du plug-in doit respecter les normes et être mise à jour parallèlement à la mise à jour du plug-in.

**Tableau 6-47. Pratiques utiles pour l'implémentation des plug-ins (suite)**

Composant	Élément	Description
Documentation sur l'API	Méthodes	Les méthodes sont décrites dans la documentation sur l'API et ne doivent jamais générer l'exception <code>no xyz method / property</code> sur un objet. Au contraire, les méthodes doivent renvoyer <code>null</code> lorsqu'aucune propriété n'est disponible et être documentées avec des détails.
	<code>vso.xml</code>	Tous les objets et toutes les méthodes et propriétés doivent être documentés dans <code>vso.xml</code> .

## Documentation des chaînes et des API de l'interface utilisateur du plug-in

Veillez à suivre les règles validées de style et de format lorsque vous écrivez des chaînes d'interface utilisateur pour les plug-ins Orchestrator et la documentation de l'API associée.

### Recommandations générales

- Utilisez les noms officiels des produits VMware impliqués dans le plug-in. Par exemple, utilisez les noms officiels des produits suivants et la terminologie VMware.

Terme correct	Ne pas utiliser
vCenter Server	VC ou vCenter
vCloud Director	vCloud

- Terminez toutes les descriptions de workflows avec un point. Par exemple, `Creates a new Organization.` est une description de workflow.
- Utilisez un éditeur de texte doté d'un correcteur orthographique pour écrire les descriptions, puis déplacez ces dernières dans le plug-in.
- Assurez-vous que le nom du plug-in correspond exactement au nom du produit tiers approuvé auquel il est associé.

### Workflows et actions

- Écrivez des descriptions informatives. Une ou deux phrases suffisent pour la plupart des actions et des workflows.
- Les workflows spécialisés peuvent inclure des descriptions et des commentaires plus détaillés.
- Commencez les descriptions par un verbe, par exemple `Creates...`. N'utilisez pas d'auto-référence telle que `This workflow creates.`
- Mettez un point à la fin des descriptions formées de phrases complètes.
- Décrivez les effets d'un workflow ou d'une action plutôt que ses modalités d'implémentation.

- Les workflows et les actions sont généralement inclus dans les dossiers et les modules. Incluez également une brève description de ces dossiers et de ces modules. Par exemple, un dossier de workflow peut présenter une description de type `Set of workflows related to vApp Template management`.

## Paramètres des workflows et des actions

- Commencez les descriptions des workflows et des actions par une phrase nominative descriptive, par exemple `Name of`. N'utilisez pas d'expressions telles que `It's the name of`.
- Ne mettez pas de point à la fin des descriptions des paramètres et des actions. Il ne s'agit pas de phrases complètes.
- Les paramètres d'entrée des workflows doivent présenter une étiquette affichant les noms appropriés dans la vue de présentation. Dans de nombreuses situations, vous pouvez associer les entrées connexes dans un groupe d'affichage. Par exemple, au lieu de créer deux entrées avec les étiquettes `Nom de l'entreprise` et `Nom complet de l'entreprise`, vous pouvez créer un groupe d'affichage avec l'étiquette `Entreprise` et placer les entrées `Nom` et `Nom complet` dans le groupe `Entreprise`.
- Pour les étapes et les groupes d'affichage, ajoutez également les descriptions et les commentaires qui apparaissent dans la présentation du workflow.

## API du plug-in

- La documentation de l'API se rapporte à l'ensemble de la documentation du fichier `vso.xml` et des fichiers Java source.
- Pour le fichier `vso.xml`, utilisez les mêmes règles pour les descriptions des objets de recherche et de script que pour les méthodes employées avec les workflows et les actions. Les descriptions des attributs des objets et des paramètres des méthodes utilisent les mêmes règles que pour les paramètres des workflows et des actions.
- Évitez les caractères spéciaux dans le fichier `vso.xml` et placez les descriptions à l'intérieur d'une balise `<![CDATA[insert your description here!]]>`.
- Utilisez le style Javadoc standard pour les fichiers Java source.

# Création de plug-ins à l'aide de Maven

# 7

L'Orchestrator Appliance propose un référentiel contenant des artefacts Maven que vous pouvez utiliser pour créer des projets de plug-ins à partir d'archétypes.

Adresse du référentiel : `https://orchestrator_server:8281/vco-repo/` ou `http://orchestrator_server:8280/vco-repo/` si votre version de Maven ne prend pas en charge le protocole HTTPS. L'emplacement est intégré au fichier `pom.xml` des projets de plug-ins Orchestrator Maven standard. Vous pourrez uniquement accéder au référentiel si vous avez déployé le Orchestrator Appliance.

Ce chapitre contient les rubriques suivantes :

- [Créer un plug-in Orchestrator avec Maven à partir d'un archétype](#)
- [Archétypes Maven](#)
- [Recommandations pour le développement de plug-ins basés sur Maven](#)

## Créer un plug-in Orchestrator avec Maven à partir d'un archétype

Vous pouvez créer un plug-in Orchestrator Maven standard à partir d'un archétype en exécutant des commandes dans l'interface de lignes de commandes.

### Conditions préalables

- Vérifiez que vous avez installé Orchestrator Appliance 5.5.1 ou une version ultérieure.
- Vérifiez que vous avez installé Apache Maven 3.0.4 ou 3.0.5.

## Procédure

- 1 Créez un projet en mode interactif en choisissant un archétype.

```
mvn archetype:generate -DarchetypeCatalog=https://orchestrator_server:8281/vco-repo/archetype-catalog.xml -DrepoUrl=https://orchestrator_server:8281/vco-repo -Dmaven.repo.remote=https://orchestrator_server:8281/vco-repo -Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
```

**Note** Vous pouvez accéder au référentiel Maven uniquement si vous avez déployé le dispositif Orchestrator Appliance.

- 2 (Facultatif) Si vous ne pouvez pas accéder au référentiel via HTTPS, vous pouvez y accéder via HTTP. Si vous accédez au référentiel via HTTP ou que vous disposez d'un certificat SSL valide, vous pouvez créer un projet sans utiliser l'indicateur `-Dmaven.wagon.http.ssl.allowall=true`.

```
mvn archetype:generate -DarchetypeCatalog=http://orchestrator_server:8280/vco-repo/archetype-catalog.xml -DrepoUrl=http://orchestrator_server:8280/vco-repo -Dmaven.repo.remote=http://orchestrator_server:8280/vco-repo -Dmaven.wagon.http.ssl.insecure=true
```

- 3 Naviguez jusqu'au répertoire du projet et créez le plug-in.

```
cd project_dir && mvn clean install -Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
```

Si le processus aboutit, le fichier de plug-in `.dar` est généré dans le répertoire `target/` du module DAR.

## Archétypes Maven

Vous pouvez utiliser des archétypes Maven prédéfinis en tant que modèles pour développer les plug-ins Orchestrator.

Le tableau suivant décrit les archétypes Maven par défaut disponibles dans Orchestrator.

**Tableau 7-1. Archétypes Maven par défaut**

Archétype	Description
<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>	Projet Maven basé uniquement sur le contenu. Il peut être utilisé pour conserver les modules sous leur format source afin d'améliorer l'interaction avec le traitement RCS, différé, postérieur, etc.
<code>com.vmware.o11n:011n-client-archetype-rest</code>	Outil de ligne de commande simple qui communique avec l'API REST d'Orchestrator et appelle un workflow.
<code>com.vmware.o11n:011n-plugin-archetype-inventory</code>	Plug-in qui illustre l'utilisation de l'inventaire. Le plug-in implémente un référentiel, un adaptateur et une fabrique pour un type. L'inventaire est stocké dans un fichier sur un disque.
<code>com.vmware.o11n:011n-archetype-inventory-annotation</code>	Plug-in dont le descripteur <code>vso.xml</code> est généré par dessus les annotations.



**Tableau 7-1. Archétypes Maven par défaut (suite)**

Archétype	Description
<code>com.vmware.o11n:011n-archetype-spring</code>	Plug-in qui utilise un SDK basé sur Spring, fournit un environnement DI et confère un niveau de services supérieur par rapport aux API standard de plug-ins.
<code>com.vmware.o11n:011n-plugin-archetype-modeldriven</code>	Archétype qui génère une structure pour créer des plug-ins avec ModelDriven.

## Recommandations pour le développement de plug-ins basés sur Maven

Vous pouvez améliorer le processus de fourniture de plug-ins Orchestrator créés avec Maven en réalisant un ensemble de tâches.

### Utilisation d'un gestionnaire de référentiels

Si vous créez des plug-ins dans une organisation importante, utilisez un gestionnaire de référentiels d'entreprise pour configurer le référentiel Orchestrator par défaut. Il sera ajouté en tant que référentiel proxy. L'utilisation d'un référentiel centralisé améliore la gestion et la collaboration sur les projets de plug-ins. Lorsque vous avez terminé la première version d'un nouveau référentiel, le gestionnaire de référentiels met en cache les artefacts du référentiel Orchestrator et vous pouvez désactiver le référentiel par défaut.

### Verrouillage des workflows

Après avoir vérifié que l'ensemble des workflows de votre plug-in fonctionnent de la manière souhaitée, verrouillez-les pour éviter toute modification non autorisée. Le verrouillage des workflows permet de s'assurer que les fonctions de base du plug-in ne peuvent pas être compromises. Si les utilisateurs doivent modifier un workflow par défaut pour une raison quelconque, ils peuvent créer une copie du workflow d'origine, puis modifier cette copie.

Il existe deux manières de produire des versions avec des workflows verrouillés.

- Transmettez le paramètre `-DallowedMask=vf` à Maven.
- Modifiez le document `pom.xml` et modifiez la valeur du paramètre `allowedMask` en `vf`.

```
<allowedMask>vf</allowedMask>
```

### Utilisation d'un certificat de signature de modules

Utilisez un certificat auto-signé ou un certificat signé par une autorité de certification pour assurer l'intégrité et l'authenticité des plug-ins. Stockez le certificat dans le keystore sous l'alias `_dunesrsa_alias_` en l'important avec le `keytool` dans le JDK.

Il existe deux manières d'indiquer le chemin d'accès au fichier du keystore et le mot de passe du keystore.

- Définissez les paramètres de ligne de commande `-DkeystoreLocation` et `-DkeystorePassword` pour la variable `MAVEN_OPTS`.
- Modifiez le fichier `pom.xml` afin d'insérer les valeurs manuellement. Par exemple,

```
<keystore>chemin d'accès au fichier keystore</keystore>  
<storepass>mot de passe keystore</storepass>
```

Si aucun keystore n'est importé, le fichier `.package` est signé avec le fichier `archetype.keystore`.