Consignes de codage pour WFA

OnCommand Workflow Automation 5.1

NetApp
April 19, 2024

Sommaire

Consignes de codage pour WFA

Vous devez connaître OnCommand Workflow Automation les directives de codage, les conventions de nommage et les recommandations générales relatives à la création d'éléments de base comme les filtres, les fonctions, les commandes et les workflows.

Instructions pour les variables

Avant de créer une commande ou un type de source de données, vous devez connaître les instructions relatives aux variables PowerShell et Perl dans OnCommand Workflow Automation (WFA).

Variables PowerShell

Directives	Exemple

Pour les paramètres d'entrée de script :
• Utilisez le cas de Pascal.
• Ne pas utiliser de trait de soulignement.
• N'utilisez pas d'abréviations.
	`$VolumeName`
	`$AutoDeleteOptions`
	`$Size`

Pour les variables internes de script :
• Utiliser l'étui Camel.
• Ne pas utiliser de trait de soulignement.
• N'utilisez pas d'abréviations.
	`$newVolume`
	`$qtreeName`
	`$time`

Pour les fonctions : • Utilisez le cas de Pascal. • Ne pas utiliser de trait de soulignement. • N'utilisez pas d'abréviations.	`GetVolumeSize`
Les noms de variables ne sont pas sensibles à la casse. Cependant, pour améliorer la lisibilité, vous ne devez pas utiliser une majuscule différente pour le même nom.	`$variable` est identique à `$Variable.`
Les noms de variables doivent être en anglais brut et doivent être liés à la fonctionnalité du script.	Utiliser `$name` et non `$a.`
Déclarez explicitement le type de données pour chaque variable.	taille [int]

Directives	Exemple
N’utilisez pas de caractères spéciaux (! @ # & % , .) et espaces.	Aucune
N’utilisez pas de mots clés réservés PowerShell.	Aucune

Regroupez les paramètres d’entrée en plaçant les paramètres obligatoires d’abord suivis des paramètres facultatifs.	`param(`
	`[parameter(Mandatory=$true)]`
	`[string]$Type,`
	
	`[parameter(Mandatory=$true)]`
	`[string]$Ip,`
	
	`[parameter(Mandatory=$false)]`
	`[string]$VolumeName`
	`)`

Commenter toutes les variables d’entrée à l’aide de *`HelpMessage`* annotation avec un message d’aide significatif.	`[parameter(Mandatory=$false,HelpMessage="LUN to map")]` `[string]$LUNName`
N’utilisez pas « Filer » comme nom de variable ; utilisez plutôt « Array ».	Aucune

Utiliser *`ValidateSet`* annotation dans les cas où l’argument obtient des valeurs énumérées. Ceci se traduit automatiquement par le type de données Enum pour le paramètre.	`[parameter(Mandatory=$false,HelpMessage="Volume state")]`
	`[ValidateSet("online","offline","restricted")]`
	`[string]$State`

Ajoutez un alias à un paramètre qui se termine par “_Capacity” pour indiquer que le paramètre est de type de capacité.	La commande « Create Volume » utilise les alias comme suit :
	
	`[parameter(Mandatory=$false,HelpMessage="Volume increment size in MB")]`
	`[Alias("AutosizeIncrementSize_Capacity")]`
	`[int]$AutosizeIncrementSize`

Directives	Exemple
Ajoutez un alias à un paramètre qui se termine par “_Password” pour indiquer que le paramètre est de type mot de passe.	`param ([parameter(Mandatory=$false, HelpMessage="In order to create an Active Directory machine account for the CIFS server or setup CIFS service for Storage Virtual Machine, you must supply the password of a Windows account with sufficient privileges")] [Alias("Pwd_Password")] [string]$ADAdminPassword)`

Variables Perl

Directives	Exemple

Pour les paramètres d’entrée de script :
• Utilisez le cas de Pascal.
• Ne pas utiliser de trait de soulignement.
• N’utilisez pas d’abréviations.
	`$VolumeName`
	`$AutoDeleteOptions`
	`$Size`

N’utilisez pas d’abréviations pour les variables internes de script.	`$new_volume`
	`$qtree_name`
	`$time`

N’utilisez pas d’abréviations pour les fonctions.	`get_volume_size`
Les noms de variables sont sensibles à la casse. Pour améliorer la lisibilité, vous ne devez pas utiliser de majuscules différentes pour le même nom.	`$variable` n’est pas identique à `$Variable.`
Les noms de variables doivent être en anglais brut et doivent être liés à la fonctionnalité du script.	Utiliser `$name` et non `$a.`
Regroupez les paramètres d’entrée en plaçant d’abord les paramètres obligatoires, puis les paramètres facultatifs.	Aucune

Directives	Exemple

Dans la fonction GetOptions, déclarez explicitement le type de données de chaque variable pour les paramètres d'entrée.	`GetOptions(`
	`  "Name=s"=>\$Name,`
	`  "Size=i"=>\$Size`
	`)`

N'utilisez pas « Filer » comme nom de variable ; utilisez plutôt « Array ».	Aucune

Perl n'inclut pas le *`ValidateSet`* annotation des valeurs énumérées. Utilisez des déclarations explicites « si » pour les cas où l'argument obtient des valeurs énumérées.	`if`
	`(defined$SpaceGuarantee&&!($SpaceG`
	`uaranteeeq'none'`

	$SpaceGuaranteeeq'volume'
	$SpaceGuaranteeeq'file')) { die'Illegal SpaceGuarantee argument: \".$SpaceGuarantee.'\"; } ----

Toutes les commandes Perl WFA doivent utiliser le pragma "dit" pour décourager l'utilisation de constructions dangereuses pour les variables, les références et les sous-routines.	`use strict;`
	`# the above is equivalent to`
	`use strictvars;`
	`use strictsubs;`
	`use strictrefs;`

Toutes les commandes Perl WFA doivent utiliser les modules Perl suivants :
• Getopt
Ceci est utilisé pour spécifier les paramètres d'entrée.
• Util. Wutil
Cette fonction est utilisée pour les fonctions d'utilitaire fournies pour la journalisation des commandes, la génération de rapports sur la progression des commandes, la connexion aux contrôleurs de matrice, etc.
	`use Getopt::Long;`
	`use NaServer;`
	`use WFAUtil;`

Directives pour la mise en retrait

Vous devez connaître les consignes d'indentation lors de l'écriture d'un script PowerShell ou Perl pour OnCommand Workflow Automation (WFA).

Directives	Exemple
Un onglet est égal à quatre espaces vides.	

Utilisez les onglets et les accolades pour montrer le début et la fin d'un bloc.	Script PowerShell
	`if`
	`($pair.length-ne 2)`
	`{`
	`throw "Got wrong input data"`
	`}`
	Script Perl
	`if`
	`(defined $MaxDirectorySize)`
	`{`
	`# convert from MBytes to Bytes`
	`my $MaxDirectorySizeBytes =`
	`$MaxDirectorySize *`
	`1024 * 1024;`
	`}`

Ajoutez des lignes vides entre des ensembles d'opérations ou des segments de code.	`$options=$option.trim();`
	`$pair=$option.split(" ");`
	`Get-WFAlogger -Info -messages`
	`$("split options: "+`
	`$Pair)`

Lignes directrices pour les commentaires

Vous devez connaître les instructions relatives aux commentaires PowerShell et Perl dans les scripts pour OnCommand Workflow Automation (WFA).

Commentaires de PowerShell

Directives	Exemple
Utilisez le caractère # pour un commentaire sur une seule ligne.	`# Single line comment` `$options=$option.trim();`
Utilisez le caractère # pour un commentaire de fin de ligne.	`$options=$option.trim(); # End of line` `comment`

Utilisez les caractères <# et #> pour un commentaire de bloc.	`<#`
	`This is`
	`a`
	`block comment`
	`#>`
	`$options=$option.trim();`

Commentaires Perl

Directives	Exemple

Utilisez le caractère # pour un commentaire sur une seule ligne.	`# convert from MBytes to Bytes`
	`my $MaxDirectorySizeBytes =`
	`$MaxDirectorySize *`
	`1024 * 1024;`

Utilisez le caractère # pour le commentaire de fin de ligne.	`my $MaxDirectorySizeBytes =`
	`$MaxDirect`
	`orySiZe * 1024 * 1024; # convert`
	`to Bytes`

Directives	Exemple

Utilisez le caractère # dans chaque ligne avec un # vide au début et à la fin pour créer une bordure de commentaire pour les commentaires multilignes.	`#`
	`# This is a multi-line comment. Perl 5, unlike`
	`# Powershell, does not have direct support for`
	`# multi-line comments. Please use a '#'in every line`
	`# with an empty '#' at the beginning and end to create`
	`# a comment border`
	`#`

N’incluez pas de code commenté et mort dans les commandes WFA. Cependant, à des fins de test, vous pouvez utiliser le mécanisme POD (Plain Old Documentation) pour commenter le code.	`=begin comment`
	`  # Set deduplication`
	`  if(defined $Deduplication && $Deduplication eq "enabled")`
	`  {`
	`  $wfaUtil->sendLog("Enabling Deduplication");`
	`  }`
	`=end comment`
	`=cut`

Instructions pour l’enregistrement

Vous devez tenir compte des instructions à suivre pour la connexion lors de l’écriture d’un script PowerShell ou Perl pour OnCommand Workflow Automation (WFA).

La connexion PowerShell

Directives	Exemple
Utilisez l’applet de commande Get-WFALogger pour la consignation.	`Get-WFALogger -Info -message “Creating volume”`

Directives	Exemple
Consigner toutes les actions nécessitant une interaction avec des packages internes tels que Data ONTAP, VMware et PowerCLI.tous les messages de journal sont disponibles dans les journaux d'exécution dans l'historique d'état d'exécution des flux de travail.	Aucune
Consignez tous les arguments pertinents transmis aux packages internes.	Aucune
Utilisez les niveaux de journal appropriés lorsque vous utilisez l'applet de commande Get-WFALogger, en fonction du contexte d'utilisation. -Info, -Error, -Waren et -Debug sont les différents niveaux de journal disponibles. Si un niveau de journal n'est pas spécifié, le niveau de journal par défaut est Déboguer.	Aucune

Journalisation Perl

Directives	Exemple

Utilisez WFAUtil sendLog pour la consignation.	`my wfa_util = WFAUtil->new();`
	`eval {`
	`$wfa_util->sendLog('INFO',`
	`"Connecting to the`
	`cluster: $DestinationCluster");`
	`}`

Consignez chaque action nécessitant une interaction avec des éléments externes à la commande, tels que Data ONTAP, VMware et WFA. Tous les messages de journal que vous créez à l'aide de la routine WFAUtil sendLog sont stockés dans la base de données WFA. Ces messages de journal sont disponibles pour le workflow et la commande exécutés.	Aucune
Consigner tous les arguments pertinents transmis à la routine appelée.	Aucune
Utilisez les niveaux de journal appropriés.-Info, -erreur, -avertir et -Debug sont les différents niveaux de journal disponibles.	Aucune

Directives	Exemple

Lors de la connexion au niveau -Info, soyez précis et concis. Ne spécifiez pas de détails d'implémentation tels que le nom de classe et le nom de fonction dans les messages de journal. Décrivez l'étape exacte ou l'erreur exacte en anglais.	L'extrait de code suivant montre un exemple de bon message et un message incorrect :
	`$wfa_util->sendLog('WARN', "Removing volume: '.$VolumeName); # Good Message`
	`$wfa_util->sendLog('WARN', 'Invoking volume-destroy ZAPI: '.$VolumeName); # Bad message`

Consignes de gestion des erreurs

Vous devez connaître les consignes relatives à la gestion des erreurs lors de l'écriture d'un script PowerShell ou Perl pour OnCommand Workflow Automation (WFA).

Gestion des erreurs PowerShell

Directives	Exemple

Les paramètres communs ajoutés aux applets de commande par le runtime PowerShell comprennent des paramètres de gestion des erreurs tels que ErrorAction et WarningAction :
• Le paramètre ErrorAction détermine comment une cmdlet doit réagir à une erreur de non-fin à partir de la commande.
• Le paramètre WarningAction détermine comment une cmdlet doit réagir à un avertissement de la commande.
• Stop, SilentlyCongue, Inquid et continue sont les valeurs valides pour les paramètres ErrorAction et WarningAction.
Pour plus d'informations, vous pouvez utiliser le `Get-Help about_CommonParameters` Commande dans l'interface de ligne de commandes PowerShell.
	ErrorAction : l'exemple suivant montre comment gérer une erreur de non-fin en tant qu'erreur de fin :
	`New-NcIgroup-Name $IgroupName-Protocol $Protocol-Type$OSType-ErrorActionstop`
	Warningaction
	`New-VM-Name $VMName-VM $SourceVM-DataStore$DataStoreName-VMHost$VMHost-WarningActionSilentlyContinue`

Directives	Exemple

Utilisez l’instruction générale « try/Catch » si le type d’exception entrante est inconnu.	`try`
	`{`
	`"In Try/catch block"`
	`}`
	`catch`
	`{`
	`"Got exception"`
	`}`

Utilisez l’instruction « try/Catch » spécifique si le type d’exception entrante est connu.	`try`
	`{`
	`"In Try/catch block"`
	`}`
	`catch[System.Net.WebExceptional],`
	`[System.IO.`
	`IOException]`
	`{`
	`"Got exception"`
	`}`

Utilisez la déclaration « enfin » pour libérer des ressources.	`try`
	`{`
	`"In Try/catch block"`
	`}`
	`catch`
	`{`
	`"Got exception"`
	`}`
	`finally`
	`{`
	`"Release resources"`
	`}`

Directives	Exemple

Utilisez les variables automatiques de PowerShell pour accéder aux informations relatives aux exceptions.	`try`
	`{`
	`Get-WFALogger -Info -message`
	`$("Creating`
	`Ipspace: " + $Ipspace)`
	`New-NaNetIpspace-Name $Ipspace`
	`}`
	`catch`
	`{`
	`Throw "Failed to create Ipspace.`
	`Message:`
	`" + $_.Exception.Message;`
	`}`

Gestion des erreurs Perl

Directives	Exemple
Perl n’inclut pas la prise en charge du langage natif pour les blocs Try/Catch. Utilisez des blocs d’évaluation pour vérifier et gérer les erreurs. Conserver les blocs d’évaluation aussi petits que possible.	`eval {` `$wfa_util->sendLog('INFO', "Quiescing the relationship : $DestinationCluster://$DestinationVserver /$DestinationVolume"); $server->snapmirror_quiesce('destination-vserver' => $DestinationVserver, 'destination-volume' => $DestinationVolume); $wfa_util->sendLog('INFO', 'Quiesce operation started successfully.'); };`  `$wfa_util->checkEvalFailure("Failed to quiesce the SnapMirror relationship $DestinationCluster://$DestinationVserver /$DestinationVolume", $@);`

Conventions générales PowerShell et Perl pour WFA

Vous devez comprendre certaines conventions PowerShell et Perl utilisées dans WFA pour créer des scripts cohérents avec les scripts existants.

- Utilisez des variables qui permettent de clarifier ce que vous voulez faire le script.
- Écrire un code lisible qui peut être compris sans commentaires.
- Privilégiez les scripts et les commandes aussi simplement que possible.
- Pour les scripts PowerShell :
  - Utiliser les applets de commande lorsque cela est possible.
  - Appelez le code .NET lorsqu’aucune applet de commande n’est disponible.
- Pour les scripts Perl :

- Toujours mettre fin aux instructions "comme" avec des caractères de newline.

  En l'absence d'un caractère newline, le numéro de ligne de script est imprimé, ce qui n'est pas utile pour le débogage des commandes Perl exécutées par WFA.

- Dans le module « Getopt », rendre obligatoire les arguments de chaîne à une commande.

Modules Perl fournis avec Windows

Certains modules Perl sont fournis avec la distribution Perl à état actif Windows pour OnCommand Workflow Automation (WFA). Vous pouvez utiliser ces modules Perl dans votre code Perl pour écrire des commandes, uniquement si elles sont fournies avec Windows.

Le tableau suivant répertorie les modules de base de données Perl fournis avec Windows pour WFA.

Module de base de données	Description
DBD::mysql	Pilote d'interface de base de données perl5 qui vous permet de vous connecter à la base de données MySQL.
Essayez::minuscule	Réduit les erreurs courantes grâce à des blocs d'évaluation.
XML::libxml	Interface avec libxml2 qui fournit des analyseurs XML et HTML avec des interfaces DOM, SAX et XMLReader.
DBD::Cassandra	Pilote d'interface de base de données perl5 pour Cassandra qui utilise le langage de requête CQL3.

Considérations relatives à l'ajout de modules PowerShell et Perl personnalisés

Vous devez prendre en compte certaines considérations avant d'ajouter des modules PowerShell et Perl personnalisés à OnCommand Workflow Automation (WFA). Les modules personnalisés PowerShell et Perl vous permettent d'utiliser des commandes personnalisées pour créer des flux de travail.

- Au cours de l'exécution des commandes WFA, tous les modules PowerShell personnalisés sont ajoutés au répertoire d'installation de WFA `/Posh/modules` sont automatiquement importées.
- Tous les modules Perl personnalisés ajoutés au `WFA/perl` Le répertoire est inclus dans la bibliothèque *@Inc*.
- Les modules PowerShell et Perl personnalisés ne sont pas sauvegardés dans le cadre des opérations de sauvegarde WFA.
- Les modules PowerShell et Perl personnalisés ne sont pas restaurés lors de l'opération de restauration WFA.

Vous devez sauvegarder manuellement des modules PowerShell et Perl personnalisés afin de les copier vers une nouvelle installation WFA.

Le nom du dossier dans le répertoire des modules doit être identique à celui du nom du module.

Applets de commande et fonctions WFA

OnCommand Workflow Automation (WFA) propose plusieurs applets de commande PowerShell, ainsi que des fonctions PowerShell et Perl que vous pouvez utiliser dans vos commandes WFA.

Vous pouvez afficher toutes les applets de commande PowerShell et fonctions fournies par le serveur WFA à l'aide des commandes PowerShell suivantes :

- `Get-Command -Module WFAWrapper`
- `Get-Command -Module WFA`

Vous pouvez afficher toutes les fonctions Perl fournies par le serveur WFA dans le `WFAUtil.pm` module. Les sections d'aide, les applets de commande WFA PowerShell et les méthodes WFA Perl aident à accéder aux liens du module d'aide WFA (support), qui permettent d'accéder à toutes les fonctions et applets de commande PowerShell ainsi qu'aux fonctions Perl.

Modules PowerShell et Perl WFA

Pour écrire des scripts pour vos flux de production, vous devez avoir connaissance des modules PowerShell ou Perl pour OnCommand Workflow Automation (WFA).

Modules PowerShell

Directives	Exemple
Utilisez le kit d'outils PS Data ONTAP pour appeler des API dès que le kit d'outils est disponible.	Le `Add VLAN` la commande utilise la boîte à outils comme suit : `Add-NaNetVlan-Interface $Interface-Vlans$VlanID`
S'il n'y a pas d'applets de commande disponibles dans le kit d'outils Data ONTAP PS, utilisez le `Invoke-SSH` Commande permettant d'appeler l'interface de ligne de commandes sur Data ONTAP.	`Invoke-NaSsh-Name $ArrayName-Command "ifconfig -a"-Credential $Credentials`

Modules Perl

Le module NaServer est utilisé dans les commandes WFA. Le module NaServer permet l'invocation des API Data ONTAP, utilisées dans la gestion active des systèmes Data ONTAP.

Directives	Exemple

Utilisez le module NaServer pour appeler des API dès que le SDK de gestion NetApp est disponible.	L'exemple suivant montre comment le module NaServer est utilisé pour une opération de reprise SnapMirror :
	
	``
	eval {
	
	$wfa_util->sendLog('INFO',
	"Connecting to the cluster: $DestinationCluster"
	);
	my $server
	= $wfa_util->connect($DestinationClusterIp, $DestinationVserver);
	
	my $sm_info = $server->snapmirror_get(
	'destination-vserver' => $DestinationVserver,
	'destination-volume' => $DestinationVolume
	);
	
	my $sm_state = $sm_info->{'attributes'}->{'snapmirror-info'}->{'mirror-state'};
	my $sm_status = $sm_info->{'attributes'}->{'snapmirror-info'}->{'relationship-status'};
	
	$wfa_util->sendLog('INFO',
	"SnapMirror relationship is $sm_state ($sm_status)");
	
	if ($sm_status ne 'quiesced') {
	$wfa_util->sendLog('INFO',
	'The status needs to be quiesced to resume transfer.');
	} else {
	my $result = $server->snapmirror_resume(
	'destination-vserver' => $DestinationVserver,
	'destination-volume' => $DestinationVolume
	);
	$wfa_util->sendLog('INFO',
	``

Directives	Exemple
Si une API Data ONTAP n'est pas disponible, appelez l'interface de ligne de commande Data ONTAP à l'aide de la méthode utilitaire executeSystemsmake. ExecuteSystemsmaTM n'est pas pris en charge et n'est actuellement disponible que pour Data ONTAP 7-mode.	Aucune

Considérations relatives à la conversion des commandes PowerShell en Perl

Vous devez prendre en compte certaines considérations importantes lorsque vous convertissez des commandes PowerShell en Perl, car PowerShell et Perl possèdent des fonctionnalités différentes.

Types d'entrée de commande

OnCommand Workflow Automation (WFA) permet aux concepteurs de flux de production d'utiliser des baies et un hachage comme entrées pour la commande lors de la définition d'une commande. Ces types d'entrée ne peuvent pas être utilisés lorsque la commande est définie à l'aide de Perl. Si vous voulez qu'une commande Perl accepte les entrées de tableau et de hachage, vous pouvez définir l'entrée comme une chaîne dans le concepteur. La définition de la commande peut alors analyser l'entrée, qui est transmise pour créer une matrice ou un hachage selon les besoins. La description de l'entrée décrit le format dans lequel l'entrée est attendue.

```
my @input_as_array  = split(',', $InputString); #Parse the input string of
format val1,val2 into an array

my %input_as_hash = split /[;=]/, $InputString; #Parse the input string of
format key1=val1;key2=val2 into a hash.
```

Déclaration PowerShell

Les exemples suivants montrent comment une entrée de tableau peut être transmise à PowerShell et Perl. Les exemples décrivent le CronMonth d'entrée, qui spécifie le mois où le travail cron est planifié pour s'exécuter. Les valeurs valides sont des nombres entiers compris entre -1 et 11. Une valeur de -1 indique que le planning s'exécute tous les mois. Toute autre valeur indique un mois donné, 0 étant janvier et 11 décembre.

```
[parameter(Mandatory=$false, HelpMessage="Months in which the schedule
executes. This is a comma separated list of values from 0 through 11.
Value -1 means all months.")]
  [ValidateRange(-1, 11)]
  [array]$CronMonths,
```

Instructions Perl

```
GetOptions(
    "Cluster=s"         => \$Cluster,
    "ScheduleName=s"    => \$ScheduleName,
    "Type=s"            => \$Type,
    "CronMonths=s"      => \$CronMonths,
) or die 'Illegal command parameters\n';

sub get_cron_months {
    return get_cron_input_hash('CronMonths', $CronMonths, 'cron-month',
-1,
        11);
}

sub get_cron_input_hash {
    my $input_name   = shift;
    my $input_value  = shift;
    my $zapi_element = shift;
    my $low          = shift;
    my $high         = shift;
    my $exclude      = shift;

    if (!defined $input_value) {
        return undef;
    }

    my @values = split(',', $input_value);

    foreach my $val (@values) {
        if ($val !~ /^[+-]?\d+$/) {
            die
                "Invalid value '$input_value' for $input_name: $val must
be an integer.\n";
        }
        if ($val < $low || $val > $high) {
            die
                "Invalid value '$input_value' for $input_name: $val must
be from $low to $high.\n";
        }
        if (defined $exclude && $val == $exclude) {
            die
                "Invalid value '$input_value' for $input_name: $val is not
valid.\n";
        }
    }
    # do something
}
```

Définition de commande

Il peut être nécessaire d'étendre une expression à une seule ligne dans PowerShell à l'aide d'un opérateur de tuyauterie en plusieurs blocs d'instructions en Perl afin d'obtenir la même fonctionnalité. Le tableau suivant illustre un exemple de l'une des commandes d'attente.

Déclaration PowerShell	Instructions Perl
`# Get the latest job which moves the specified volume to the specified aggregate.` `$job = Get-NcJob -Query $query`	where {$_.JobDescription -eq "Split" + $VolumeCloneName}

Select-Object -First 1 ----	`my $result = $server->job_get_iter(`
	`  'query' => {'job-type' => 'VOL_CLONE_SPLIT'},`
	`  'desired-attributes' => {`
	`  'job-type' => '',`
	`  'job-description' => '',`
	`  'job-progress' => '',`
	`  'job-state' => ''`
	`  }`
	`);`
	`my @jobarray;`
	`for my $job (@{ $result->{'attributes-list'}})`
	`{`
	`  my $description = $job->{'job-description'};`
	`  if($description =~ /$VolumeCloneName/)`
	`  {`
	`  push(@jobarray, $job)`
	`  }`
	`}`

Directives pour les éléments de base WFA

Vous devez connaître les instructions relatives à l'utilisation des éléments de base de Workflow Automation.

Instructions pour SQL dans WFA

Vous devez connaître les instructions relatives à l'utilisation de SQL dans OnCommand Workflow Automation (WFA) pour écrire des requêtes SQL pour WFA.

SQL est utilisé dans les emplacements suivants de WFA :

- Requêtes SQL permettant de renseigner les entrées utilisateur pour la sélection
- Requêtes SQL pour la création de filtres permettant de filtrer des objets d'un type d'entrée de dictionnaire spécifique
- Données statiques dans les tables de la base de données du terrain de jeu
- Type de source de données personnalisé de type SQL où les données doivent être extraites d'une source de données externe telle qu'une base de données de gestion de configuration personnalisée (CMDB).
- Requêtes SQL pour les scripts de réservation et de vérification

Directives	Exemple

Les mots-clés réservés SQL doivent être en majuscules.	`SELECT`
	`  vserver.name`
	`FROM`
	`  cm_storage.vserver vserver`

Les noms de tables et de colonnes doivent être en caractères minuscules.	Tableau : agrégat Colonne : espace_utilisé_mb
Séparez les mots par un caractère de soulignement (_). Les espaces ne sont pas autorisés.	performances_de_la_baie
Le nom de la table est défini en singulier. Une table est une collection d'une ou plusieurs entrées.	« fonction », et non « fonctions »

Directives	Exemple

Utilisez des alias de table avec des noms significatifs dans LES requêtes DE SÉLECTION.	`SELECT`
	`  vserver.name`
	`FROM`
	`  cm_storage.cluster cluster,`
	`  cm_storage.vserver vserver`
	`WHERE`
	`  vserver.cluster_id = cluster.id`
	`  AND cluster.name = '${ClusterName}'`
	`  AND vserver.type = 'cluster'`
	`ORDER BY`
	`  vserver.name ASC`

Directives	Exemple

Si vous devez faire référence à un paramètre d'entrée de filtre ou à un paramètre d'entrée utilisateur dans une requête de filtre ou d'utilisateur, utilisez la syntaxe comme '${inputVariableName}.vous pouvez également utiliser la syntaxe pour faire référence à un paramètre de définition de commande dans les scripts de réservation et de vérification.	`SELECT`
	`  volume.name AS Name,`
	`  aggregate.name as Aggregate,`
	`  volume.size_mb AS 'Total Size (MB)',`
	`  voulme.used_size_mb AS 'Used Size (MB)',`
	`  volume.space_guarantee AS 'Space Guarantee'`
	`FROM`
	`  cm_storage.cluster,`
	`  cm_storage.aggregate,`
	`  cm_storage.vserver,`
	`  cm_storage.volume`
	`WHERE`
	`  cluster.id = vserver.cluster_id`
	`  AND aggregate.id = volume.aggregate_id`
	`  AND vserver.id = voulme.vserver_id`
	`  AND vserver.name = '${VserverName}'`
	`  AND cluster.name = '${ClusterName}'`
	`ORDER BY`
	`  volume.name ASC`

Utilisez des commentaires pour les requêtes complexes. Certains styles de commentaires pris en charge dans les requêtes sont les suivants :
• «»--» jusqu'à la fin de la ligne
Un espace est obligatoire après le second tiret dans ce style de commentaire.
• D'un caractère «»#» jusqu'à la fin de la ligne
• À partir d'un ""/**" to the following "**/« séquence
	`/*`
	`multi-line`
	`comment`
	`*/`
	`--line comment`
	`SELECT`
	`  ip as ip, # comment till end of this line`
	`  NAME as name`
	`FROM --end of line comment`
	`  storage.array`

Directives pour les fonctions WFA

Vous pouvez créer des fonctions pour encapsuler une logique couramment utilisée et plus complexe dans une fonction nommée, puis réutiliser la fonction comme valeurs de paramètre de commande ou valeurs de paramètres de filtre dans OnCommand Workflow Automation (WFA).

Directives	Exemple
Utilisez Camel case pour un nom de fonction.	Calcul VolumeSize
Les noms de variables doivent être en anglais clair et liés à la fonctionnalité de la fonction.	SplitByDelimiter
N'utilisez pas d'abréviations.	CalculateVolumeSize, *NOT* calciVolSize
Les fonctions sont définies à l'aide de MVFLEX expression Language (MVEL).	Aucune
La définition de la fonction doit être spécifiée conformément aux directives officielles du langage de programmation Java.	Aucune

Instructions pour les entrées de dictionnaire WFA

Vous devez connaître les instructions de création d'entrées de dictionnaire dans OnCommand Workflow Automation (WFA).

Directives	Exemple
Les noms d'entrée du dictionnaire ne doivent contenir que des caractères alphanumériques et des traits de soulignement.	Licence_cluster Switch_23
Les noms d'entrée du dictionnaire doivent commencer par un caractère en majuscules. Commencez chaque mot du nom par un caractère en majuscule et séparez chaque mot par un trait de soulignement (_).	Volumétrie Licence_baie
Les noms des attributs d'entrée du dictionnaire ne doivent pas inclure le nom de l'entrée du dictionnaire.	Aucune
Les attributs et les références d'une entrée de dictionnaire doivent être en caractères minuscules.	agrégat, size_mb
Séparez les mots par un trait de soulignement. Les espaces ne sont pas autorisés.	pool_ressources

Directives	Exemple
Les entrées du dictionnaire ne peuvent pas inclure de références provenant d'un schéma différent. Lorsqu'une entrée de dictionnaire nécessite une référence croisée à un objet dans un autre schéma, assurez-vous que toutes les clés naturelles de l'objet auquel il est fait référence sont présentes dans l'entrée du dictionnaire.	L'entrée du dictionnaire Array_Performance nécessite toutes les clés naturelles de l'entrée du dictionnaire Array comme attributs directs.
Utilisez les types de données appropriés pour les attributs.	Aucune
Utilisez le type de données long pour les attributs relatifs à la taille ou à l'espace.	Size_mb et Available_size_mb dans Storage.Volume Dictionary
Utilisez Enum lorsqu'un attribut possède un ensemble de valeurs fixe.	raid_type dans l'entrée Storage.Volume Dictionary
Définissez « à mettre en cache » comme vrai pour un attribut ou une référence lorsqu'une source de données fournit une valeur pour cet attribut ou cette référence.pour la source de données Active IQ Unified Manager, ajoutez des attributs de mise en cache si la source de données peut lui fournir la valeur.	Aucune
Définissez « peut être nul » comme vrai si la source de données fournissant la valeur de cet attribut ou de cette référence peut renvoyer NULL.	Aucune
Fournissez une description pertinente à chaque attribut et référence.la description s'affiche dans les détails de la commande lors de la conception d'un flux de travail.	Aucune
N'utilisez pas « ID » comme nom d'attribut dans les entrées de dictionnaire. Il est réservé à l'utilisation interne de WFA.	Aucune

Informations connexes

Références à du matériel d'apprentissage

Instructions pour les commandes

Vous devez connaître les instructions à suivre pour créer des commandes dans OnCommand Workflow Automation (WFA).

Directives	Exemple
Utiliser un nom facilement identifiable pour les commandes.	`Create Qtree`
Utilisez des espaces pour délimiter les mots et chaque mot doit commencer par un caractère en majuscules.	`Create Volume`
Fournissez une description pour expliquer la fonctionnalité de la commande, y compris le résultat attendu des paramètres facultatifs.	Aucune
Par défaut, le délai d'attente des commandes standard est de 600 secondes. Le délai par défaut est défini lors de la création de la commande. Modifiez la valeur par défaut uniquement si la commande peut prendre plus de temps.	`Create Volume` commande
Dans le cas d'opérations de longue durée, créez deux commandes : l'une pour appeler l'opération de longue durée et l'autre pour signaler périodiquement la progression de l'opération. La première commande doit être un `Standard Execution` le type de commande et le second doit être `Wait for Condition` type de commande.	`Create VSM` et `Wait for VSM` commandes
Préfixer le `Wait for condition` Noms de commande avec « attendre » pour une identification facile.	`Wait for CM Volume Move`
Utilisez un intervalle d'attente approprié pour les commandes « attendre condition ». La valeur spécifiée régit l'intervalle d'exécution de la commande d'interrogation pour vérifier si l'opération longue durée est terminée.	intervalle d'échantillonnage de 60 s pour le `Wait for VSM` commande
Pour le `Wait for condition` les commandes, utilisez un délai d'attente approprié en fonction du temps prévu pour l'opération de longue durée. Le temps prévu peut être considérablement plus long si l'opération implique un transfert de données sur un réseau.	Un transfert de ligne de base VSM peut prendre plusieurs jours. Par conséquent, le délai spécifié est de 6 jours.

Représentation de chaîne

La représentation de chaîne d'une commande affiche les détails d'une commande dans une conception de flux de travail lors de la planification et de l'exécution. Seuls les paramètres de commande peuvent être utilisés dans la représentation de chaîne d'une commande.

Directives	Exemple
Évitez d'utiliser des attributs qui n'ont aucune valeur. Un attribut sans valeur s'affiche sous la forme NA.	Volname 10.68.66.212[NA]aggr1/testVol7
Séparez les différentes entrées de la représentation de chaîne à l'aide des délimiteurs suivants : [ ] , / :	*ArrayName[ArrayIp]*
Fournir des étiquettes significatives à chaque valeur dans la représentation de chaîne.	*Volume name=VolumeName*

Langage de définition de commande

Les commandes peuvent être écrites à l'aide des langages de script pris en charge suivants :

- PowerShell
- Perl

Définition du paramètre de commande

Les paramètres de la commande sont décrits par Nom, Description, Type, une valeur par défaut pour le paramètre et si le paramètre est obligatoire. Le type de paramètre peut être String, Boolean, Integer, long, Double, Enum, DateTime, Capacity, Array, Hashtable, Mot de passe ou XmlDocument. Bien que les valeurs de la plupart des types soient intuitives, les valeurs de Array et Hashtable doivent être dans un format particulier, comme décrit dans le tableau suivant :

Directives	Exemple
Assurez-vous que la valeur d'un type d'entrée Tableau est une liste de valeurs séparées par des virgules.	`[parameter(Mandatory=$false, HelpMessage="Months in which the schedule executes.")] [array]$CronMonths` L'entrée est passée comme suit : 0,3,6,9
Assurez-vous que la valeur d'un type d'entrée Hashtable est une liste de paires clé=valeur, séparées par un point-virgule.	`[parameter(Mandatory=$false, HelpMessage="Volume names and size (in MB)")] [hashtable]$VolumeNamesAndSize` L'entrée est transmise comme suit : Volume1=100;Volume2=250;Volume3=50

Instructions pour les flux de travail

Vous devez connaître les instructions de création ou de modification d'un workflow prédéfini pour OnCommand Workflow Automation (WFA).

Directives générales

Directives	Exemple
Nommez le flux de travail de sorte qu'il reflète l'opération exécutée par l'opérateur de stockage.	`Create a CIFS Share`
Pour les noms de flux de travail, mettez en majuscule la lettre initiale du premier mot et chaque mot qui est un objet. Lettres de majuscule pour les abréviations et les acronymes.	Volumétrie Qtree Créez un partage CIFS en qtree Data ONTAP
Pour les descriptions de flux de travail, incluez toutes les étapes importantes du flux de travail, y compris les prérequis, le résultat du flux de travail ou les aspects conditionnels de l'exécution.	Voir la description de l'exemple de flux de travail `Create VMware NFS Datastore on Clustered Data ONTAP Storage`, qui inclut les prérequis.
Définissez « prêt pour la production » sur `true` uniquement lorsque le flux de travail est prêt pour la production et peut être affiché sur la page portail.	Aucune
Par défaut, définissez « considérer les éléments réservés » sur vrai. Lors de l'aperçu d'un flot de travaux pour exécution, WFA Planner prend en compte tous les objets réservés avec les objets existants dans la base de données de cache. Les effets d'autres flux de travail planifiés ou de flux de travail s'exécutant en parallèle sont pris en compte lors de la planification d'un flux de travail spécifique si cette option est définie sur `true`.	• Scénario 1 Le flux de travail 1 crée un volume et est programmé pour l'exécuter une semaine plus tard. Il crée des qtrees ou des LUN dans des volumes recherchés. Si le workflow 2 est exécuté en une journée environ, vous devez désactiver « considérer les éléments réservés » pour le workflow 2 afin d'éviter qu'il ne considère le volume à créer en une semaine. • Scénario 2 Le flux de travail 1 utilise le `Create Volume` commande. Si un flux de travail planifié 2 consomme 100 Go d'un agrégat, celui-ci doit prendre en compte les exigences du flux de travail 2 lors de la planification.

Directives	Exemple
Par défaut, « Activer la validation de l'existence d'élément » est défini sur `true`.	• Scénario 1 Si vous créez un flux de travail qui supprime d'abord un volume par son nom à l'aide de la commande `Remove Volume` ce n'est que si le volume existe et que le recrée à l'aide d'une autre commande telle que `Create Volume` ou `Clone Volume`, le flux de travail ne doit alors pas utiliser cet indicateur. L'effet de la suppression du volume ne sera pas disponible pour le `Create volume` entraînant l'échec du workflow. • Scénario 2 Le `Create Volume` la commande est utilisée dans un workflow portant un nom spécifique « vol198 ». Si cette option est définie sur true, WFA Planner vérifie lors de la planification pour vérifier si un volume dont ce nom existe dans la baie donnée. Si le volume existe, le flux de travail échoue pendant la planification.
Lorsque la même commande est sélectionnée plusieurs fois dans un flux de travail, indiquez les noms d'affichage appropriés pour les instances de commande.	L'exemple de workflow « Créer, mapper et protéger des LUN avec SnapVault » utilise le `Create Volume` commande deux fois. Toutefois, il utilise les noms d'affichage comme `Create Primary Volume` et `Create Secondary Volume` adapté au volume primaire et au volume de destination en miroir.

Entrées utilisateur

Directives	Exemple

Noms :
 	
• Commencez le nom par le caractère «»$».
• Utilisez une lettre majuscule au début de chaque mot.
• Utilisez des lettres majuscules pour tous les termes et abréviations.
• Ne pas utiliser de trait de soulignement.
	`$Array`
	
	`$VolumeName`

Directives	Exemple

Noms d'affichage :
 	
• Utilisez une lettre majuscule au début de chaque mot.
• Séparez les mots par des espaces.
• Si les entrées ont des unités spécifiques, spécifiez l'unité entre crochets dans le nom d'affichage directement.
	`Volume Name`
	
	`Volume Size (MB)`

Descriptions : • Fournissez une description pertinente pour chaque entrée utilisateur. • Fournissez des exemples lorsque cela est nécessaire. Vous devez le faire particulièrement lorsque la saisie utilisateur doit avoir un format spécifique. Les descriptions des entrées utilisateur sont affichées sous forme d'info-bulles pour les entrées utilisateur lors de l'exécution du workflow.	Initiateurs à ajouter à un « iGroup ». Par exemple, IQN ou WWPN de l'initiateur.
Type : sélectionnez Enum comme type si vous souhaitez limiter l'entrée à un ensemble spécifique de valeurs.	Protocole : « iscsi », « fcp », « mixed »

Type : sélectionnez Query comme type lorsque l'utilisateur peut sélectionner parmi les valeurs disponibles dans le cache WFA.	$Array : type DE REQUÊTE avec requête comme suit :
	
	`SELECT`
	`  ip, name`
	`FROM`
	`  storage.array`

Type : permet de marquer l'entrée utilisateur comme verrouillée lorsque l'entrée utilisateur doit être limitée aux valeurs obtenues à partir d'une requête ou uniquement aux types d'Enum pris en charge.	$Array: Locked Query type: Seules les matrices du cache peuvent être sélectionnées.$Protocol: Locked Enum type avec des valeurs valides iSCSI, fcp, mixte. Aucune autre valeur que la valeur valide n'est prise en charge.
Type : Query TypeAjoutez des colonnes supplémentaires en tant que valeurs de retour dans la requête lorsqu'il aide l'opérateur de stockage à faire le bon choix d'entrée utilisateur.	$aggrate : indiquez le nom, la taille totale, la taille disponible pour que l'opérateur connaisse les attributs avant de sélectionner l'agrégat.

Directives	Exemple

Type : la requête TypeSQL pour les entrées utilisateur peut faire référence à toute autre entrée utilisateur qui la précède. Il peut être utilisé pour limiter les résultats d'une requête basée sur d'autres entrées utilisateur telles que les unités vFiler d'une baie, les volumes d'un agrégat ou les LUN d'un SVM (Storage Virtual machine).	Dans l'exemple de flux de travail `Create a Clustered Data ONTAP Volume`, La requête de VserverName est la suivante :
	
	`SELECT`
	`  vserver.name`
	`FROM`
	`  cm_storage.cluster cluster,`
	`  cm_storage.vserver vserver`
	`WHERE`
	`  vserver.cluster_id = cluster.id`
	`  AND cluster.name = '${ClusterName}'`
	`  AND vserver.type = 'cluster'`
	`ORDER BY`
	`  vserver.name ASC`
	
	La requête fait référence à ${clustername}, où $clustername est le nom de l'entrée utilisateur précédant l'entrée utilisateur $VserverName.

Type : utilisez le type booléen avec des valeurs comme « vrai, faux » pour les entrées utilisateur qui sont de nature booléenne. Cela permet d'écrire des expressions internes dans la conception du flux de travail à l'aide de l'entrée utilisateur directement. Par exemple, $UserInputName plutôt que $UserInputName == "Oui'.	`$CreateCIFSShare`: Type booléen avec des valeurs valides comme « vrai » ou « faux »
Type:pour le type de chaîne et de nombre, utilisez des expressions régulières dans la colonne valeurs pour valider la valeur avec des formats spécifiques. Utilisez des expressions régulières pour les entrées d'adresse IP et de masque réseau.	L'entrée utilisateur spécifique à un emplacement peut être exprimée comme « »[A-Z][A-Z]\-0[1-9]». Cette entrée utilisateur accepte des valeurs telles que « US-01 », « NB-02 », mais pas « nb-00 ».
Type : pour le type de nombre, une validation basée sur une plage peut être spécifiée dans la colonne valeurs.	Pour le nombre de LUN à créer, l'entrée de la colonne valeurs est 1-20.
Groupe : regroupe les entrées utilisateur associées dans les compartiments appropriés et nommez le groupe.	« Détails de stockage » pour toutes les entrées utilisateur liées au stockage. "`détails du magasins'" pour toutes les entrées utilisateur relatives à VMware.

Directives	Exemple
Obligatoire : si la valeur d'une entrée utilisateur est nécessaire pour que le flux de travail s'exécute, marquez l'entrée utilisateur comme obligatoire. Cela permet de s'assurer que l'écran de saisie de l'utilisateur accepte cette entrée de la part de l'utilisateur.	« »$VolumeName » dans le workflow « Create NFS Volume ».
Valeur par défaut : si une entrée utilisateur a une valeur par défaut qui peut fonctionner pour la plupart des exécutions de flux de travail, fournissez les valeurs. Cela permet à l'utilisateur de fournir moins d'entrées lors de l'exécution, si la valeur par défaut sert le but.	Aucune

Constantes, variables et renvoie les paramètres

Directives	Exemple
Constantes : définissez des constantes lors de l'utilisation d'une valeur commune pour la définition de paramètres sur plusieurs commandes.	*AGGREGATE_OVERENGAGEMENT_THRESHOLD* dans l' `Create, map, and protect LUNs with SnapVault sample workflow`.
Constantes:noms • Utilisez une lettre majuscule au début de chaque mot. • Utilisez des lettres majuscules pour tous les termes et abréviations. • Ne pas utiliser de trait de soulignement. • Utilisez des lettres majuscules pour toutes les lettres de noms constants.	*AGGREGATE_USED_SPACE_THRESHOLD* *ActualVolumeSizeInMB*
Variables : fournissez un nom à un objet défini dans l'une des zones de paramètres de commande. Les variables sont générées automatiquement et peuvent être modifiées.	Aucune
Variables : les noms utilisent des caractères minuscules pour les noms de variables.	volume1 partage cifs

Paramètres de retour : utilisez les paramètres de retour lorsque la planification et l'exécution du flux de travail doivent renvoyer certaines valeurs calculées ou sélectionnées pendant la planification. Les valeurs sont disponibles en mode aperçu lorsque le flux de travail est exécuté à partir d'un service Web également.	Agrégat : si l'agrégat est sélectionné à l'aide de la logique de sélection des ressources, alors l'agrégat sélectionné réel peut être défini comme paramètre de retour.

Instructions pour la création de scripts de validation pour les types de systèmes distants

Vous devez connaître les instructions permettant de créer des scripts de validation utilisés pour tester les types de systèmes distants que vous définissez dans OnCommand Workflow Automation (WFA).

- Le script Perl que vous créez doit être similaire à l'exemple de script fourni dans la fenêtre script de validation.
- Le résultat de votre script de validation doit être similaire à celui de l'exemple de script.

Exemple de script de validation

```
# Check connectivity.
# Return 1 on success.
# Return 0 on failure and set $message
sub checkCredentials {
my ($host, $user, $passwd, $protocol, $port, $timeout) = @_;
#
# Please add the code to check connectivity to $host using $protocol here.
#
return 1;
}
```

Instructions pour la création de types de sources de données

Vous devez connaître les consignes de création de types de sources de données utilisées pour définir des sources de données personnalisées pour OnCommand Workflow Automation (WFA).

Vous pouvez définir un type de source de données à l'aide de l'une des méthodes suivantes :

- SQL : vous pouvez utiliser les instructions de WFA SQL pour définir des requêtes de sélection à partir de sources de données basées sur une base de données externe.
- SCRIPT : vous pouvez écrire un script PowerShell qui fournit les données d'un schéma spécifique d'entrées de dictionnaire.

Les instructions de création de types de sources de données sont les suivantes :

- Vous devez utiliser le langage PowerShell pour créer un script.
- Le script PowerShell doit fournir la sortie de chaque entrée de dictionnaire dans son répertoire de travail courant.
- Les fichiers de données doivent être nommés `dictionary_entry.csv,` où le nom de l'entrée du dictionnaire doit être en caractères minuscules.

  Le type de source de données prédéfini qui collecte les informations de Performance Advisor utilise un type de source de données BASÉ SUR DES SCRIPTS. Les fichiers de sortie sont nommés `array_performance.csv` et `aggregate_performance.csv.`

- Le `.csv` le fichier doit inclure le contenu dans l'ordre exact des attributs d'entrée du dictionnaire.

  Une entrée de dictionnaire inclut des attributs dans l'ordre suivant : Array_ip, date, jour, heure, cpu_Busy, total_ops_per_sec, débit_disque_par_sec

  Le script PowerShell ajoute des données au `.csv` fichier dans le même ordre.

  ```
  $values = get-Array-CounterValueString ([REF]$data)
  Add-Content $arrayFile ([byte[]][char[]] "\N
  t$arrayIP't$date't$day't$hour't$values'n")
  ```

- Vous devez utiliser le codage pour vous assurer que les données issues du script sont correctement chargées dans le cache WFA.
- Vous devez utiliser \N lors de la saisie d'une valeur nulle dans `.csv` fichier.