Directrices de codificación para WFA

OnCommand Workflow Automation 5.0

NetApp
April 19, 2024

Tabla de contenidos

Directrices de codificación para WFA

Debería comprender las directrices de codificación general de OnCommand Workflow Automation (WFA), las convenciones de nomenclatura y las recomendaciones sobre la creación de elementos básicos como filtros, funciones, comandos y flujos de trabajo.

Pautas para las variables

Debe tener en cuenta las directrices para PowerShell y las variables Perl de OnCommand Workflow Automation (WFA) cuando cree un comando o un tipo de origen de datos.

Variables de PowerShell

Directrices	Ejemplo

Para parámetros de entrada de script:
• Utilice la caja Pascal.
• No utilice guiones bajos.
• No utilice abreviaturas.
	`$VolumeName`
	`$AutoDeleteOptions`
	`$Size`

Para variables internas de script:
• Utilice la funda Camel.
• No utilice guiones bajos.
• No utilice abreviaturas.
	`$newVolume`
	`$qtreeName`
	`$time`

Para funciones: • Utilice la caja Pascal. • No utilice guiones bajos. • No utilice abreviaturas.	`GetVolumeSize`
Los nombres de variables no distinguen entre mayúsculas y minúsculas. Sin embargo, para mejorar la legibilidad, no debe utilizar mayúsculas diferente para el mismo nombre.	`$variable` es igual que `$Variable`.
Los nombres de variables deben estar en inglés sin formato y deben estar relacionados con la funcionalidad del script.	Uso `$name` y no `$a`.
Declare explícitamente el tipo de datos de cada variable.	[cadena]nombre tamaño [int]

Directrices	Ejemplo
No utilice caracteres especiales (! @ # & % , .) y espacios de aplicaciones.	Ninguno
No utilice palabras clave reservadas de PowerShell.	Ninguno

Agrupe los parámetros de entrada colocando primero los parámetros obligatorios seguidos de los parámetros opcionales.	`param(`
	`[parameter(Mandatory=$true)]`
	`[string]$Type,`
	
	`[parameter(Mandatory=$true)]`
	`[string]$Ip,`
	
	`[parameter(Mandatory=$false)]`
	`[string]$VolumeName`
	`)`

Comentar todas las variables de entrada utilizando la anotación HelpMessage con un mensaje de ayuda significativo.	`[parameter(Mandatory=$false,HelpMessage="LUN to map")]` `[string]$LUNName`
No utilice «'Archivador'» como nombre de variable; utilice «'Array'» en su lugar.	Ninguno

Utilice la anotación ValidateSet en casos en los que el argumento obtiene valores enumerados. Esto se traduce automáticamente al tipo de datos Enum para el parámetro.	`[parameter(Mandatory=$false,HelpMessage="Volume state")]`
	`[ValidateSet("online","offline","restricted")]`
	`[string]$State`

Agregue un alias a un parámetro que termine con "'_Capacity'" para indicar que el parámetro es de tipo capacidad.	El comando «'Create Volume'» utiliza alias de la siguiente forma:
	
	`[parameter(Mandatory=$false,HelpMessage="Volume increment size in MB")]`
	`[Alias("AutosizeIncrementSize_Capacity")]`
	`[int]$AutosizeIncrementSize`

Directrices	Ejemplo

Agregue un alias a un parámetro que termine con "'_Password'" para indicar que el parámetro es de tipo de contraseña.	`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

Directrices	Ejemplo

Para parámetros de entrada de script:
• Utilice la caja Pascal.
• No utilice guiones bajos.
• No utilice abreviaturas.
	`$VolumeName`
	`$AutoDeleteOptions`
	`$Size`

No utilice abreviaturas para las variables internas del script.	`$new_volume`
	`$qtree_name`
	`$time`

No utilice abreviaturas para las funciones.	`get_volume_size`
Los nombres de variables distinguen mayúsculas de minúsculas. Para mejorar la legibilidad, no debe utilizar mayúsculas diferente para el mismo nombre.	`$variable` no es lo mismo que `$Variable`.
Los nombres de variables deben estar en inglés sin formato y deben estar relacionados con la funcionalidad del script.	Uso `$name` y no `$a`.
Agrupe los parámetros de entrada colocando primero los parámetros obligatorios, seguidos de los parámetros opcionales.	Ninguno

Directrices	Ejemplo

En la función GetOptions, declare explícitamente el tipo de datos de cada variable para los parámetros de entrada.	`GetOptions(`
	`"Name=s"=>\$Name,`
	`"Size=i"=>\$Size`
	`)`

No utilice «'Archivador'» como nombre de variable; utilice «'Array'» en su lugar.	Ninguno

Perl no incluye la `ValidateSet` anotación para valores enumerados. Utilice declaraciones explícitas «'if'» para casos en los que el argumento obtenga valores enumerados.	`if`
	`(defined$SpaceGuarantee&&!($SpaceG`
	`uaranteeeq'none'`

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

Todos los comandos Perl WFA deben utilizar el pragma "strict" para desalentar el uso de construcciones inseguras para variables, referencias y subrutinas.	`use strict;`
	`# the above is equivalent to`
	`use strictvars;`
	`use strictsubs;`
	`use strictrefs;`

Todos los comandos Perl de WFA deben utilizar los siguientes módulos Perl:
• Getopt
Se utiliza para especificar parámetros de entrada.
• WFAUtil
Esto se utiliza para las funciones de utilidad que se proporcionan para el registro de comandos, la generación de informes sobre el progreso de comandos, la conexión con las controladoras de la cabina, etc.
	`use Getopt::Long;`
	`use NaServer;`
	`use WFAUtil;`

Directrices para indentación

Debe tener en cuenta las directrices para la sangría cuando se escribe un script de

PowerShell o Perl para OnCommand Workflow Automation (WFA).

Directrices	Ejemplo
Una pestaña es igual a cuatro espacios vacíos.	

Utilice tabulaciones y llaves para mostrar el principio y el final de un bloque.	Script de 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;`
	`}`

Agregue líneas en blanco entre conjuntos de operaciones o fragmentos de código.	`$options=$option.trim();`
	`$pair=$option.split(" ");`
	`Get-WFAlogger -Info -messages`
	`$("split options: "+`
	`$Pair)`

Directrices para comentarios

Debe tener en cuenta las directrices para PowerShell y comentarios Perl en sus secuencias de comandos para OnCommand Workflow Automation (WFA).

Comentarios sobre PowerShell

Directrices	Ejemplo
Utilice el carácter # para un comentario de una sola línea.	`# Single line comment` `$options=$option.trim();`
Utilice el carácter # para un comentario de fin de línea.	`$options=$option.trim(); # End of line` `comment`

Utilice los caracteres <# y #> para un comentario de bloque.	`<#`
	`This is`
	`a`
	`block comment`
	`#>`
	`$options=$option.trim();`

Comentarios Perl

Directrices	Ejemplo

Utilice el carácter # para el comentario de una línea.	`# convert from MBytes to Bytes`
	`my $MaxDirectorySizeBytes =`
	`$MaxDirectorySize *`
	`1024 * 1024;`

Utilice el carácter # para el comentario de fin de línea.	`my $MaxDirectorySizeBytes =`
	`$MaxDirect`
	`orySiZe * 1024 * 1024; # convert`
	`to Bytes`

Directrices	Ejemplo

Utilice el carácter # en cada línea con un # vacío al principio y al final para crear un borde de comentario para comentarios de varias líneas.	`#`
	`# 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`
	`#`

No incluya código inactivo y comentado en los comandos de WFA. Sin embargo, para realizar pruebas, puede utilizar el mecanismo de documentación antigua simple (POD) para comentar el código.	`=begin comment`
	`  # Set deduplication`
	`  if(defined $Deduplication && $Deduplication eq "enabled")`
	`  {`
	`  $wfaUtil->sendLog("Enabling Deduplication");`
	`  }`
	`=end comment`
	`=cut`

Directrices para el registro

Debe conocer las directrices para iniciar sesión al escribir un script PowerShell o Perl para OnCommand Workflow Automation (WFA).

Registro de PowerShell

Directrices	Ejemplo
Utilice el cmdlet Get-WFALogger para el registro.	`Get-WFALogger -Info -message “Creating volume”`

Directrices	Ejemplo
Registre todas las acciones que requieran la interacción con paquetes internos como Data ONTAP, VMware y PowerCLI. Todos los mensajes de registro están disponibles en registros de ejecución en el historial de estado de ejecución de flujos de trabajo.	Ninguno
Registre todos los argumentos relevantes que se pasan a paquetes internos.	Ninguno
Utilice los niveles de registro adecuados cuando utilice el cmdlet Get-WFALogger, en función del contexto de uso. -Info, -error, -warn y -Debug son los distintos niveles de registro disponibles. Si no se especifica un nivel de registro, el nivel de registro predeterminado es Debug.	Ninguno

Registro Perl

Directrices	Ejemplo

Utilice WFAUtil SendLog para el registro.	`my wfa_util = WFAUtil->new();`
	`eval {`
	`$wfa_util->sendLog('INFO',`
	`"Connecting to the`
	`cluster: $DestinationCluster");`
	`}`

Registrar cada acción que requiere la interacción con cualquier elemento externo al comando, como Data ONTAP, VMware y WFA. Todos los mensajes de registro que cree con la rutina WFAUtil SendLog se almacenan en la base de datos WFA. Estos mensajes de registro están disponibles para el flujo de trabajo y el comando ejecutados.	Ninguno
Registre cada argumento relevante pasado a la rutina que se llamó.	Ninguno
Use los niveles de registro adecuados.-Info, -error, -warn y -Debug son los distintos niveles de registro disponibles.	Ninguno

Directrices	Ejemplo

Cuando inicie sesión en el nivel -Info, sea preciso y conciso. No especifique detalles de implementación como el nombre de clase y el nombre de función en los mensajes de registro. Describa el paso exacto o el error exacto en inglés normal.	El siguiente fragmento de código muestra un ejemplo de un mensaje correcto y un mensaje erróneo:
	`$wfa_util->sendLog('WARN', "Removing volume: '.$VolumeName); # Good Message`
	`$wfa_util->sendLog('WARN', 'Invoking volume-destroy ZAPI: '.$VolumeName); # Bad message`

Directrices para la gestión de errores

Debe tener en cuenta las directrices para la gestión de errores al escribir un script PowerShell o Perl para OnCommand Workflow Automation (WFA).

Gestión de errores de PowerShell

Directrices	Ejemplo

Los parámetros comunes añadidos a los cmdlets de PowerShell Runtime incluyen parámetros de control de errores como ErrorAction y WarningAction:
• El parámetro ErrorAction determina la forma en que un cmdlet debe reaccionar ante un error que no es de terminación del comando.
• El parámetro WarningAction determina cómo debe reaccionar un cmdlet a una advertencia del comando.
• Stop, SilentlyContinue, Inquire y Continue son los valores válidos para los parámetros ErrorAction y WarningAction.
Para obtener más información, puede utilizar la `Get-Help about_CommonParameters` En la CLI de PowerShell.
	ErrorAction: El ejemplo siguiente muestra cómo manejar un error que no es de terminación como un error de terminación:
	`New-NcIgroup-Name $IgroupName-Protocol $Protocol-Type$OSType-ErrorActionstop`
	WarningAction
	`New-VM-Name $VMName-VM $SourceVM-DataStore$DataStoreName-VMHost$VMHost-WarningActionSilentlyContinue`

Directrices	Ejemplo

Utilice la instrucción general "'try/catch'" si no se conoce el tipo de excepción entrante.	`try`
	`{`
	`"In Try/catch block"`
	`}`
	`catch`
	`{`
	`"Got exception"`
	`}`

Utilice la instrucción específica «'try/catch'» si conoce el tipo de excepción entrante.	`try`
	`{`
	`"In Try/catch block"`
	`}`
	`catch[System.Net.WebExceptional],`
	`[System.IO.`
	`IOException]`
	`{`
	`"Got exception"`
	`}`

Utilice la declaración «'finally'» para liberar recursos.	`try`
	`{`
	`"In Try/catch block"`
	`}`
	`catch`
	`{`
	`"Got exception"`
	`}`
	`finally`
	`{`
	`"Release resources"`
	`}`

Directrices	Ejemplo

Utilice variables automáticas de PowerShell para acceder a la información acerca de excepciones.	`try`
	`{`
	`Get-WFALogger -Info -message`
	`$("Creating`
	`Ipspace: " + $Ipspace)`
	`New-NaNetIpspace-Name $Ipspace`
	`}`
	`catch`
	`{`
	`Throw "Failed to create Ipspace.`
	`Message:`
	`" + $_.Exception.Message;`
	`}`

Gestión de errores Perl

Directrices	Ejemplo

Perl no incluye compatibilidad con el idioma nativo para bloques try/catch. Utilice bloques de evaluación para comprobar y manejar errores. Mantenga los bloques de evaluación lo más pequeños posible.	`eval {`
	
	`$wfa_util->sendLog('INFO',`
	`"Quiescing the relationship :`
	`$DestinationCluster://$Destination`
	`Vserver`
	`/$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://$Destination`
	`Vserver`
	`/$DestinationVolume",`
	`$@`
	`);`

PowerShell general y convenciones Perl para WFA

Debe comprender ciertas convenciones de PowerShell y Perl que se utilizan en WFA para crear scripts consistentes con los scripts existentes.

- Utilice variables que le ayudarán a aclarar lo que desea que haga el script.
- Escriba un código legible que se pueda entender sin comentarios.
- Mantenga las secuencias de comandos y comandos tan simples como sea posible.
- Para scripts de PowerShell:
  - Use los cmdlets siempre que sea posible.
  - Invoque el código .NET cuando no haya ningún cmdlet disponible.
- Para scripts Perl:

- Termine siempre las declaraciones "da" con caracteres de nueva línea.

  En ausencia de un carácter de salto de línea, se imprime el número de línea de script, lo cual no es útil para depurar comandos Perl ejecutados por WFA.

- En el módulo «'getopt'», haga que los argumentos de cadena sean obligatorios en un comando.

Módulos Perl con Windows

Algunos módulos Perl están integrados con la distribución Perl de Windows Active state para OnCommand Workflow Automation (WFA). Puede utilizar estos módulos Perl en su código Perl para escribir comandos, solo si se combinan con Windows.

La siguiente tabla enumera los módulos de base de datos Perl que se combinan con Windows para WFA.

Módulo de base de datos	Descripción
DBD::mysql	Controlador de interfaz de base de datos Perl5 que permite conectarse a la base de datos MySQL.
Pruebe::diminuto	Minimiza los errores comunes con bloques de evaluación.
XML::libxml	Interfaz para libxml2 que proporciona a los analizadores XML y HTML con interfaces DOM, SAX y XMLReader.
DBD::Cassandra	Controlador de interfaz de base de datos Perl5 para Cassandra que utiliza el lenguaje de consulta CQL3.

Consideraciones sobre la adición de PowerShell y módulos Perl personalizados

Debe tener en cuenta diferentes consideraciones antes de agregar PowerShell personalizado y módulos Perl a OnCommand Workflow Automation (WFA). Los módulos Perl y PowerShell personalizados permiten usar comandos personalizados para crear flujos de trabajo.

- Durante la ejecución de comandos de WFA, todos los módulos personalizados de PowerShell se añaden al directorio de instalación de WFA */Posh/modules* se importan automáticamente.
- Todos los módulos Perl personalizados añadidos al directorio *WFA/perl* se incluyen en la biblioteca @*INC*.
- No se realiza un backup de los módulos Perl y PowerShell personalizados como parte de la operación de backup de WFA.
- Los módulos Perl y PowerShell personalizados no se restauran como parte de la operación de restauración de WFA.

Debe realizar manualmente la copia de seguridad de los módulos de PowerShell y Perl personalizados para copiarlos en una nueva instalación de WFA.

El nombre de la carpeta del directorio de módulos debe ser el mismo que el del nombre del módulo.

Cmdlets y funciones DE WFA

OnCommand Workflow Automation (WFA) proporciona varios cmdlets de PowerShell, así como funciones PowerShell y Perl que puede utilizar en sus comandos WFA.

Puede ver todos los cmdlets y las funciones de PowerShell que proporciona el servidor de WFA utilizando los siguientes comandos de PowerShell:

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

Puede ver todas las funciones Perl proporcionadas por el servidor WFA en el `WFAUtil.pm` módulo. Las secciones de ayuda, los cmdlets de WFA PowerShell ayudan y los métodos Perl de la ayuda del módulo de ayuda WFA Support Links permite el acceso a todos los cmdlets y funciones de PowerShell y a las funciones Perl.

Módulos de WFA PowerShell y Perl

Debe tener en cuenta la PowerShell o los módulos Perl de OnCommand Workflow Automation (WFA) para escribir scripts para sus flujos de trabajo.

Módulos de PowerShell

Directrices	Ejemplo
Utilice el kit de herramientas PS de Data ONTAP para invocar API siempre que el kit de herramientas esté disponible.	La `Add VLAN` command utiliza el kit de herramientas de la siguiente manera: `Add-NaNetVlan-Interface $Interface-Vlans$VlanID`
Si no hay cmdlets disponibles en el Kit de herramientas de PS de Data ONTAP, utilice el comando Invoke-SSH para invocar la CLI en Data ONTAP.	`Invoke-NaSsh-Name $ArrayName-Command "ifconfig -a"-Credential $Credentials`

Módulos Perl

El módulo NaServer se utiliza en los comandos WFA. El módulo NaServer permite la invocación de API Data ONTAP, que se utilizan en la administración activa de sistemas Data ONTAP.

Directrices	Ejemplo
Utilice el módulo NaServer para invocar las API siempre que esté disponible el SDK de capacidad de gestión de NetApp.	El siguiente ejemplo muestra cómo se utiliza el módulo NaServer para reanudar la operación 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',
```

Directrices	Ejemplo
Si no hay disponible una API de Data ONTAP, invoque la CLI de Data ONTAP mediante el método executeSysteminterd. ExecuteSystemS4 no es compatible y actualmente sólo está disponible para Data ONTAP en 7-Mode.	Ninguno

Consideraciones que tener en cuenta al convertir comandos de PowerShell en Perl

Debe tener en cuenta diferentes consideraciones al convertir los comandos de PowerShell en Perl, porque PowerShell y Perl tienen diferentes funcionalidades.

Tipos de entrada de comandos

OnCommand Workflow Automation (WFA) permite a los diseñadores de flujos de trabajo utilizar cabinas y hash como entrada para el comando al definir un comando. Estos tipos de entrada no se pueden utilizar cuando el comando se define usando Perl. Si desea que un comando Perl acepte entradas array y hash, puede definir la entrada como una cadena en el diseñador. La definición de comandos puede analizar la entrada, que se pasa para crear una matriz o hash según sea necesario. La descripción de la entrada describe el formato en el que se espera la entrada.

```
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.
```

Declaración de PowerShell

Los siguientes ejemplos muestran cómo puede pasar una entrada de cabina a PowerShell y Perl. En los ejemplos se describe la entrada CronMonth, que especifica el mes en el que está programado que se ejecute el trabajo cron. Los valores válidos son números enteros -1 a 11. Un valor de -1 indica que la programación se ejecuta cada mes. Cualquier otro valor denota un mes específico, siendo 0 enero y 11 diciembre.

```
[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,
```

Instrucción 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
}
```

Definición de comandos

Una expresión de una línea en PowerShell que use un operador de canalización puede tener que ampliarse a múltiples bloques de sentencias en Perl para lograr la misma funcionalidad. En la siguiente tabla se muestra un ejemplo de uno de los comandos de espera.

Declaración de PowerShell	Instrucción 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)`
	`  }`
	`}`

Directrices para los elementos básicos de WFA

Debe conocer las directrices para usar los elementos básicos de Workflow Automation.

Directrices para SQL en WFA

Debe estar al tanto de las directrices para usar SQL en OnCommand Workflow Automation (WFA) a fin de escribir consultas SQL para WFA.

SQL se utiliza en las siguientes ubicaciones de WFA:

- Consultas SQL para rellenar las entradas del usuario para su selección
- Consultas SQL para crear filtros para filtrar objetos de un tipo de entrada de diccionario específico
- Datos estáticos en tablas en la base de datos de juegos
- Tipo de origen de datos personalizado de tipo SQL en el que los datos deben extraerse de un origen de datos externo, como una base de datos de administración de configuración personalizada (CMDB).
- Consultas SQL para secuencias de comandos de reserva y verificación

Directrices	Ejemplo

Las palabras clave reservadas de SQL deben tener caracteres en mayúsculas.	`SELECT`
	`  vserver.name`
	`FROM`
	`  cm_storage.vserver vserver`

Los nombres de tablas y columnas deben tener caracteres en minúsculas.	Tabla: Agregado Columna: Used_Space_mb
Separe las palabras con un carácter de subrayado (_). No se permiten espacios.	rendimiento_de_cabina
El nombre de la tabla se define en singular. Una tabla es un conjunto de una o más entradas.	«'function'», no «'functions'»

Directrices	Ejemplo

Utilice alias de tabla con nombres significativos en consultas SELECCIONADAS.	`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`

Directrices	Ejemplo

Si tiene que hacer referencia a un parámetro de entrada de filtro o a un parámetro de entrada de usuario en una consulta de filtro o una consulta de usuario, utilice la sintaxis como '${inputVariableName}.también puede utilizar la sintaxis para hacer referencia a un parámetro de definición de comandos en secuencias de comandos de reserva y secuencias de comandos de verificación.	`SELECT`
	`  volume.name AS Name,`
	`  aggregate.name as Aggregate,`
	`  volume.size_mb AS 'Total Size (MB)',`
	`  voulme.used_size_mb AS 'Used Size (MB)',`
	`  volume.space_guarantee AS 'Space Guarantee'`
	`FROM`
	`  cm_storage.cluster,`
	`  cm_storage.aggregate,`
	`  cm_storage.vserver,`
	`  cm_storage.volume`
	`WHERE`
	`  cluster.id = vserver.cluster_id`
	`  AND aggregate.id = volume.aggregate_id`
	`  AND vserver.id = voulme.vserver_id`
	`  AND vserver.name = '${VserverName}'`
	`  AND cluster.name = '${ClusterName}'`
	`ORDER BY`
	`  volume.name ASC`

Utilice comentarios para consultas complejas. Algunos de los estilos de comentario admitidos en las consultas son los siguientes:
• «»--» hasta el final de la línea
Es obligatorio un espacio después del segundo guión de este estilo de comentario.
• De un carácter «»#» hasta el final de la línea
• De "/*" a la siguiente secuencia "*/"
	`/*`
	`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`

Directrices para las funciones de WFA

Puede crear funciones para encapsular la lógica más compleja y utilizada comúnmente en una función llamada y, a continuación, reutilizar la función como valores de parámetros de comandos o valores de parámetros de filtro en OnCommand Workflow Automation (WFA).

Directrices	Ejemplo
Utilice la caja Camel para un nombre de función.	CalculateVolumeSize
Los nombres de las variables deben estar en inglés normal y relacionados con la funcionalidad de la función.	SplitByDelimiter
No utilice abreviaturas.	CalculateVolumeSize, *not* calcVolSize
Las funciones se definen mediante MVFLEX Expression Language (MVEL).	Ninguno
La definición de la función debe especificarse de acuerdo con las directrices oficiales del lenguaje de programación Java.	Ninguno

Directrices para las entradas del diccionario WFA

Debe conocer las directrices para crear entradas de diccionario en OnCommand Workflow Automation (WFA).

Directrices	Ejemplo
Los nombres de las entradas del diccionario sólo deben contener caracteres alfanuméricos y guiones bajos.	Licencia_clúster Switch_23
Los nombres de las entradas del diccionario deben comenzar con un carácter en mayúsculas. Comience cada palabra en el nombre con un carácter en mayúsculas y separe cada palabra con un guión bajo (_).	Volumen Array_License
Los nombres de atributos de entrada de diccionario no deben incluir el nombre de la entrada de diccionario.	Ninguno
Los atributos y las referencias de una entrada de diccionario deben tener caracteres en minúsculas.	agregado, size_mb

Directrices	Ejemplo
Separe las palabras con un guión bajo. No se permiten espacios.	pool_recursos
Las entradas del diccionario no pueden incluir referencias de un esquema diferente. Cuando una entrada de diccionario requiere referencias cruzadas a un objeto de un esquema diferente, asegúrese de que todas las claves naturales del objeto al que se hace referencia estén presentes en la entrada de diccionario.	La entrada del diccionario Array_Performance requiere todas las claves naturales de la entrada del diccionario Array como atributos directos en él.
Utilice los tipos de datos adecuados para los atributos.	Ninguno
Utilice el tipo de datos Long para los atributos relacionados con el tamaño o el espacio.	Size_mb y available_size_mb en la entrada del diccionario Storage.Volume
Utilice Enum cuando un atributo tenga un conjunto fijo de valores.	raid_TYPE en la entrada del diccionario Storage.Volume
Establezca "'para ser almacenado en caché'" como **verdadero** para un atributo o referencia cuando un origen de datos proporcione valor para ese atributo o referencia. Para el origen de datos de Unified Manager de OnCommand, añada atributos en caché si el origen de datos puede proporcionarlo.	Ninguno
Establezca "'puede ser Nulo'" como **verdadero** si el origen de datos que proporciona el valor para este atributo o referencia puede devolver NULL.	Ninguno
Proporcione una descripción significativa de cada atributo y referencia. La descripción se muestra en los detalles del comando al diseñar un flujo de trabajo.	Ninguno
No utilice "'id'" como nombre de un atributo en las entradas del diccionario. Está reservado para el uso interno de WFA.	Ninguno

Directrices para comandos

Debe tener en cuenta las directrices para crear comandos en OnCommand Workflow

Automation (WFA).

Directrices	Ejemplo
Utilice un nombre fácilmente identificable para los comandos.	`Create Qtree`
Utilice espacios para delimitar palabras y cada palabra debe comenzar con un carácter en mayúscula.	`Create Volume`
Proporcione una descripción para explicar la funcionalidad del comando, incluido el resultado esperado de los parámetros opcionales.	Ninguno
De manera predeterminada, el tiempo de espera para los comandos estándar es de 600 segundos. El tiempo de espera predeterminado se configura al crear el comando. Cambie el valor predeterminado solo si el comando puede tardar más tiempo en completarse.	`Create Volume` comando
En caso de operaciones de ejecución prolongada, cree dos comandos, uno para invocar la operación de ejecución prolongada y otro para informar periódicamente sobre el progreso de la operación. El primer comando debería ser un `Standard Execution` el tipo de comando y el segundo debería ser `Wait for Condition` tipo de comando.	`Create VSM` y.. `Wait for VSM` comandos
Anteponga el `Wait for condition` Nombres de comandos con "'wait'" para facilitar la identificación.	`Wait for CM Volume Move`
Utilizar un intervalo de espera adecuado para los comandos "'wait for condition'". El valor especificado rige el intervalo en el que se ejecuta el comando Polling para comprobar si la operación de ejecución prolongada ha finalizado.	60 intervalo de muestreo para `Wait for VSM` comando
Para la `Wait for condition` comandos, utilice un tiempo de espera apropiado según el tiempo esperado para completar la operación de ejecución prolongada. El tiempo esperado podría ser considerablemente mayor si la operación implica transferencia de datos a través de una red.	Una transferencia de base VSM puede tardar varios días en completarse. Por lo tanto, el tiempo de espera especificado es de 6 días.

Representación de cadena

La representación de cadena de un comando muestra los detalles de un comando en un diseño de flujo de trabajo durante la planificación y ejecución. Sólo se pueden utilizar los parámetros de comando en la

representación de cadena de un comando.

Directrices	Ejemplo
Evite utilizar atributos que no tengan ningún valor. Un atributo sin valor se muestra como NA.	`VolName 10.68.66.212[NA]aggr1/testVol7`
Separar diferentes entradas en la representación de cadena usando los siguientes delimitadores: [ ] , / :	`ArrayName[ArrayIp]`
Proporcione etiquetas significativas a todos los valores de la representación de cadenas.	`Volume name=VoumeName`

Lenguaje de definición de comandos

Los comandos se pueden escribir utilizando los siguientes lenguajes de secuencias de comandos compatibles:

- PowerShell
- Perl

Definición de parámetros de comando

Los parámetros de comando se describen por Nombre, Descripción, Tipo, valor predeterminado del parámetro y si el parámetro es obligatorio. El tipo de parámetro puede ser String, Boolean, Integer, Long, Double, Enumeración, DateTime, capacidad, matriz, Hashtable, Una contraseña o un XmlDocument. Aunque los valores para la mayoría de los tipos son intuitivos, los valores para Array y Hashtable deben tener un formato determinado tal como se describe en la siguiente tabla:

Directrices	Ejemplo
Asegúrese de que el valor de un tipo de entrada Array es una lista de valores, separados por comas.	`[parameter(Mandatory=$false, HelpMessage="Months in which the schedule executes.")] [array]$CronMonths` La entrada se pasa como sigue: 0,3,6,9
Asegúrese de que el valor de un tipo de entrada Hashtable es una lista de pares clave=valor, separados por punto y coma.	`[parameter(Mandatory=$false, HelpMessage="Volume names and size (in MB)")] [hashtable]$VolumeNamesAndSize` La entrada se pasa como sigue: Volume1=100;Volume2=250;Volume3=50

Directrices para flujos de trabajo

Debe tener en cuenta las directrices para crear o modificar un flujo de trabajo predefinido para OnCommand Workflow Automation (WFA).

Directrices generales

Directrices	Ejemplo
Asigne un nombre al flujo de trabajo de modo que refleje la operación que ejecuta el operador de almacenamiento.	`Create a CIFS Share`
En el caso de los nombres de flujo de trabajo, capitalice la letra inicial de la primera palabra y cada palabra que sea un objeto. Capitalice letras para abreviaturas y acrónimos.	Volumen Qtree Cree un recurso compartido CIFS Qtree de Clustered Data ONTAP
En las descripciones del flujo de trabajo, incluya todos los pasos importantes del flujo de trabajo, incluidos los requisitos previos, el resultado del flujo de trabajo o aspectos condicionales de la ejecución.	Consulte la descripción del flujo de trabajo de ejemplo `Create VMware NFS Datastore on Clustered Data ONTAP Storage`, que incluye los requisitos previos.
Establezca "'Ready for Production'" en **true** sólo cuando el flujo de trabajo esté listo para la producción y se pueda mostrar en la página del portal.	Ninguno
De forma predeterminada, establezca "'considerar elementos reservados'" en **verdadero**. Al previsualizar un flujo de trabajo para su ejecución, el planificador de WFA considera todos los objetos que están reservados junto con los objetos existentes en la base de datos de caché. Los efectos de otros flujos de trabajo programados o que se ejecutan en paralelo se tienen en cuenta al planificar un flujo de trabajo específico si esta opción está establecida en **true**.	• Situación 1 El flujo de trabajo 1 crea un volumen y está programado para ejecutarse una semana después. El flujo de trabajo 2 crea qtrees o LUN en los volúmenes en los que se busca y, si el flujo de trabajo 2 se ejecuta en un día aproximadamente, debería desactivar «'considerar elementos reservados» para el flujo de trabajo 2 para impedir que se considere el volumen que debe crearse en una semana. • Situación 2 El flujo de trabajo 1 utiliza la `Create Volume` comando. Si hay un flujo de trabajo programado 2 que consume 100 GB de un agregado, el flujo de trabajo 1 debe tener en cuenta los requisitos para el flujo de trabajo 2 durante la planificación.

Directrices	Ejemplo
De forma predeterminada, "'Habilitar validación de existencia de elementos'" se establece en **verdadero**.	• Situación 1 Si crea un flujo de trabajo que primero quita un volumen por nombre mediante el comando `Remove Volume` solo si hay un volumen y se vuelve a crear con otro comando, como `Create Volume` o. `Clone Volume`, entonces el flujo de trabajo no debe utilizar este indicador. El efecto de eliminar el volumen no estará disponible para el `Create volume` comando, lo que provoca un error en el flujo de trabajo. • Situación 2 La `Create Volume` el comando se utiliza en un flujo de trabajo con un nombre específico denominado «'vol198'». Si esta opción está establecida en **verdadero**, el planificador de WFA comprueba durante la planificación si existe un volumen por ese nombre en la matriz indicada. Si hay un volumen, el flujo de trabajo falla durante la planificación.
Cuando se selecciona el mismo comando más de una vez en un flujo de trabajo, proporcione los nombres de visualización adecuados para las instancias de comandos.	El flujo de trabajo de ejemplo «'Crear, asignar y proteger las LUN con SnapVault'» utiliza `Create Volume` comando dos veces. Sin embargo, utiliza los nombres de visualización como `Create Primary Volume` y.. `Create Secondary Volume` adecuadamente para el volumen primario y el volumen de destino reflejado.

Entradas del usuario

Directrices	Ejemplo

Nombres:
 	
• Inicie el nombre con el carácter « »$».
• Utilice una letra mayúscula al principio de cada palabra.
• Utilice letras mayúsculas para todos los términos y abreviaturas.
• No utilice guiones bajos.
	`$Array`
	
	`$VolumeName`

Directrices	Ejemplo

Nombres para mostrar:
 	
• Utilice una letra mayúscula al principio de cada palabra.
• Separe las palabras con espacios.
• Si las entradas tienen unidades específicas, especifique la unidad entre paréntesis directamente en el nombre de visualización.
	`Volume Name`
	
	`Volume Size (MB)`

Descripciones: • Proporcione una descripción significativa para cada información de usuario. • Proporcione ejemplos cuando sea necesario. Debe hacer esto especialmente cuando se espera que la entrada del usuario esté en un formato específico. Las descripciones de entrada del usuario se muestran como información sobre herramientas para las entradas del usuario durante la ejecución del flujo de trabajo.	Iniciadores que se van a agregar a un «'iGroup'». Por ejemplo, IQN o WWPN del iniciador.
Escriba: Seleccione Enum como el tipo si desea restringir la entrada a un conjunto específico de valores.	Protocolo: «» iscsi», «'fcp», «mixta»

Tipo: Seleccione Query como tipo cuando el usuario pueda seleccionar de entre los valores disponibles en la caché de WFA.	$Array: Tipo DE CONSULTA con la siguiente consulta:
	
	`SELECT`
	`  ip, name`
	`FROM`
	`  storage.array`

Tipo: Marque la entrada del usuario como bloqueada cuando la entrada del usuario deba restringirse a los valores que se obtienen de una consulta o estar restringida sólo a los tipos de Enum admitidos.	$Array: Tipo de consulta bloqueado: Sólo se pueden seleccionar las matrices de la caché.$Protocol: Tipo de Enum bloqueado con valores válidos como iscsi, fcp, mixto. No se admite ningún otro valor distinto del válido.

Directrices	Ejemplo
Tipo: Tipo de query Agregue columnas adicionales como valores devueltos en la consulta cuando ayude al operador de almacenamiento a elegir correctamente la entrada del usuario.	$aggregate: Proporcione nombre, tamaño total y tamaño disponible para que el operador conozca los atributos antes de seleccionar el agregado.

Tipo: Consulta TypeSQL para entradas de usuario puede hacer referencia a cualquier otra entrada de usuario anterior. Esto puede utilizarse para limitar los resultados de una consulta basada en otras entradas de usuario, como unidades vFiler de una cabina, volúmenes de un agregado o LUN de una máquina virtual de almacenamiento (SVM).	En el flujo de trabajo de ejemplo `Create a Clustered Data ONTAP Volume`, La consulta para VserverName es la siguiente:
	
	`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 consulta hace referencia a ${ClusterName}, donde $ClusterName es el nombre de la entrada de usuario que precede a la entrada de usuario $VserverName.

Tipo: Use el tipo booleano con valores como "'true, false'" para las entradas de usuario que son de naturaleza booleana. Esto ayuda a escribir expresiones internas en el diseño del flujo de trabajo utilizando la entrada del usuario directamente. Por ejemplo, $UserInputName en lugar de $UserInputName == "Yes'.	`$CreateCIFSShare:` Tipo booleano con valores válidos como «'true'» o «'false'»
Tipo: Para tipo de cadena y número, utilice expresiones regulares en la columna valores cuando desee validar el valor con formatos específicos. Utilice expresiones regulares para las entradas de dirección IP y máscara de red.	La entrada de usuario específica de la ubicación puede expresarse como "'[A-Z][A-Z]\-0[1-9]'". Esta información del usuario acepta valores como «'US-01», «'NB-02», pero no «'nb-00».

Directrices	Ejemplo
Tipo: Para el tipo de número, se puede especificar una validación basada en rango en la columna valores.	Para el número de LUN que se van a crear, la entrada en la columna valores es 1-20.
Grupo: Agrupe las entradas de usuario relacionadas con el grupo en los bloques apropiados y nombre el grupo.	«Información de almacenamiento» para todas las entradas del usuario relacionadas con el almacenamiento. «Detalles de Datastore» para todas las entradas del usuario relacionadas con VMware.
Obligatorio: Si el valor de cualquier entrada de usuario es necesario para la ejecución del flujo de trabajo, marque la entrada del usuario como obligatoria. Esto garantiza que la pantalla de entrada del usuario acepte mandatorily esa entrada del usuario.	«»$VolumeName» en el flujo de trabajo «'Create NFS Volume».
Valor predeterminado: Si una entrada de usuario tiene un valor predeterminado que puede funcionar para la mayoría de las ejecuciones del flujo de trabajo, proporcione los valores. Esto ayuda a permitir al usuario proporcionar menos entradas durante la ejecución, si el valor predeterminado cumple con el propósito.	Ninguno

Constantes, variables y parámetros de retorno

Directrices	Ejemplo
Constantes: Defina constantes cuando se utiliza un valor común para definir parámetros en varios comandos.	*AGGREGATE_OVERPROMISO_THRESHOLD* en `Create, map, and protect LUNs with SnapVault` ejemplo de flujo de trabajo.
Constantes: Nombres • Utilice una letra mayúscula al principio de cada palabra. • Utilice letras mayúsculas para todos los términos y abreviaturas. • No utilice guiones bajos. • Utilice letras mayúsculas para todas las letras de nombres constantes.	*AGGREGATE_USED_SPACE_THRESHOLD* *ActualVolumeSizeInMB*

Directrices	Ejemplo
Variables: Proporcione un nombre a un objeto definido en uno de los cuadros de parámetros de comando. Las variables se generan automáticamente nombres y se pueden cambiar.	Ninguno
Variables: Los nombres utilizan caracteres en minúscula para los nombres de variables.	volume1 recurso_compartido_cifs
Parámetros de retorno: Utilice parámetros de retorno cuando la planificación y ejecución del flujo de trabajo devuelva algunos valores calculados o seleccionados durante la planificación. Los valores se ponen a disposición en el modo de vista previa cuando el flujo de trabajo se ejecuta también desde un servicio web.	Agregado: Si se selecciona el agregado mediante la lógica de selección de recursos, el agregado seleccionado real se puede definir como un parámetro return.

Directrices para crear scripts de validación para tipos de sistema remoto

Debe tener en cuenta las directrices para crear scripts de validación que se utilicen para probar los tipos de sistema remoto que defina en OnCommand Workflow Automation (WFA).

- El script Perl que cree debe ser similar al script de ejemplo que se proporciona en la ventana Script de validación.
- El resultado del script de validación debe ser similar al del script de muestra.

Ejemplo de script de validación

```
# 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;
}
```

Directrices para crear tipos de origen de datos

Debe tener en cuenta las directrices para crear los tipos de origen de datos que se utilizan para definir orígenes de datos personalizados para OnCommand Workflow Automation (WFA).

Puede definir un tipo de origen de datos mediante uno de los siguientes métodos:

- SQL: Puede utilizar las directrices de WFA SQL para definir consultas seleccionadas de orígenes de datos basadas en una base de datos externa.
- SCRIPT: Puede escribir una secuencia de comandos de PowerShell que proporcione los datos de un esquema específico de entradas del diccionario.

Las directrices para crear tipos de origen de datos son las siguientes:

- Se debe utilizar el idioma de PowerShell para crear un script.
- El script de PowerShell debe proporcionar la salida de cada entrada de diccionario en su directorio de trabajo actual.
- Se debe dar nombre a los archivos de datos `dictionary_entry.csv`, donde el nombre de la entrada del diccionario debe tener caracteres en minúsculas.

  El tipo de origen de datos predefinido que recopila información de Performance Advisor utiliza un tipo de origen de datos basado en SCRIPTS. Se denomina a los archivos de salida `array_performance.csv` y.. `aggregate_performance.csv`.

- El archivo .csv debe incluir el contenido en el orden exacto de los atributos de entrada del diccionario.

  Una entrada de diccionario incluye atributos en el siguiente orden: Array_ip, date, day, hour, cpu_busy, total_ops_por_seg, disk_throughput_per_s.

  El script de PowerShell añade datos al `.csv` archivar en el mismo orden.

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

- Debe utilizar codificación para asegurarse de que la salida de datos del script se carga en la caché de WFA de forma precisa.
- Debe utilizar \N al introducir un valor Null en `.csv` archivo.