為應用程式開發外掛程式

SnapCenter Software 4.6

NetApp
January 18, 2024

目錄

為應用程式開發外掛程式

總覽

利用此伺服器SnapCenter 、您可以將應用程式部署及管理為SnapCenter 支援的外掛程式。您可將所選的應用程式插入SnapCenter 到《資料保護與管理功能》的《伺服器》中。

使用支援各種程式設計語言、開發自訂外掛程式。SnapCenter您可以使用Perl、Java、批次或其他指令碼語言來開發自訂外掛程式。

若要在SnapCenter Suse中使用自訂外掛程式、您必須執行下列工作：

- 依照本指南的指示、為應用程式建立外掛程式
- 建立說明檔案
- 匯出自訂外掛程式、將其安裝在SnapCenter 支援的主機上
- 將外掛程式的壓縮檔上傳至SnapCenter 32個伺服器

所有**API**呼叫的一般外掛處理

對於每個API呼叫、請使用下列資訊：

- 外掛程式參數
- 結束代碼
- 記錄錯誤訊息
- 資料一致性

使用外掛程式參數

每次API呼叫時、都會將一組參數傳送至外掛程式。下表列出參數的特定資訊。

參數	目的
行動	決定工作流程名稱。例如、探索、備份、檔案VolRestore或cloneVolAndLun
資源	列出要保護的資源。資源會以UID和類型來識別。清單會以下列格式呈現給外掛程式： 「<UID>、<type >、<UID>、<type >」。例如「Instance1、Instance;instance2\DB1、Database」
app_name	決定要使用的外掛程式。例如DB2、MySQL。支援所列應用程式的內建支援。SnapCenter此參數區分大小寫。

參數	目的
app_ignore錯誤	（Y或N）SnapCenter 這會導致出現應用程式錯誤時、不退出或不退出。當您正在備份多個資料庫、但不希望單一故障停止備份作業時、此功能非常實用。
<resource_name>_APP_instance_username	設定資源的認證資料。SnapCenter
<resource_name>_APP_instance_password	設定資源的認證資料。SnapCenter
<resource_name>_<custom_Param>	每個資源層級的自訂金鑰值都可用於以「<resource_name>_」開頭的外掛程式。例如，如果名稱為「MySQLDB」之資源的自訂金鑰為「master_slave」，則該金鑰會以MySQLDB_master_slave的形式提供

使用結束代碼

外掛程式會透過結束代碼、將作業狀態傳回主機。每個程式碼都有特定意義、而且外掛程式會使用正確的結束程式碼來表示相同的意義。

下表說明錯誤代碼及其意義。

結束代碼	目的
0	成功營運。
99	不支援或實作要求的作業。
100	失敗的作業、跳過取消靜止、然後結束。依預設、取消靜止。
101.	作業失敗、請繼續備份作業。
其他	失敗的作業、執行Unquiesce、然後結束。

記錄錯誤訊息

錯誤訊息會從外掛程式傳遞至SnapCenter 該伺服器。訊息包括訊息、記錄層級和時間戳記。

下表列出層級及其用途。

參數	目的
資訊	資訊訊息
警告	警告訊息

參數	目的
錯誤	錯誤訊息
偵錯	偵錯訊息
追蹤	追蹤訊息

維持資料一致性

自訂外掛程式會在執行相同工作流程的作業之間保留資料。例如、外掛程式可以在靜止結束時儲存資料、以便在靜止作業期間使用。

要保留的資料會透過外掛程式設定為結果物件的一部分。它採用特定格式、並在每種外掛程式開發方式下詳細說明。

基於Perl的開發

使用Perl開發外掛程式時、您必須遵循特定慣例。

- 內容必須可讀取
- 必須實作必要的作業設定、靜止和取消靜止
- 必須使用特定語法將結果傳回代理程式
- 內容應儲存為<plugin_name>.PM檔案

可用的作業包括

- 設定
- 版本
- 靜止
- 取消靜止
- Clone（克隆）_pre, clone（複製）_POST
- reet_pre,還原
- 清理

一般外掛處理

使用結果物件

每個自訂外掛程式作業都必須定義結果物件。此物件會將訊息、結束程式碼、stdout和stderr傳回主機代理程式。

結果物件：

```
my $result = {
```

```
      exit_code => 0,
      stdout => "",
      stderr => "",
};
```

傳回結果物件：

```
return $result;
```

維持資料一致性

在執行相同工作流程時、可以保留作業之間的資料（清除除外）。這是使用金鑰值配對來完成。關鍵值資料配對會設定為結果物件的一部分、並保留在相同工作流程的後續作業中。

下列程式碼範例會設定要保留的資料：

```
my $result = {
  exit_code => 0,
  stdout => "",
  stderr => "",
};
  $result->{env}->{'key1'} = 'value1';
  $result->{env}->{'key2'} = 'value2';
  ….
  return $result
```

上述程式碼會設定兩個金鑰值配對、作為後續作業的輸入。可使用下列程式碼存取兩個金鑰值配對：

```
sub setENV {
    my ($self, $config) = @_;
    my $first_value = $config->{'key1'};
    my $second_value = $config->{'key2'};
    …
}
```

```
 === Logging error messages
```
每項作業都能將訊息傳回主機代理程式、以顯示及儲存內容。訊息包含訊息層級、時間戳記及訊息文字。支援多行訊息。

```
Load the SnapCreator::Event Class:
my $msgObj = new SnapCreator::Event();
my @message_a = ();
```

使用msgobj使用collect方法擷取訊息。

```
$msgObj->collect(\@message_a, INFO, "My INFO Message");
$msgObj->collect(\@message_a, WARN, "My WARN Message");
$msgObj->collect(\@message_a, ERROR, "My ERROR Message");
$msgObj->collect(\@message_a, DEBUG, "My DEBUG Message");
$msgObj->collect(\@message_a, TRACE, "My TRACE Message");
```

將訊息套用至結果物件：

```
$result->{message} = \@message_a;
```

使用外掛程式存根

自訂外掛程式必須公開外掛程式存根。這些是SnapCenter 根據工作流程、由伺服器呼叫的方法。

外掛程式**Stub**	選用/必要	目的
設定	必要	此存根會設定環境和組態物件。 任何環境剖析或處理都應在此處完成。每次呼叫一個存根時、就會先呼叫setenv存根。只有Perl型外掛程式才需要。
版本	選用	此存根可用來取得應用程式版本。

外掛程式**Stub**	選用/必要	目的
探索	選用	此存根可用來探索代理程式或主機上裝載的執行個體或資料庫等應用程式物件。 外掛程式預期會以特定格式傳回探索到的應用程式物件、做為回應的一部分。只有在應用程式與SnapDrive 適用於Unix的解決方法整合時、才會使用此存根。 支援Linux檔案系統（Linux Flavors）。不支援AIX/Solaris（Unix Flavors）。
探索完成	選用	此存根可用來探索代理程式或主機上裝載的執行個體或資料庫等應用程式物件。 外掛程式預期會以特定格式傳回探索到的應用程式物件、做為回應的一部分。只有在應用程式與SnapDrive 適用於Unix的解決方法整合時、才會使用此存根。 支援Linux檔案系統（Linux Flavors）。不支援AIX和Solaris（Unix Flavors）。
靜止	必要	此存根負責執行靜止、也就是將應用程式置於可建立Snapshot複本的狀態。這在Snapshot複製作業之前稱為。保留的應用程式中繼資料應設定為回應的一部分、在後續複製或還原作業期間、應以組態參數的形式、傳回對應儲存Snapshot複本上的中繼資料。
取消靜止	必要	此存根負責執行靜止、也就是將應用程式置於正常狀態。建立Snapshot複本之後就會呼叫此功能。
Clone預先複製	選用	此存根負責執行預先複製工作。這假設您使用內建SnapCenter 的「還原伺服器複製」介面、並在執行複製作業時觸發。

外掛程式**Stub**	選用/必要	目的
Clone複製POST	選用	此存根負責執行複製後的工作。這假設您使用內建SnapCenter 的「還原伺服器複製」介面、而且只有在執行複製作業時才會觸發。
reet_pre	選用	此存根負責執行預先儲存的工作。這假設您使用內建SnapCenter 的還原伺服器介面、並在執行還原作業時觸發。
還原	選用	此存根負責執行應用程式還原工作。這是假設您使用內建SnapCenter 的「還原伺服器」介面、而且只有在執行還原作業時才會觸發。
清理	選用	此存根負責在備份、還原或複製作業之後執行清除作業。清除作業可以是在正常工作流程執行期間、或是在工作流程失敗時進行。您可以參照組態參數動作來推斷呼叫清除的工作流程名稱、此動作可以是備份、cloneVolAndLun或fileVolRestore。組態參數error_message會指出執行工作流程時是否有任何錯誤。如果已定義「錯誤」訊息而非「空」、則會在工作流程失敗執行期間呼叫清除。
app_version	選用	這個虛設常式是SnapCenter 由效能分析用來取得外掛程式所管理的應用程式版本詳細資料。

外掛程式套件資訊

每個外掛程式都必須具備下列資訊：

```
package MOCK;
our @ISA = qw(SnapCreator::Mod);
=head1 NAME
MOCK - class which represents a MOCK module.
=cut
=head1 DESCRIPTION
MOCK implements methods which only log requests.
=cut
use strict;
use warnings;
use diagnostics;
use SnapCreator::Util::Generic qw ( trim isEmpty );
use SnapCreator::Util::OS qw ( isWindows isUnix getUid
createTmpFile );
use SnapCreator::Event qw ( INFO ERROR WARN DEBUG COMMENT ASUP
CMD DUMP );
my $msgObj = new SnapCreator::Event();
my %config_h = ();
```

營運

您可以對自訂外掛程式所支援的各種作業進行程式碼處理、例如：setenv、Version、Quiesce和unquiesce。

setenv作業

使用Perl建立的外掛程式需要設定作業。您可以設定ENV並輕鬆存取外掛程式參數。

```
sub setENV {
    my ($self, $obj) = @_;
    %config_h = %{$obj};
    my $result = {
      exit_code => 0,
      stdout => "",
      stderr => "",
    };
    return $result;
}
```

版本作業

版本作業會傳回應用程式版本資訊。

```
sub version {
  my $version_result = {
    major => 1,
    minor => 2,
    patch => 1,
    build => 0
  };
  my @message_a = ();
  $msgObj->collect(\@message_a, INFO, "VOLUMES
$config_h{'VOLUMES'}");
  $msgObj->collect(\@message_a, INFO,
"$config_h{'APP_NAME'}::quiesce");
  $version_result->{message} = \@message_a;
  return $version_result;
}
```

靜止作業

靜止作業會對資源參數中所列的資源執行應用程式靜止作業。

```
sub quiesce {
  my $result = {
      exit_code => 0,
      stdout => "",
      stderr => "",
  };
  my @message_a = ();
  $msgObj->collect(\@message_a, INFO, "VOLUMES
$config_h{'VOLUMES'}");
  $msgObj->collect(\@message_a, INFO,
"$config_h{'APP_NAME'}::quiesce");
  $result->{message} = \@message_a;
  return $result;
}
```

取消靜止作業

若要取消靜止應用程式、必須執行「取消靜止」作業。資源清單可在資源參數中找到。

```
sub unquiesce {
  my $result = {
      exit_code => 0,
      stdout => "",
      stderr => "",
  };
  my @message_a = ();
  $msgObj->collect(\@message_a, INFO, "VOLUMES
$config_h{'VOLUMES'}");
  $msgObj->collect(\@message_a, INFO,
"$config_h{'APP_NAME'}::unquiesce");
  $result->{message} = \@message_a;
  return $result;
}
```

原生風格

支援非Perl程式設計或指令碼語言來建立外掛程式。SnapCenter這稱為原生樣式程式設計、可以是指令碼或批次檔。

原生型外掛程式必須遵循下列特定慣例：

外掛程式必須可執行

- 對於Unix系統、執行代理程式的使用者必須擁有外掛程式的執行權限
- 對於Windows系統、PowerShell外掛程式必須有.ps1字尾、其他Windows指令碼必須有.cmd或.bat字尾、而且必須由使用者執行
- 外掛程式必須回應命令列引數、例如「-quiesce」、「-unquiesce」
- 外掛程式必須傳回結束代碼99、以防未實作任何作業或功能
- 外掛程式必須使用特定語法、才能將結果傳回伺服器

一般外掛處理

記錄錯誤訊息

每項作業都能將訊息傳回伺服器、以顯示及儲存內容。訊息包含訊息層級、時間戳記及訊息文字。支援多行訊息。

格式：

```
SC_MSG#<level>#<timestamp>#<message>
SC_MESSAGE#<level>#<timestamp>#<message>
```

使用外掛程式存根

選購外掛程式必須實作外掛程式存根。SnapCenter以上是SnapCenter 根據特定工作流程、由伺服器呼叫的方法。

外掛程式**Stub**	選用/必要	目的
靜止	必要	此存根負責執行靜止。它會將應用程式置於可建立Snapshot複本的狀態。這在儲存Snapshot複製作業之前稱為。
取消靜止	必要	此存根負責執行取消靜止。它會將應用程式置於正常狀態。這是在儲存Snapshot複製作業之後呼叫的。
Clone預先複製	選用	此存根負責執行預先複製工作。這是假設您使用內建SnapCenter 的還原複製介面、而且只會在執行「clone_vol或clone_LUN」動作時觸發。
Clone複製POST	選用	此存根負責執行複製後的工作。這假設您使用內建SnapCenter 的還原複製介面、而且只會在執行「clone_vol或clone_LUN」作業時觸發。
reet_pre	選用	此存根負責執行預先還原工作。這假設您使用內建SnapCenter 的還原介面、而且只有在執行還原作業時才會觸發。
還原	選用	此存根負責執行所有還原動作。假設您未使用內建還原介面。在執行還原作業時觸發。

範例

Windows PowerShell

檢查指令碼是否可在您的系統上執行。如果無法執行指令碼、請為指令碼設定Set-ExecutionPolicy略過、然後重試該作業。

```
if ($args.length -ne 1) {
  write-warning "You must specify a method";
  break;
}
function log ($level, $message) {
  $d = get-date
  echo "SC_MSG#$level#$d#$message"
}
function quiesce {
  $app_name = (get-item env:APP_NAME).value
  log "INFO" "Quiescing application using script $app_name";
  log "INFO" "Quiescing application finished successfully"
}
function unquiesce {
  $app_name = (get-item env:APP_NAME).value
  log "INFO" "Unquiescing application using script $app_name";
  log "INFO" "Unquiescing application finished successfully"
}
  switch ($args[0]) {
    "-quiesce" {
    quiesce;
}
"-unquiesce" {
    unquiesce;
}
default {
    write-error "Function $args[0] is not implemented";
    exit 99;
  }
}
exit 0;
```

Java樣式

Java自訂外掛程式會直接與資料庫、執行個體等應用程式互動。

限制

使用Java程式設計語言開發外掛程式時、您應該注意到某些限制。

外掛程式特性	**Java**外掛程式
複雜度	從低到中

外掛程式特性	**Java**外掛程式
記憶體佔用空間	最高10至20 MB
相依於其他程式庫	用於應用程式通訊的程式庫
執行緒數量	1.
執行緒執行時間	不到一小時

Java限制的理由

此解決方案的目標SnapCenter 是確保應用程式整合持續、安全且健全。藉由支援Java外掛程式、外掛程式可能會導致記憶體洩漏及其他不必要的問題。這些問題很難解決、尤其是為了讓事情變得簡單易用。如果外掛程式的複雜度不太複雜、則開發人員較不可能導入錯誤。Java外掛程式的危險在於它們與SnapCenter Sof the Sof the Sof the Same JVM.外掛程式當機或洩漏記憶體時、也可能對代理程式造成負面影響。

支援的方法

方法	必要	說明	何時及由誰呼叫？
版本	是的	需要傳回外掛程式的版本。	由伺服器或代理程式要求外掛程式的版本。SnapCenter
靜止	是的	需要在應用程式上執行靜止。在大多數情況下、這表示將應用程式置於SnapCenter 某個狀態、讓該應用程式能夠建立備份（例如Snapshot複本）。	在該伺服器SnapCenter 建立Snapshot複本或執行一般備份之前。
取消靜止	是的	需要在應用程式上執行靜止。在大多數情況下、這表示應用程式會恢復正常運作狀態。	在支援伺服器SnapCenter 建立Snapshot複本或執行一般備份之後。
清理	否	負責清除外掛程式需要清理的任何項目。	當在伺服器SnapCenter 上完成工作流程（成功或失敗）時。
clonewePre	否	應在執行複製作業之前執行必要的動作。	當使用者觸發「clonewal Vol」或「clonewlun」動作、並使用內建的複製精靈（GUI / CLI）。

方法	必要	說明	何時及由誰呼叫？
clonePost	否	應在執行複製作業之後執行必要的動作。	當使用者觸發「clonewalVol」或「clonewlun」動作、並使用內建的複製精靈（GUI / CLI）。
重述預先發	否	應在叫用還原作業之前執行必要的動作。	當使用者觸發還原作業時。
還原	否	負責執行應用程式的還原/還原。	當使用者觸發還原作業時。
應用程式版本	否	擷取由外掛程式管理的應用程式版本。	在每個工作流程（例如備份/還原/複製）中、ASUP資料收集的一部分。

教學課程

本節說明如何使用Java程式設計語言建立自訂外掛程式。

設定月食

1. 在Eclipse中建立新的Java專案「圖則外掛程式」
2. 單擊*完成*
3. 右鍵單擊*新項目*→*屬性*→* Java建置路徑*→*資源庫*→*新增外部JAR*
4. 瀏覽至主機代理程式的./lib/資料夾、然後選取Jar scAgent-5.0-core.jar和common-5.0.jar
5. 選取專案、然後在* src資料夾*→*新增*→套件*上按一下滑鼠右鍵、然後建立名稱為com.netapp.snapcreator.agent.plugin.TutorialPlugin的新套件
6. 在新套件上按一下滑鼠右鍵、然後選取「New（新增）」→「Java Class（Java類別）」。
   a. 輸入圖則外掛程式名稱。
   b. 按一下超類別瀏覽按鈕、然後搜尋「*抽象外掛程式」。只能顯示一個結果：

```
 "AbstractPlugin - com.netapp.snapcreator.agent.nextgen.plugin".
.. 單擊*完成*。
.. Java類別：
```

```
package com.netapp.snapcreator.agent.plugin.TutorialPlugin;
import
com.netapp.snapcreator.agent.nextgen.common.result.Describe
Result;
import
com.netapp.snapcreator.agent.nextgen.common.result.Result;
import
com.netapp.snapcreator.agent.nextgen.common.result.VersionR
esult;
import
com.netapp.snapcreator.agent.nextgen.context.Context;
import
com.netapp.snapcreator.agent.nextgen.plugin.AbstractPlugin;
public class TutorialPlugin extends AbstractPlugin {
  @Override
  public DescribeResult describe(Context context) {
    // TODO Auto-generated method stub
    return null;
  }
  @Override
  public Result quiesce(Context context) {
    // TODO Auto-generated method stub
    return null;
  }
  @Override
  public Result unquiesce(Context context) {
    // TODO Auto-generated method stub
    return null;
  }
  @Override
  public VersionResult version() {
    // TODO Auto-generated method stub
    return null;
  }
}
```

實作所需方法

靜止、取消靜止和版本是每個自訂Java外掛程式必須實作的必要方法。

以下是傳回外掛程式版本的版本方法。

```
@Override
public VersionResult version() {
    VersionResult versionResult = VersionResult.builder()
                                            .withMajor(1)
                                            .withMinor(0)
                                            .withPatch(0)
                                            .withBuild(0)
                                            .build();
    return versionResult;
}
```

```
Below is the implementation of quiesce and unquiesce method. These will be
interacting with   the application, which is being protected by SnapCenter
Server. As this is just a tutorial, the
application part is not explained, and the focus is more on the
functionality that SnapCenter   Agent provides the following to the plug-
in developers:
```

```
@Override
  public Result quiesce(Context context) {
    final Logger logger = context.getLogger();
    /*
      * TODO: Add application interaction here
    */
```

```
logger.error("Something bad happened.");
logger.info("Successfully handled application");
```

```
    Result result = Result.builder()
                    .withExitCode(0)
                    .withMessages(logger.getMessages())
                    .build();
    return result;
}
```

方法會在內容物件中傳遞。其中包含多個協助工具、例如Logger和內容存放區、以及目前作業的相關資訊（工作流程ID、工作ID）。我們可以呼叫最終Logger logger = context.getLogger（）；來取得記錄程式。記錄程式物件提供類似於其他記錄架構的方法、例如登入。在結果物件中、您也可以指定結束程式碼。在此範例中、零會傳回、因為沒有問題。其他結束代碼可對應至不同的故障情況。

使用結果物件

結果物件包含下列參數：

參數	預設	說明
組態	空組態	此參數可用於將組態參數傳回伺服器。它可以是外掛程式想要更新的參數。此變更是否實際反映在SnapCenter 支援伺服器上的組態中、取決於組態中的APP_CON_PONY面=Y或N參數。
exitCode	0	表示作業狀態。「0」表示作業已成功執行。其他值表示錯誤或警告。
stdout	空白清單	這可用來將stdout訊息傳回SnapCenter 至該伺服器。
stderr	空白清單	這可用來將stderr訊息傳回SnapCenter 至該伺服器。
訊息	空白清單	此清單包含外掛程式要傳回伺服器的所有訊息。該伺服器會在CLI或GUI中顯示這些訊息。SnapCenter

此功能可為建置者提供支援SnapCenter ("建構者模式"）的所有結果類型。這讓使用者變得非常簡單：

```
Result result = Result.builder()
                     .withExitCode(0)
                     .withStdout(stdout)
                     .withStderr(stderr)
                     .withConfig(config)
                     .withMessages(logger.getMessages())
                     .build()
```

例如、將結束程式碼設為0、設定stdout和Stderr的清單、設定組態參數、以及附加將傳送回伺服器的記錄訊息。如果您不需要所有參數、請只傳送所需的參數。由於每個參數都有一個預設值、因此如果您從下列程式碼中移除.withExitCode（0）、則不會影響結果：

```
Result result = Result.builder()
                       .withExitCode(0)
                       .withMessages(logger.getMessages())
                       .build();
```

版本結果

版本結果會將SnapCenter 外掛程式版本通知到該伺服器。由於它也從結果繼承、因此包含config、exitCode、stdout、stderr和Messages參數。

參數	預設	說明
主要	0	外掛程式的主要版本欄位。
次要	0	外掛程式的次要版本欄位。
修補程式	0	外掛程式的「修補版本」欄位。
建置	0	外掛程式的建置版本欄位。

例如：

```
VersionResult result = VersionResult.builder()
                                .withMajor(1)
                                .withMinor(0)
                                .withPatch(0)
                                .withBuild(0)
                                .build();
```

使用內容物件

內容物件提供下列方法：

內容方法	目的
字串getWorkflowId()；	傳回SnapCenter 目前工作流程使用的流程ID。
config getconfig（）；	傳回SnapCenter 正在從功能表伺服器傳送至代理程式的組態。

工作流程**ID**

Workflow ID是SnapCenter 指由伺服器用來參照特定執行中工作流程的ID。

組態

此物件包含（大部分）使用者可在SnapCenter 物件伺服器的組態中設定的參數。不過、由於安全性原因、部分參數可能會在伺服器端篩選。以下是如何存取Config和擷取參數的範例：

```
final Config config = context.getConfig();
String myParameter =
config.getParameter("PLUGIN_MANDATORY_PARAMETER");
```

現在、「//myParexer」包含SnapCenter 從效能分析伺服器上的組態讀取的參數。如果組態參數鍵不存在、則會傳回空白字串（""）。

匯出外掛程式

您必須匯出外掛程式、才能將其安裝在SnapCenter 該支援主機上。

在Eclipse中執行下列工作：

1. 在外掛程式的基礎套件上按一下滑鼠右鍵（請參閱範例com.netapp.snapcreator.agent.plugin.TutorialPlugin）。
2. 選擇*匯出*→* Java*→* Jar檔案*
3. 單擊 * 下一步 * 。
4. 在下列視窗中、指定目的地Jar檔案路徑：tutorial_plugin.jar外掛程式的基礎類別名稱為"TutorialPlugin.class"、外掛程式必須新增至名稱相同的資料夾。

如果外掛程式取決於其他程式庫、您可以建立下列資料夾：lib/

您可以新增與外掛程式相依的Jar檔案（例如資料庫驅動程式）。當程式庫載入外掛程式時、它會自動將此資料夾中的所有Jar檔案與其相關聯、並將其新增至類路徑。SnapCenter

自訂SnapCenter 插件

自訂SnapCenter 插件

使用Java、Perl或原生樣式所建立的自訂外掛程式、可以使用SnapCenter 支援應用程式資料保護的功能、安裝在主機上。您必須先匯出外掛程式、SnapCenter 才能使用本教學課程所提供的程序將其安裝在支援主機上。

建立外掛程式說明檔案

對於每個建立的外掛程式、您都必須擁有說明檔案。說明檔案會說明外掛程式的詳細資料。檔案名稱必須是Plugin_filer.xml。

使用外掛程式描述元檔案屬性及其重要性

屬性	說明
名稱	外掛程式的名稱。允許使用英數字元。例如DB2、MySQL、MongoDB 對於以原生樣式建立的外掛程式、請確定您未提供檔案副檔名。例如、如果外掛程式名稱為MongoDB.sh、請將名稱指定為MongoDB。

屬性	說明
版本	外掛程式版本。可同時包含主要和次要版本。例如1.0、1.1、2.0、2.1
顯示名稱	顯示在SnapCenter Is供 伺服器中的外掛程式名稱。如果寫入相同外掛程式的多個版本、請確認所有版本的顯示名稱都相同。
PluginType	用於建立外掛程式的語言。支援的值包括Perl、Java和Native。原生外掛程式類型包括Unix/Linux Shell指令碼、Windows指令碼、Python或任何其他指令碼語言。
OSName	安裝外掛程式的主機OS名稱。有效值為Windows和Linux。單一外掛程式可以部署在多種作業系統類型上、例如Perl類型的外掛程式。
作業系統版本	安裝外掛程式的主機作業系統版本。
資源名稱	外掛程式可支援的資源類型名稱。例如資料庫、執行個體、集合。
父	在這種情況下、資源名稱會階層式相依於其他資源類型、然後由父資源類型決定父資源類型。 例如，DB2外掛程式的資源名稱「資料庫」具有父「執行個體」。
RequireFileSystemPlugin	是或否決定還原精靈中是否顯示「恢復」索引標籤。
資源要求驗證	是或否決定自動探索或未自動探索的資源、在探索儲存設備後、是否需要認證資料來執行資料保護作業。
RequireFileSystemClone	是或否決定外掛程式是否需要檔案系統外掛程式整合、才能完成複製工作流程。

以下是自訂外掛程式DB2的Plugin_descriptor.xml檔案範例：

```
<Plugin>
<SMSServer></SMSServer>
<Name>DB2</Name>
<Version>1.0</Version>
<PluginType>Perl</PluginType>
<DisplayName>Custom DB2 Plugin</DisplayName>
<SupportedOS>
<OS>
<OSName>windows</OSName>
<OSVersion>2012</OSVersion>
</OS>
<OS>
<OSName>Linux</OSName>
<OSVersion>7</OSVersion>
</OS>
</SupportedOS>
<ResourceTypes>
<ResourceType>
<ResourceName>Database</ResourceName>
<Parent>Instance</Parent>
</ResourceType>
<ResourceType>
<ResourceName>Instance</ResourceName>
</ResourceType>
</ResourceTypes>
<RequireFileSystemPlugin>no</RequireFileSystemPlugin>
<ResourceRequiresAuthentication>yes</ResourceRequiresAuthentication>
<SupportsApplicationRecovery>yes</SupportsApplicationRecovery>
</Plugin>
```

建立壓縮檔

在開發外掛程式並建立描述元檔案之後、您必須將外掛程式檔案和Plugin_descriptor.xml檔案新增至資料夾、然後將其壓縮。

在建立ZIP檔案之前、您必須考量下列事項：

- 指令碼名稱必須與外掛程式名稱相同。
- 對於Perl外掛程式、ZIP資料夾必須包含含有指令碼檔案的資料夾、且描述元檔案必須位於此資料夾之外。資料夾名稱必須與外掛程式名稱相同。
- 對於Perl外掛程式以外的外掛程式、ZIP資料夾必須包含描述元和指令碼檔案。
- 作業系統版本必須為數字。

範例：

- DB2外掛程式：將DB2.PM和Plugin_descriptor.xml檔案新增至「DB2.zip」。
- 使用Java開發的外掛程式：將Jar檔案、相依的Jar檔案和Plugin_descriptor.xml檔案新增至資料夾、然後將其壓縮。

正在上傳外掛程式**ZIP**檔案

您必須將外掛程式的ZIP檔案上傳SnapCenter 至Sfor the plug-in Server、以便在所需的主機上部署外掛程式。

您可以使用UI或Cmdlet上傳外掛程式。

使用者介面：

- 將外掛程式ZIP檔案上傳為「新增」或「修改主機」工作流程精靈的一部分
- 按一下*「Select to upload custom plug-in（選擇上傳自訂外掛程式）」*
- PowerShell：*
- uple-SmPluginPackage Cmdlet

  例如、PS-Ups>Upse-SmPluginPackage -AbsolutePath c：\DB2_1.zip

  如需PowerShell Cmdlet的詳細資訊、請使用SnapCenter 支援程式指令程式說明或參閱Cmdlet參考資訊。

"《軟件指令程式參考指南》SnapCenter"。

部署自訂外掛程式

上傳的自訂外掛程式現在可在*新增*和*修改主機*工作流程的所需主機上進行部署。您可以將多個版本的外掛程式上傳至SnapCenter 支援伺服器、也可以選擇要部署在特定主機上的版本。

如需如何上傳外掛程式的詳細資訊、請參閱： "新增主機並在遠端主機上安裝外掛程式套件"