Custom plug-in in SnapCenter

SnapCenter Software

Nirupama Sriram
June 17, 2021

Table of Contents

Custom plug-in in SnapCenter 1
Custom plug-in in SnapCenter 1

Custom plug-in in SnapCenter

Custom plug-in in SnapCenter

The custom plug-in created using Java, PERL, or NATIVE style can be installed on the host using SnapCenter Server to enable data protection of your application. You must have exported the plug-in to install it on the SnapCenter host using the procedure provided in this tutorial.

Creating a plug-in description file

For every plug-in created, you must have a description file. The description file describes the details of the plug-in. The name of the file must be Plugin_descriptor.xml.

Using plug-in descriptor file attributes and its significance

Attribute	Description
Name	Name of the plug-in. Alpha numeric characters are allowed. For example, DB2, MYSQL, MongoDB For plug-ins created in NATIVE style, ensure that you do not provide the extension of the file. For example, if the plug-in name is MongoDB.sh, specify the name as MongoDB.
Version	Plug-in version. Can include both major and minor version. For example, 1.0, 1.1, 2.0, 2.1
DisplayName	The plug-in name to be displayed in SnapCenter Server. If multiple versions of the same plug-in are written, ensure that the display name is the same across all versions.
PluginType	Language used to create the plug-in. Supported values are Perl, Java and Native. Native plug-in type includes Unix/Linux shell scripts, Windows scripts, Python or any other scripting language.
OSName	The host OS name where the plug-in is installed. Valid values are Windows and Linux. It is possible for a single plug-in to be available for deployment on multiple OS types, like PERL type plug-in.
OSVersion	The host OS version where plug-in is installed.
ResourceName	Name of resource type that the plug-in can support. For example, database, instance, collections.

Attribute	Description
Parent	In case, the ResourceName is hierarchically dependent on another Resource type, then Parent determines the parent ResourceType. For instance, DB2 plug-in, the ResourceName “Database” has a parent “Instance”.
RequireFileSystemPlugin	Yes or No. Determines if the recovery tab is displayed in the restore wizard.
ResourceRequiresAuthentication	Yes or No. Determines if the resources, which are auto discovered or have not been auto discovered need credentials to perform the data protection operations after discovering the storage.
RequireFileSystemClone	Yes or No. Determines if the plug-in requires FileSystem plug-in integration for clone workflow.

An example of the Plugin_descriptor.xml file for custom plug-in DB2 is as follows:

```
<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>
```

Creating a ZIP file

After a plug-in is developed and a descriptor file is created, you must add the plug-in files and the Plugin_descriptor.xml file to a folder and zip it.

You must consider the following before creating a ZIP file:

- The script name must be same as the plug-in name.
- For PERL plug-in, the ZIP folder must contain a folder with the script file and the descriptor file must be outside this folder. The folder name must be the same as the plug-in name.
- For plug-ins other than the PERL plug-in, the ZIP folder must contain the descriptor and the script files.
- The OS version must be a number.

Examples:

- DB2 plug-in: add DB2.pm and Plugin_descriptor.xml file to “DB2.zip”.

- Plug-in developed using Java: add jar files, dependent jar files, and Plugin_descriptor.xml file to a folder and zip it.

Uploading the plug-in ZIP file

You must upload the plug-in ZIP file to SnapCenter Server so that the plug-in is available for deployment on the desired host.

You can upload the plug-in using the UI or cmdlets.

UI:

- Upload the plug-in ZIP file as part of **Add** or **Modify Host** workflow wizard
- Click **"Select to upload custom plug-in"**

PowerShell:

- Upload-SmPluginPackage cmdlet

  For example, PS> Upload-SmPluginPackage -AbsolutePath c:\DB2_1.zip

  For detailed information about PowerShell cmdlets, use the SnapCenter cmdlet help or see the cmdlet reference information.

SnapCenter Software Cmdlet Reference Guide.

Deploying the custom plug-ins

The uploaded custom plug-in is now available for deployment on the desired host as part of the **Add** and **Modify Host** workflow. You can have multiple version of plug-ins uploaded to the SnapCenter Server and you can select the desired version to deploy on a specific host.

For more information on how to upload the plug-in see, Add hosts and install plug-in packages on remote hosts