응용 프로그램용 플러그인을 개발합니다

SnapCenter Software 5.0

NetApp
July 18, 2024

목차

응용 프로그램용 플러그인을 개발합니다

개요

SnapCenter 서버를 사용하면 SnapCenter에 대한 플러그인으로 응용 프로그램을 배포 및 관리할 수 있습니다. 원하는 애플리케이션을 SnapCenter 서버에 연결하여 데이터 보호 및 관리 기능을 활용할 수 있습니다.

SnapCenter를 사용하면 다양한 프로그래밍 언어를 사용하여 사용자 지정 플러그인을 개발할 수 있습니다. Perl, Java, 배치 또는 기타 스크립팅 언어를 사용하여 사용자 지정 플러그인을 개발할 수 있습니다.

SnapCenter에서 사용자 지정 플러그인을 사용하려면 다음 작업을 수행해야 합니다.

- 이 가이드의 지침에 따라 응용 프로그램용 플러그인을 만듭니다
- 설명 파일을 만듭니다
- 사용자 지정 플러그인을 내보내어 SnapCenter 호스트에 설치합니다
- 플러그인 zip 파일을 SnapCenter 서버에 업로드합니다

모든 **API** 호출의 일반 플러그인 처리

모든 API 호출에 대해 다음 정보를 사용합니다.

- 플러그인 매개 변수
- 종료 코드
- 오류 메시지를 기록합니다
- 데이터 정합성

플러그인 매개 변수를 사용합니다

매개 변수 집합은 모든 API 호출의 일부로 플러그인으로 전달됩니다. 다음 표에서는 매개 변수에 대한 특정 정보를 보여 줍니다.

매개 변수	목적
조치	워크플로 이름을 결정합니다. 예를 들어, 검색, 백업, fileOrVolRestore 또는 cloneVolAndLun을 사용할 수 있습니다
리소스	보호할 리소스를 나열합니다. 리소스는 UID 및 유형으로 식별됩니다. 이 목록은 다음 형식으로 플러그인에 표시됩니다. "<UID>, <type>;<UID>, <type>". 예: “Instance1, 인스턴스;Instance2\\DB1, 데이터베이스”

매개 변수	목적
APP_NAME입니다	사용 중인 플러그인을 결정합니다. 예: DB2, MySQL. SnapCenter 서버는 나열된 응용 프로그램에 대한 지원을 기본적으로 제공합니다. 이 매개 변수는 대/소문자를 구분합니다.
app_ignore_error	(Y 또는 N) 응용 프로그램 오류가 발생하면 SnapCenter가 종료되거나 종료되지 않습니다. 이 기능은 여러 데이터베이스를 백업할 때 단일 오류로 인해 백업 작업이 중지되지 않도록 하는 경우에 유용합니다.
resource_name>__app_instance_username입니다	리소스에 대해 SnapCenter 자격 증명이 설정되었습니다.
resource_name>_app_instance_password	리소스에 대해 SnapCenter 자격 증명이 설정되었습니다.
resource_name>_<custom_pRAM>	모든 리소스 수준 사용자 지정 키 값은 "<resource_name>_" 접두사가 붙은 플러그인에서 사용할 수 있습니다. 예를 들어 사용자 지정 키가 "MySQLDB"라는 리소스의 "master_slave"인 경우 MySQLDB_master_slave로 사용할 수 있습니다

종료 코드를 사용합니다

플러그인은 종료 코드를 통해 작업 상태를 호스트로 다시 반환합니다. 각 코드는 특정 의미를 가지고 있으며 플러그인은 동일한 것을 나타내기 위해 오른쪽 종료 코드를 사용합니다.

다음 표에서는 오류 코드와 그 의미를 보여 줍니다.

종료 코드입니다	목적
0	작업이 성공했습니다.
99	요청된 작업이 지원되지 않거나 구현되지 않았습니다.
100	작업이 실패했습니다. 일시 중지 해제를 건너뛰고 를 종료합니다. 일시 중지 해제는 기본적으로 사용됩니다.
101	작업이 실패했습니다. 백업 작업을 계속합니다.
기타	작업이 실패했습니다. 중지 해제를 실행하고 종료합니다.

오류 메시지를 기록합니다

오류 메시지는 플러그인에서 SnapCenter 서버로 전달됩니다. 메시지에는 메시지, 로그 수준 및 타임 스탬프가 포함됩니다.

다음 표에는 레벨 및 그 용도가 나와 있습니다.

매개 변수	목적
정보	정보 메시지입니다
경고	경고 메시지
오류	오류 메시지
디버그	디버그 메시지입니다
트레이스	Trace 메시지

데이터 일관성 유지

사용자 지정 플러그인은 동일한 워크플로우 실행 작업 간 데이터를 보존합니다. 예를 들어, 플러그인은 일시 중지 종료 시 데이터를 저장할 수 있으며, 일시 중지 해제 작업 중에 사용할 수 있습니다.

보존할 데이터는 플러그인에 의해 결과 객체의 일부로 설정됩니다. 특정 형식을 따르며 각 플러그인 개발 스타일 아래에 자세히 설명되어 있습니다.

Perl 기반 개발

PERL을 사용하여 플러그인을 개발하는 동안 특정 규칙을 따라야 합니다.

- 내용을 읽을 수 있어야 합니다
- 필수 작업 setenv, quiesce 및 unquiesce를 구현해야 합니다
- 결과를 에이전트로 다시 전달하려면 특정 구문을 사용해야 합니다
- 콘텐츠는 <plugin_name>.pm 파일로 저장해야 합니다

사용 가능한 작업은 입니다

- 설정
- 버전
- 정지
- 정지 해제
- clone_pre, clone_post
- restore_pre, 복구하십시오
- 정리

일반적인 플러그인 처리

결과 개체 사용

모든 사용자 지정 플러그인 작업은 결과 개체를 정의해야 합니다. 이 개체는 메시지, 종료 코드, stdout 및 stderr를 다시 호스트 에이전트로 보냅니다.

결과 개체:

```
my $result = {
```

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

결과 객체 반환:

```
return $result;
```

데이터 일관성 유지

정리 작업을 제외한 작업 간 데이터를 동일한 워크플로 실행의 일부로 보존할 수 있습니다. 이 작업은 키 값 쌍을 사용하여 수행됩니다. 데이터의 키 값 쌍은 결과 개체의 일부로 설정되며 동일한 워크플로의 후속 작업에서 사용 가능한 것으로 유지됩니다.

다음 코드 샘플은 보존할 데이터를 설정합니다.

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

위의 코드는 두 개의 키 값 쌍을 설정하며, 이러한 키 값 쌍은 후속 작업에서 입력으로 사용할 수 있습니다. 다음 코드를 사용하여 두 키 값 쌍에 액세스할 수 있습니다.

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

```
 === Logging error messages
```

각 작업은 콘텐츠를 표시하고 저장하는 호스트 에이전트로 메시지를 다시 보낼 수 있습니다. 메시지에는 메시지 수준, 타임스탬프 및 메시지 텍스트가 포함됩니다. 여러 줄 메시지가 지원됩니다.

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

msgObj 를 사용하여 Collect 메서드를 사용하여 메시지를 캡처합니다.

```
$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");
```

결과 객체에 메시지 적용:

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

플러그인 스텁 사용

사용자 지정 플러그인은 플러그인 스텁을 노출해야 합니다. SnapCenter 서버가 워크플로에 따라 호출하는 메서드입니다.

플러그인 스텁	선택 사항/필수 요소입니다	목적
설정	필수 요소입니다	이 스텁은 환경과 구성 개체를 설정합니다. 모든 환경 구문 분석 또는 처리는 여기에서 수행해야 합니다. 스텁이 호출될 때마다 setenv 스텁이 바로 전에 호출됩니다. PERL 스타일 플러그인에만 필요합니다.
버전	선택 사항	이 스텁은 응용 프로그램 버전을 가져오는 데 사용됩니다.
파악	선택 사항	이 스텁은 에이전트나 호스트에서 호스팅되는 인스턴스 또는 데이터베이스와 같은 애플리케이션 객체를 검색하는 데 사용됩니다. 플러그인은 응답의 일부로 검색된 애플리케이션 객체를 특정 형식으로 반환해야 합니다. 이 스텁은 응용 프로그램이 Unix용 SnapDrive와 통합된 경우에만 사용됩니다. Linux 파일 시스템(Linux 파일 시스템)이 지원됩니다. AIX/Solaris(Unix 유형)는 지원되지 않습니다.
discovery_complete(검색 완료	선택 사항	이 스텁은 에이전트나 호스트에서 호스팅되는 인스턴스 또는 데이터베이스와 같은 애플리케이션 객체를 검색하는 데 사용됩니다. 플러그인은 응답의 일부로 검색된 애플리케이션 객체를 특정 형식으로 반환해야 합니다. 이 스텁은 응용 프로그램이 Unix용 SnapDrive와 통합된 경우에만 사용됩니다. Linux 파일 시스템(Linux 파일 시스템)이 지원됩니다. AIX 및 Solaris(Unix 유형)는 지원되지 않습니다.

플러그인 스텁	선택 사항/필수 요소입니다	목적
정지	필수 요소입니다	이 스텁은 일시 중지를 수행합니다. 즉, 스냅샷을 생성할 수 있는 상태로 애플리케이션을 배치합니다. 이를 스냅샷 작업 이전이라고 합니다. 보존할 애플리케이션의 메타데이터는 응답의 일부로 설정되어야 하며, 이 메타데이터는 구성 매개 변수의 형태로 해당 스토리지 스냅샷의 후속 클론 또는 복원 작업 중에 반환됩니다.
정지 해제	필수 요소입니다	이 스텁은 중지 해제를 수행하는 역할을 하며, 이는 애플리케이션을 정상 상태로 전환하는 것을 의미합니다. 스냅샷을 생성한 후에 이 메시지가 호출됩니다.
Clone_pre	선택 사항	이 스텁은 사전 클론 작업을 수행합니다. 기본 제공 SnapCenter 서버 클론 생성 인터페이스를 사용 중이며 클론 작업을 수행할 때 트리거됩니다.
clone_post	선택 사항	이 스텁은 사후 클론 작업을 수행하는 역할을 합니다. 이는 사용자가 기본 제공 SnapCenter 서버 클론 생성 인터페이스를 사용하고 있다고 가정하고 클론 작업을 수행할 때만 트리거됩니다.
restore_pre	선택 사항	이 스텁은 PreRestore 작업을 수행하는 역할을 합니다. 이는 사용자가 기본 제공 SnapCenter 서버 복원 인터페이스를 사용하고 있으며 복원 작업을 수행하는 동안 트리거된다고 가정합니다.
복원	선택 사항	이 스텁은 애플리케이션 복구 작업을 수행하는 역할을 합니다. 이는 사용자가 기본 제공 SnapCenter 서버 복원 인터페이스를 사용하고 있다고 가정하고 복원 작업을 수행할 때만 트리거됩니다.

플러그인 스텁	선택 사항/필수 요소입니다	목적
정리	선택 사항	이 스텁은 백업, 복구 또는 클론 작업 후 정리 작업을 수행합니다. 정리 작업은 정상적인 워크플로 실행 중 또는 워크플로 오류가 발생한 경우에 가능합니다. 백업, cloneVolAndLun 또는 fileOrVolRestore 등의 구성 매개 변수 작업을 참조하여 정리 작업이 호출되는 워크플로 이름을 유추할 수 있습니다. 구성 매개 변수 ERROR_MESSAGE는 워크플로우를 실행하는 동안 오류가 있는지 여부를 나타냅니다. ERROR_MESSAGE가 정의되어 있고 NULL이 아닌 경우 Workflow 장애 실행 중에 정리가 호출됩니다.
APP_VERSION	선택 사항	이 스텁은 SnapCenter에서 플러그인으로 관리되는 애플리케이션 버전 세부 정보를 가져오는 데 사용됩니다.

플러그인 패키지 정보

모든 플러그인에는 다음 정보가 있어야 합니다.

```
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 = ();
```

운영

사용자 지정 플러그인에서 지원하는 setenv, Version, Quiesce 및 Unquiesce와 같은 다양한 작업을 코딩할 수 있습니다.

setenv 작동

PERL을 사용하여 만든 플러그인에는 setenv 작업이 필요합니다. ENV를 설정하고 플러그인 매개변수에 쉽게 액세스할 수 있습니다.

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

버전 작업

버전 작업은 응용 프로그램 버전 정보를 반환합니다.

```
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;
}
```

중지 작업

Quiesce 작업은 resources 매개 변수에 나열된 리소스에 대해 응용 프로그램 중지 작업을 수행합니다.

```
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;
}
```

작업 중지 해제

응용 프로그램 정지 해제를 위해서는 중지 해제 작업이 필요합니다. 리소스 목록은 resources 매개 변수에서 사용할 수 있습니다.

```
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;
}
```

네이티브 스타일

SnapCenter는 플러그인 생성을 위해 비 PERL 프로그래밍 또는 스크립팅 언어를 지원합니다. 이를 스크립트 또는 배치 파일일 수 있는 네이티브 스타일 프로그래밍이라고 합니다.

네이티브 스타일 플러그인은 다음과 같은 특정 규칙을 따라야 합니다.

플러그인은 실행 가능해야 합니다

- Unix 시스템의 경우 에이전트를 실행하는 사용자는 플러그인에 대한 실행 권한이 있어야 합니다
- Windows 시스템의 경우 PowerShell 플러그인에는 접미사가 .ps1이어야 하며, 다른 Windows 스크립트에는 .cmd 또는 .bat 접미사가 있어야 하며 사용자가 실행할 수 있어야 합니다
- 플러그인은 "-quiesce", "-unquiesce"와 같은 명령줄 인수에 반응해야 합니다.
- 작업 또는 기능이 구현되지 않은 경우 플러그인은 종료 코드 99를 반환해야 합니다
- 플러그인은 특정 구문을 사용하여 결과를 서버로 다시 전달해야 합니다

일반적인 플러그인 처리

오류 메시지를 로깅합니다

각 작업은 콘텐츠를 표시하고 저장하는 서버로 메시지를 다시 보낼 수 있습니다. 메시지에는 메시지 수준, 타임스탬프 및 메시지 텍스트가 포함됩니다. 여러 줄 메시지가 지원됩니다.

형식:

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

플러그인 스텁 사용

SnapCenter 플러그인은 플러그인 스텁을 구현해야 합니다. SnapCenter 서버가 특정 워크플로를 기반으로 호출하는 메서드입니다.

플러그인 스텁	선택 사항/필수 요소입니다	목적
정지	필수 요소입니다	이 스텁은 정지 작업을 수행합니다. 애플리케이션이 스냅샷을 생성할 수 있는 상태로 배치됩니다. 이 작업을 스토리지 스냅샷 작업 전에 호출합니다.
정지 해제	필수 요소입니다	이 스텁은 정지 해제를 수행하는 역할을 합니다. 응용 프로그램을 정상 상태로 만듭니다. 이를 스토리지 스냅샷 작업 후라고 합니다.
Clone_pre	선택 사항	이 스텁은 사전 클론 작업을 수행합니다. 이는 사용자가 기본 제공 SnapCenter 클론 생성 인터페이스를 사용하고 있으며 작업 "clone_vol 또는 clone_lun"을 수행하는 동안에만 트리거된다는 것을 전제로 합니다.

플러그인 스텝	선택 사항/필수 요소입니다	목적
clone_post	선택 사항	이 스텝은 사후 클론 작업을 수행하는 역할을 합니다. 이는 사용자가 기본 제공 SnapCenter 클론 생성 인터페이스를 사용 중이며 "clone_vol 또는 clone_lun" 작업을 수행하는 동안에만 트리거된다고 가정합니다.
restore_pre	선택 사항	이 스텝은 사전 복원 작업을 수행하는 역할을 합니다. 이는 사용자가 기본 제공 SnapCenter 복원 인터페이스를 사용하고 있다고 가정하고 복원 작업을 수행하는 동안에만 트리거됩니다.
복원	선택 사항	이 스텝은 모든 복구 작업을 수행합니다. 기본 제공 복원 인터페이스를 사용하지 않는 것으로 가정합니다. 복구 작업을 수행하는 동안 트리거됩니다.

예

Windows PowerShell

시스템에서 스크립트를 실행할 수 있는지 확인합니다. 스크립트를 실행할 수 없는 경우 스크립트에 대해 Set-ExecutionPolicy bypass를 설정하고 작업을 재시도하십시오.

```
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 스타일

Java 사용자 지정 플러그인은 데이터베이스, 인스턴스 등과 같은 응용 프로그램과 직접 상호 작용합니다.

제한 사항

Java 프로그래밍 언어를 사용하여 플러그인을 개발하는 동안 알아야 할 몇 가지 제한 사항이 있습니다.

플러그인 특성	**Java** 플러그인
복잡성	낮음-중간

플러그인 특성	**Java** 플러그인
메모리 공간	최대 10-20MB
다른 라이브러리에 대한 종속성	응용 프로그램 통신용 라이브러리
스레드 수입니다	1
스레드 런타임	1시간 미만

Java 제한에 대한 이유

SnapCenter 에이전트의 목표는 지속적이고 안전하고 강력한 애플리케이션 통합을 보장하는 것입니다. Java 플러그인을 지원함으로써 플러그인에서 메모리 누수 및 기타 원치 않는 문제를 일으킬 수 있습니다. 이러한 문제는 해결하기 어려우며, 특히 사용이 간편한 상태를 유지하는 것이 목표일 때 더욱 그렇습니다. 플러그인의 복잡성이 너무 복잡하지 않으면 개발자가 오류를 발생했을 가능성이 훨씬 줄어듭니다. Java 플러그인의 위험은 SnapCenter 에이전트와 동일한 JVM에서 실행된다는 것입니다. 플러그인이 충돌하거나 메모리를 누출하면 Agent에도 부정적인 영향을 미칠 수 있습니다.

지원되는 방법

방법	필수 요소입니다	설명	언제 누가 부르는가**?**
버전	예	플러그인 버전을 반환해야 합니다.	SnapCenter 서버 또는 에이전트가 플러그인 버전을 요청합니다.
정지	예	응용 프로그램에서 일시 중지를 수행해야 합니다. 이는 대부분의 경우 SnapCenter 서버가 백업을 생성할 수 있는 상태로 애플리케이션을 전환한다는 것을 의미합니다(예: 스냅샷).	SnapCenter 서버가 스냅샷 복사본을 생성하거나 일반적으로 백업을 수행하기 전에
정지 해제	예	응용 프로그램에서 정지 해제를 수행해야 합니다. 대부분의 경우 이는 응용 프로그램을 다시 정상 작동 상태로 전환하는 것을 의미합니다.	SnapCenter 서버가 스냅샷을 생성하거나 일반적으로 백업을 수행한 후
정리	아니요	플러그인에서 청소해야 하는 모든 것을 정리하는 역할을 합니다.	SnapCenter 서버의 워크플로가 완료되면(성공 또는 실패)

방법	필수 요소입니다	설명	언제 누가 부르는가?
ClonePre(사전	아니요	클론 작업을 수행하기 전에 수행해야 하는 작업을 수행해야 합니다.	사용자가 "cloneVol" 또는 "cloneLun" 작업을 트리거하고 내장된 클론 생성 마법사(GUI/CLI)를 사용하는 경우
ClonePost	아니요	클론 작업을 수행한 후 수행해야 하는 작업을 수행해야 합니다.	사용자가 "cloneVol" 또는 "cloneLun" 작업을 트리거하고 내장된 클론 생성 마법사(GUI/CLI)를 사용하는 경우
RestorePre	아니요	복구 작업을 호출하기 전에 수행해야 하는 작업을 수행해야 합니다.	사용자가 복구 작업을 트리거하는 경우
복원	아니요	애플리케이션의 복원/복구를 수행합니다.	사용자가 복구 작업을 트리거하는 경우
AppVersion(애플리케이션 버전)	아니요	플러그인으로 관리되는 응용 프로그램 버전을 검색합니다.	모든 워크플로우에서 백업/복원/클론과 같은 ASUP 데이터 수집의 일부

자습서

이 섹션에서는 Java 프로그래밍 언어를 사용하여 사용자 지정 플러그인을 만드는 방법에 대해 설명합니다.

일식을 설정합니다

1. Eclipse에서 새 Java 프로젝트 "TutorialPlugin"을 만듭니다
2. 마침 * 을 클릭합니다
3. 새 프로젝트 * → * 속성 * → * Java 빌드 경로 * → * 라이브러리 * → * 외부 jar 추가 * 를 마우스 오른쪽 버튼으로 클릭합니다
4. Host Agent의 ../lib/folder로 이동하여 jar scAgent-5.0-core.jar 및 common-5.0.jar 를 선택합니다
5. 프로젝트를 선택하고 * src 폴더 * → * New * → * Package * 를 마우스 오른쪽 단추로 클릭한 다음 com.netapp.snapcreator.agent.plugin.TutorialPlugin 이름의 새 패키지를 만듭니다
6. 새 패키지를 마우스 오른쪽 단추로 클릭하고 새로 만들기 → Java 클래스 를 선택합니다.
   a. 이름을 TutorialPlugin 으로 입력합니다.
   b. 수퍼클래스 찾아보기 단추를 클릭하고 "* AbstractPlugin"을 검색합니다. 하나의 결과만 표시되어야 합니다.

```
 "AbstractPlugin - com.netapp.snapcreator.agent.nextgen.plugin".
.. 마침 * 을 클릭합니다.
.. Java 클래스:
```

```
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;
  }
}
```

필요한 방법을 구현합니다

Quiesce, Unquiesce 및 version은 각 사용자 지정 Java 플러그인이 구현해야 하는 필수 메서드입니다.

다음은 플러그인 버전을 반환하는 버전 방법입니다.

```
@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;
}
```

이 메서드는 Context 개체에 전달됩니다. 여기에는 Logger 및 Context Store 같은 여러 도우미뿐만 아니라 현재 작업에 대한 정보(workflow-ID, job-ID)도 포함됩니다. FINAL Logger = CONTEXT.getLogger(); 를 호출하여 로거를 가져올 수 있습니다. Logger 개체는 다른 로깅 프레임워크에서 알려진 유사한 메서드(예: logback)를 제공합니다. 결과 개체에서 종료 코드를 지정할 수도 있습니다. 이 예제에서는 문제가 없으므로 0이 반환됩니다. 다른 종료 코드는 다른

실패 시나리오에 매핑할 수 있습니다.

결과 개체 사용

결과 개체에는 다음 매개 변수가 포함됩니다.

매개 변수	기본값	설명
구성	구성이 비어 있습니다	이 매개 변수는 구성 매개 변수를 서버로 다시 보내는 데 사용할 수 있습니다. 플러그인이 업데이트하려는 매개 변수가 될 수 있습니다. 이 변경 사항이 SnapCenter Server의 구성에 실제로 반영되는지 여부는 config의 app_CONF_persistency=Y 또는 N 매개 변수에 따라 달라집니다.
ExitCode를 참조하십시오	0	작업의 상태를 나타냅니다. "0"은 작업이 성공적으로 실행되었음을 의미합니다. 다른 값은 오류 또는 경고를 나타냅니다.
Stdout(스토우아웃)	목록이 비어 있습니다	이 기능을 사용하여 stdout 메시지를 SnapCenter 서버로 다시 전송할 수 있습니다.
Stderr	목록이 비어 있습니다	stderr 메시지를 SnapCenter 서버로 다시 전송하는 데 사용할 수 있습니다.
메시지	목록이 비어 있습니다	이 목록에는 플러그인에서 서버로 반환하려는 모든 메시지가 포함되어 있습니다. SnapCenter 서버는 이러한 메시지를 CLI 또는 GUI에 표시합니다.

SnapCenter 에이전트는 모든 결과 형식에 대해 Builders ("작성기 패턴")를 제공합니다. 따라서 다음과 같이 매우 간단하게 사용할 수 있습니다.

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

예를 들어, 종료 코드를 0으로 설정하고, stdout 및 stderr에 대한 목록을 설정하고, config 매개 변수를 설정하고, 서버로 다시 전송될 로그 메시지를 추가합니다. 모든 매개 변수가 필요하지 않으면 필요한 매개 변수만 보냅니다. 각 매개 변수에는 기본값이 있으므로 아래 코드에서 .withExitCode(0)를 제거하면 결과는 영향을 받지 않습니다.

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

버전

VersionResult 는 SnapCenter 서버에 플러그인 버전을 알립니다. 또한 result 에서 상속되므로 config, exitCode, stdout, stderr 및 messages 매개 변수가 포함됩니다.

매개 변수	기본값	설명
전공	0	플러그인의 주 버전 필드입니다.
경미합니다	0	플러그인의 부 버전 필드입니다.
패치	0	플러그인의 패치 버전 필드입니다.
빌드	0	플러그인의 빌드 버전 필드입니다.

예를 들면 다음과 같습니다.

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

컨텍스트 객체 사용

컨텍스트 개체는 다음 메서드를 제공합니다.

컨텍스트 방법입니다	목적
문자열 getWorkflowId();	현재 워크플로에 대해 SnapCenter 서버에서 사용 중인 워크플로 ID를 반환합니다.
구성 getconfig();	SnapCenter 서버에서 에이전트로 보내는 구성을 반환합니다.

Workflow-ID입니다

workflow-ID는 SnapCenter 서버가 실행 중인 특정 워크플로를 참조하는 데 사용하는 ID입니다.

구성

이 개체에는 사용자가 SnapCenter 서버의 config에서 설정할 수 있는 매개 변수가 대부분 포함되어 있습니다. 그러나 보안상의 이유로 이러한 매개 변수 중 일부는 서버 측에서 필터링될 수 있습니다. 다음은 Config에 액세스하고 매개 변수를 검색하는 방법에 대한 예입니다.

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

""//myParameter"에 이제 SnapCenter 서버의 구성에서 읽은 매개 변수가 포함됩니다. 구성 매개 변수 키가 없으면 빈 문자열("")이 반환됩니다.

플러그인을 내보내는 중입니다

SnapCenter 호스트에 설치하려면 플러그인을 내보내야 합니다.

Eclipse에서 다음 작업을 수행합니다.

1. 플러그인의 기본 패키지를 마우스 오른쪽 단추로 클릭합니다(예: com.netapp.snapcreator.agent.plugin.TutorialPlugin).
2. 내보내기 * → * Java * → * JAR 파일 * 을 선택합니다
3. 다음 * 을 클릭합니다.
4. 다음 창에서 대상 jar 파일 경로를 지정합니다. tutorial_plugin.jar 플러그인의 기본 클래스는 TutorialPlugin.class 로 명명되며, 동일한 이름의 폴더에 플러그인을 추가해야 합니다.

플러그인이 추가 라이브러리에 종속된 경우 lib / 폴더를 만들 수 있습니다

플러그인이 종속된 jar 파일을 추가할 수 있습니다(예: 데이터베이스 드라이버). SnapCenter가 플러그인을 로드하면 이 폴더의 모든 jar 파일이 자동으로 해당 파일과 연관되고 classpath에 추가됩니다.

SnapCenter의 사용자 지정 플러그인

SnapCenter의 사용자 지정 플러그인

Java, PERL 또는 네이티브 스타일을 사용하여 만든 사용자 지정 플러그인은 SnapCenter 서버를 사용하여 호스트에 설치하여 응용 프로그램의 데이터 보호를 활성화할 수 있습니다. 이 자습서에 제공된 절차를 사용하여 SnapCenter 호스트에 설치하려면 플러그인을 내보내야 합니다.

플러그인 설명 파일 생성

생성된 모든 플러그인에 대해 설명 파일이 있어야 합니다. 설명 파일은 플러그인의 세부 정보를 설명합니다. 파일 이름은 Plugin_descriptor.xml이어야 합니다.

플러그인 설명자 파일 특성 및 그 중요성을 사용합니다

속성	설명
이름	플러그인 이름입니다. 영숫자를 사용할 수 있습니다. 예: DB2, MySQL, MongoDB 네이티브 스타일로 만들어진 플러그인의 경우 파일 확장명을 제공하지 않아야 합니다. 예를 들어 플러그인 이름이 mongodb.sh인 경우 이름을 mongodb로 지정합니다.
버전	플러그인 버전입니다. 주 버전과 부 버전을 모두 포함할 수 있습니다. 예: 1.0, 1.1, 2.0, 2.1
표시 이름	SnapCenter 서버에 표시할 플러그인 이름입니다. 동일한 플러그인의 여러 버전이 기록되는 경우 모든 버전에서 표시 이름이 동일한지 확인하십시오.
PluginType입니다	플러그인을 생성하는 데 사용되는 언어입니다. 지원되는 값은 Perl, Java 및 Native입니다. 기본 플러그인 유형에는 Unix/Linux 셸 스크립트, Windows 스크립트, Python 또는 기타 스크립팅 언어가 포함됩니다.
OSNAME	플러그인이 설치된 호스트 OS 이름입니다. 유효한 값은 Windows 및 Linux입니다. PERL 유형 플러그인과 같은 여러 OS 유형에서 단일 플러그인을 배포할 수 있습니다.
OSVersion(OS버전)	플러그인이 설치된 호스트 OS 버전입니다.
리소스 이름	플러그인이 지원할 수 있는 리소스 유형의 이름입니다. 예를 들어 데이터베이스, 인스턴스, 컬렉션 등이 있습니다.
모체	이 경우 ResourceName 은 다른 리소스 형식에 따라 계층적으로 종속된 다음 Parent 는 부모 ResourceType 을 결정합니다. 예를 들어, DB2 플러그인에서 ResourceName “Database”에는 부모 “인스턴스”가 있습니다.
RequireFileSystemPlugin	예 또는 아니요 복구 탭에 복구 탭이 표시되는지 여부를 결정합니다.
ResourceRequiresAuthentication을 참조하십시오	예 또는 아니요 자동 검색되거나 자동 검색되지 않은 리소스에 스토리지 검색 후 데이터 보호 작업을 수행하는 데 자격 증명이 필요한지 여부를 결정합니다.
RequireFileSystemClone 을 참조하십시오	예 또는 아니요 플러그인에서 클론 워크플로우를 위해 FileSystem 플러그인 통합이 필요한지 여부를 결정합니다.

사용자 지정 플러그인 DB2에 대한 Plugin_descriptor.xml 파일의 예는 다음과 같습니다.

```
<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>
```

ZIP 파일 생성

플러그인을 개발하고 설명자 파일을 만든 후에는 플러그인 파일과 Plugin_descriptor.xml 파일을 폴더에 추가하고 압축해야 합니다.

ZIP 파일을 작성하기 전에 다음 사항을 고려해야 합니다.

- 스크립트 이름은 플러그인 이름과 같아야 합니다.
- PERL 플러그인의 경우 ZIP 폴더에 스크립트 파일이 있는 폴더가 있어야 하고 설명자 파일이 이 폴더 외부에 있어야 합니다. 폴더 이름은 플러그인 이름과 같아야 합니다.
- PERL 플러그인 이외의 플러그인의 경우 ZIP 폴더에는 설명자와 스크립트 파일이 포함되어 있어야 합니다.
- OS 버전은 숫자여야 합니다.

예:

- DB2 플러그인: DB2.pm 및 Plugin_descriptor.xml 파일을 “DB2.zip”에 추가합니다.
- Java를 사용하여 개발된 플러그인: jar 파일, 종속 jar 파일 및 Plugin_descriptor.xml 파일을 폴더에 추가하고 압축합니다.

플러그인 **ZIP** 파일을 업로드하는 중입니다

원하는 호스트에 플러그인을 배포할 수 있도록 플러그인 ZIP 파일을 SnapCenter 서버에 업로드해야 합니다.

UI 또는 cmdlet을 사용하여 플러그인을 업로드할 수 있습니다.

- UI: *
- 플러그인 ZIP 파일을 * 추가 * 또는 * 호스트 수정 * 워크플로우 마법사의 일부로 업로드합니다
- “사용자 지정 플러그인을 업로드하려면 선택하십시오.” * 를 클릭합니다
- PowerShell: *
- Upload-SmPluginPackage cmdlet

  예를 들어, PS > 업로드 - SmPluginPackage - AbsolutePath c:\DB2_1.zip

  PowerShell cmdlet에 대한 자세한 내용은 SnapCenter cmdlet 도움말을 사용하거나 cmdlet 참조 정보를 참조하십시오.

"SnapCenter 소프트웨어 cmdlet 참조 가이드"..

사용자 지정 플러그인 배포

이제 업로드된 사용자 지정 플러그인을 * 추가 * 및 * 호스트 수정 * 워크플로의 일부로 원하는 호스트에 배포할 수 있습니다. 여러 버전의 플러그인을 SnapCenter 서버에 업로드할 수 있으며 특정 호스트에 배포할 버전을 선택할 수 있습니다.

플러그인을 업로드하는 방법에 대한 자세한 내용은 을 참조하십시오. "호스트를 추가하고 원격 호스트에 플러그인 패키지를 설치합니다"