



アプリケーション用のプラグインを開発 SnapCenter Software 6.0

NetApp
September 23, 2024

目次

アプリケーション用のプラグインを開発.....	1
概要	1
Perlベースの開発.....	3
ネイティブ形式	11
Javaスタイル.....	14
SnapCenterのカスタムプラグイン	22

アプリケーション用のプラグインを開発

概要

SnapCenterサーバを使用すると、アプリケーションをSnapCenterのプラグインとして導入および管理できます。データ保護機能と管理機能を備えた SnapCenter サーバに、お好みのアプリケーションを接続できます。

SnapCenterでは、さまざまなプログラミング言語を使用してカスタムプラグインを開発できます。Perl、Java、バッチ、またはその他のスクリプト言語を使用してカスタムプラグインを開発できます。

SnapCenterでカスタムプラグインを使用するには、次のタスクを実行する必要があります。

- このガイドの手順に従って、アプリケーション用のプラグインを作成します。
- 記述ファイルの作成
- カスタムプラグインをエクスポートしてSnapCenterホストにインストールする
- プラグインのzipファイルをSnapCenterサーバにアップロードする

すべてのAPI呼び出しでの汎用プラグインの処理

API呼び出しごとに、次の情報を使用します。

- プラグインパラメータ
- 終了コード
- エラーメッセージをログに記録
- データの整合性

プラグインパラメータを使用

API呼び出しごとに一連のパラメータがプラグインに渡されます。次の表に、各パラメータの具体的な情報を示します。

パラメータ	目的
アクション	ワークフロー名を指定します。たとえば、discover、backup、fileOrVolRestore、またはcloneVolAndLun などです
リソース	保護するリソースを一覧表示します。リソースはUIDとタイプで識別されます。リストは次の形式でプラグインに表示されます。 「<UID>、<type>; <UID>、<type>」のように入力します。例：「Instance1、Instance ; Instance2\\DB1、Database」

パラメータ	目的
app_name	使用しているプラグインを指定します。たとえば、DB2、MySQLなどです。SnapCenterサーバには、リストされているアプリケーションのサポートが組み込まれています。このパラメータでは大文字と小文字が区別されます。
APP_IGNORE_ERROR	(YまたはN) これにより、アプリケーションエラーが発生したときにSnapCenterが終了するか、終了しません。これは、複数のデータベースをバックアップする場合に、単一障害でバックアップ処理を停止しないようにする場合に便利です。
<resource_name> ____APP_INSTANY_USERNAME	リソースに対してSnapCenterクレデンシャルが設定されている。
<resource_name> _APP_INSTANY_PASSWORD	リソースに対してSnapCenterクレデンシャルが設定されている。
<resource_name> _<custom_param> です	すべてのリソースレベルのカスタムキー値は、先頭に「<resource_name>_」を付けたプラグインで使用できます。たとえば、カスタムキーが「MySQLDB」という名前のリソースの「MASTER_SLAVE」である場合、このキーはMySQLDB_MASTER_SLAVEとして使用できます

終了コードを使用する

プラグインは、終了コードを使用して処理のステータスをホストに返します。各コードには特定の意味があり、プラグインは正しい終了コードを使用して同じことを示します。

次の表に、エラーコードとその意味を示します。

終了コード	目的
0	処理に成功しました。
99	要求された操作はサポートされていないか、
100	処理に失敗しました。休止解除をスキップして終了します。デフォルトでは休止解除が選択されます。
101	処理に失敗しました。バックアップ処理を続行してください。
その他	処理に失敗しました。休止解除を実行して終了します。

エラーメッセージをログに記録

エラー・メッセージは、プラグインからSnapCenterサーバに渡されます。メッセージには、メッセージ、ログレベル、およびタイムスタンプが含まれます。

次の表に、レベルとその目的を示します。

パラメータ	目的
情報	情報メッセージ
警告	警告メッセージ
エラー	エラーメッセージ
デバッグ	デバッグメッセージ
トレース	トレースメッセージ

データの整合性を維持

カスタムプラグインでは、同じワークフローを実行してもデータが保持されます。たとえば、プラグインは休止の終了時にデータを格納でき、休止解除処理に使用できます。

保持するデータは、プラグインによって結果オブジェクトの一部として設定されます。特定の形式に従っており、プラグイン開発の各スタイルで詳細に説明されています。

Perlベースの開発

Perlを使用してプラグインを開発するときは、特定の規則に従う必要があります。

- コンテンツは読み取り可能でなければなりません
- `setenv`、`quiesce`、および`unquiesce`の必須処理を実装する必要がある
- 結果をエージェントに渡すには、特定の構文を使用する必要があります。
- 内容は `<plugin_name>.pm` ファイルとして保存してください

実行可能な処理：

- `setenv`
- バージョン
- 休止
- 休止解除
- `clone_pre`、`clone_post`
- `restore_pre`、リストア

- クリーンアップ

一般的なプラグイン処理

結果オブジェクトの使用

すべてのカスタムプラグイン処理で結果オブジェクトを定義する必要があります。このオブジェクトは、メッセージ、終了コード、stdout、およびstderrをホストエージェントに送信します。

結果オブジェクト：

```
my $result = {
```

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

結果オブジェクトを返します。

```
return $result;
```

データの整合性の維持

同じワークフローの実行中に、処理間（クリーンアップを除く）でデータを保持することができます。これにはキーと値のペアを使用します。データのキーと値のペアは結果オブジェクトの一部として設定され、同じワークフローの後続の操作で保持されて使用できます。

次のコード例では、保持するデータを設定します。

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

上記のコードは、2つのキーと値のペアを設定します。これらのペアは、後続の操作で入力として使用できます。2つのキーと値のペアには、次のコードを使用してアクセスできます。

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

=== Logging error messages

各処理は、コンテンツを表示して保存するホストエージェントにメッセージを送信できます。メッセージには、メッセージレベル、タイムスタンプ、およびメッセージテキストが含まれます。複数行メッセージがサポートされています。

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

`msgObj`を使用して、`collect`メソッドを使用してメッセージをキャプチャします。

```
$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");
```


結果オブジェクトにメッセージを適用します。

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

プラグインスタブの使用

カスタムプラグインはプラグインスタブを公開する必要があります。これらは、ワークフローに基づいてSnapCenterサーバが呼び出すメソッドです。

プラグインスタブ	オプション / 必須	目的
setenv	必須	<p>このスタブは、環境と設定オブジェクトを設定します。</p> <p>環境の解析や処理はここで行う必要があります。スタブが呼び出されるたびに、setenvスタブが直前に呼び出されます。Perl形式のプラグインでのみ必要です。</p>
バージョン	オプション	<p>このスタブは、アプリケーションのバージョンを取得するために使用されます。</p>
検出	オプション	<p>このスタブは、エージェントまたはホストでホストされているインスタンスやデータベースなどのアプリケーションオブジェクトを検出するために使用されます。</p> <p>プラグインは、検出されたアプリケーションオブジェクトを特定の形式で応答の一部として返します。このスタブは、アプリケーションがSnapDrive for Unixと統合されている場合にのみ使用されます。</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;">  <p>Linuxファイルシステム (Linuxフレーバー) がサポートされています。AIX/Solaris (Unixフレーバー) はサポートされていません。</p> </div>

プラグインスタブ	オプション / 必須	目的
検出_完了	オプション	<p>このスタブは、エージェントまたはホストでホストされているインスタンスやデータベースなどのアプリケーションオブジェクトを検出するために使用されます。</p> <p>プラグインは、検出されたアプリケーションオブジェクトを特定の形式で応答の一部として返します。このスタブは、アプリケーションがSnapDrive for Unixと統合されている場合にのみ使用されます。</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;">  <p>Linuxファイルシステム（Linuxフレイバー）がサポートされています。AIXおよびSolaris（Unixフレイバー）はサポートされていません。</p> </div>
休止	必須	<p>このスタブは休止を実行します。つまり、アプリケーションをSnapshotを作成できる状態にします。これは、Snapshot処理の前に呼び出されます。保持するアプリケーションのメタデータは、応答の一部として設定する必要があります。このメタデータは、対応するストレージSnapshotでの後続のクローニングまたはリストア処理中に、構成パラメータの形式で返されます。</p>
休止解除	必須	<p>このスタブは、アプリケーションを通常の状態にすることを意味する休止解除を実行します。これは、Snapshotの作成後に呼び出されます。</p>
clone_pre	オプション	<p>このスタブは、クローニング前タスクを実行します。これは、組み込みのSnapCenterサーバクローニングインターフェイスを使用していることを前提としており、クローニング処理の実行時にトリガーされます。</p>

プラグインスタブ	オプション / 必須	目的
clone_post	オプション	このスタブは、クローニング後のタスクを実行します。これは、組み込みのSnapCenterサーバクローニングインターフェイスを使用していることを前提としており、クローニング処理の実行時にのみトリガーされます。
restore_pre	オプション	このスタブは、リストア前のタスクを実行します。ここでは、組み込みのSnapCenterサーバリストアインターフェイスを使用しており、リストア処理の実行中にトリガーされることを前提としています。
リストア	オプション	このスタブは、アプリケーションのリストアタスクを実行します。これは、組み込みのSnapCenterサーバリストアインターフェイスを使用していることを前提としており、リストア処理の実行時にのみトリガーされます。
クリーンアップ	オプション	このスタブは、バックアップ、リストア、またはクローン処理のあとにクリーンアップを実行します。クリーンアップは、通常のワークフロー実行中またはワークフローの障害発生時に実行できません。設定パラメータaction (backup、cloneVolAndLun、fileOrVolRestore) を参照して、クリーンアップが呼び出されるワークフロー名を推測できます。構成パラメータERROR_MESSAGEは、ワークフローの実行中にエラーが発生したかどうかを示します。ERROR_MESSAGEがNULLではなく定義されている場合、ワークフローエラーの実行中にクリーンアップが呼び出されます。
APP_VERSION	オプション	このスタブは、SnapCenter がプラグインによって管理されるアプリケーションバージョンの詳細を取得するために使用されます。

プラグインパッケージ情報

各プラグインには、次の情報が必要です。

```
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 = ();
```

運用

カスタムプラグインでは、setenv、バージョン、休止、休止解除など、さまざまな処理をコーディングできます。

setenv処理setenvシヨリ

setenv処理は、Perlを使用して作成されたプラグインに必要です。ENVを設定し、プラグインパラメータに簡単にアクセスできます。

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

バージョン処理

バージョン処理は、アプリケーションのバージョン情報を返します。

```
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;
}
```

休止処理

休止処理resourcesパラメータに指定されたリソースに対してアプリケーション休止処理を実行します。

```
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;
}
```

休止解除処理

アプリケーションの休止解除には休止解除処理が必要です。リソースのリストは、resourcesパラメータで確認できます。

```

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;
}

```

ネイティブ形式

SnapCenterでは、プラグインを作成するためにPerl以外のプログラミング言語やスクリプト言語がサポートされています。これはネイティブスタイルプログラミングと呼ばれ、スクリプトファイルまたはバッチファイルを使用できます。

ネイティブ形式のプラグインは、以下に示す特定の規則に従う必要があります。

プラグインは実行可能である必要があります

- UNIXシステムの場合、エージェントを実行するユーザーにはプラグインに対する実行権限が必要です。
- Windows システムの場合、PowerShell プラグインのサフィックスは .ps1 に、その他の Windows スクリプトのサフィックスは .cmd または .bat にする必要があります、ユーザによって実行可能である必要があります
- プラグインは、コマンドライン引数に対して「-quiesce」、 「-unquiesce」のように応答する必要があります。
- 操作または関数が実装されていない場合、プラグインは終了コード99を返す必要があります。
- プラグインは、結果をサーバーに返すために特定の構文を使用する必要があります。

一般的なプラグイン処理

エラーメッセージのロギング

各操作は、コンテンツを表示して保存するサーバーにメッセージを送り返すことができます。メッセージには、メッセージレベル、タイムスタンプ、およびメッセージテキストが含まれます。複数行メッセージがサポートされています。

形式：

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

プラグインスタブの使用

SnapCenterプラグインにはプラグインスタブが実装されている必要があります。これらは、SnapCenterサーバが特定のワークフローに基づいて呼び出すメソッドです。

プラグインスタブ	オプション / 必須	目的
休止	必須	このスタブは休止を実行します。これにより、アプリケーションがスナップショットを作成できる状態になります。これは、ストレージSnapshot処理の前に呼び出されます。
休止解除	必須	このスタブは休止解除を実行します。アプリケーションは通常の状態になります。この処理は、ストレージSnapshot処理のあとに呼び出されます。
clone_pre	オプション	このスタブは、クローニング前のタスクを実行します。ここでは、組み込みのSnapCenterクローニングインターフェイスを使用しており、「clone_vol or clone_lun」アクションの実行時にのみトリガーされます。
clone_post	オプション	このスタブは、クローニング後のタスクを実行します。これは、組み込みのSnapCenterクローニングインターフェイスを使用しており、「clone_volまたはclone_lun」処理の実行時にのみトリガーされることを前提としています。
restore_pre	オプション	このスタブは、リストア前のタスクを実行します。ここでは、組み込みのSnapCenterリストアインターフェイスを使用しており、リストア処理の実行中にのみトリガーされます。

プラグインスタブ	オプション / 必須	目的
リストア	オプション	このスタブは、すべてのリストア処理を実行します。この要件は、組み込みのリストアインターフェイスを使用していないことを前提としています。リストア処理の実行中にトリガーされます。

例

Windows PowerShell

システムでスクリプトを実行できるかどうかを確認します。スクリプトを実行できない場合は、スクリプトにSet-ExecutionPolicyバイパスを設定し、操作を再試行します。

```

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スタイル

Javaカスタムプラグインは、データベースやインスタンスなどのアプリケーションと直接対話します。

制限事項

Javaプログラミング言語を使用してプラグインを開発するときは、いくつかの制限事項に注意する必要があります。

プラグインの特性	Javaプラグイン
複雑さ	低~中
メモリフットプリント	最大10~20 MB
他のライブラリへの依存	アプリケーション通信用ライブラリ
スレッド数	1
スレッドランタイム	1時間未満

Java制限の理由

SnapCenterエージェントの目標は、継続的かつ安全で堅牢なアプリケーション統合を実現することです。Javaプラグインをサポートすることで、プラグインがメモリリークなどの望ましくない問題を引き起こす可能性があります。これらの問題は、特に物事を使いやすくすることを目的としている場合には、取り組むのが難しいです。プラグインの複雑さがそれほど複雑でない場合、開発者がエラーを導入した可能性ははるかに低くなります。Java プラグインの危険性は、SnapCenter エージェント自体と同じ JVM で実行されていることです。プラグインがクラッシュしたり、メモリがリークしたりすると、Agentに悪影響を及ぼす可能性があります。

サポートされる方法

方法	必須	説明	いつ誰に電話したの？
バージョン	はい	プラグインのバージョンを返す必要があります。	SnapCenter サーバまたはエージェントがプラグインのバージョンを要求します。
休止	はい	アプリケーションで休止を実行する必要があります。ほとんどの場合、これは、アプリケーションをSnapCenterサーバがバックアップ（スナップショットなど）を作成できる状態にすることを意味します。	SnapCenter サーバが Snapshot コピーを作成する前、または一般的なバックアップを実行します。
休止解除	はい	アプリケーションで休止解除を実行する必要があります。ほとんどの場合、これはアプリケーションを通常の動作状態に戻すことを意味します。	SnapCenterサーバがスナップショットを作成した後、または一般的にバックアップを実行した後。

方法	必須	説明	いつ誰に電話したの？
クリーンアップ	いいえ	プラグインがクリーンアップする必要があるすべての処理を担当します。	SnapCenterサーバ上のワークフローが終了したとき（正常に完了したとき、または失敗したとき）。
clonePre	いいえ	クローニング処理の実行前に必要な処理を実行する必要があります。	ユーザが「cloneVol」または「cloneLun」アクションをトリガーし、組み込みのクローニングウィザード（GUI / CLI）を使用した場合。
clonePost	いいえ	クローニング処理の実行後に必要な処理を実行する必要があります。	ユーザが「cloneVol」または「cloneLun」アクションをトリガーし、組み込みのクローニングウィザード（GUI / CLI）を使用した場合。
restorePre	いいえ	は、リストア処理の呼び出し前に実行する必要があるアクションを実行する必要があります。	ユーザがリストア処理をトリガーしたとき。
リストア	いいえ	アプリケーションのリストア/リカバリを実行します。	ユーザがリストア処理をトリガーしたとき。
アプリケーションバージョン	いいえ	プラグインで管理されているアプリケーションのバージョンを取得します。	ASUPデータ収集の一環として、バックアップ/リストア/クローンなどのすべてのワークフローで使用できます。

チュートリアル

このセクションでは、Javaプログラミング言語を使用してカスタムプラグインを作成する方法について説明します。

Eclipseの設定

1. Eclipseで新しいJavaプロジェクト「TutorialPlugin」を作成する
2. [完了]をクリックします。
3. 新しいプロジェクト * → * プロパティ * → * Java ビルドパス * → * ライブラリ * → * 外部 JAR の追加 * を右クリックします

4. ホストエージェントの../lib/フォルダに移動し、jars scAgent-5.0-core.jarおよびcommon-5.0.jarを選択します。
5. プロジェクトを選択し、* src フォルダ * → * New * → * Package * を右クリックして、com.netapp.snapcreator.agent.plugin.TutorialPlugin という名前で新しいパッケージを作成します
6. 新しいパッケージを右クリックし'新規> Javaクラスを選択します
 - a. 名前に「TutorialPlugin」と入力します。
 - b. スーパークラスの参照ボタンをクリックし、「* AbstractPlugin」を検索します。表示される結果は1つだけです。

```
"AbstractPlugin - com.netapp.snapcreator.agent.nextgen.plugin".  
.. [ 完了 ] をクリックします。  
.. Javaクラス：
```

```

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;
    }
}

```

必要なメソッドの実装

休止、休止解除、およびバージョンは、各カスタムJavaプラグインで実装する必要がある必須のメソッドです。

プラグインのバージョンを返すversionメソッドを次に示します。

```

@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 plugin 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;
}

```

メソッドはContextオブジェクトで渡されます。これには、LoggerやContext Storeなどの複数のヘルパーと、現在の操作に関する情報（ワークフローID、ジョブID）が含まれます。ロガーを取得するには、final Logger logger=context.getLogger();を呼び出します。loggerオブジェクトは、logbackなど、他のロギングフレームワークで知られている同様のメソッドを提供します。結果オブジェクトでは、終了コードを指定することもできます。この例では問題がなかったため、0が返されます。その他の終了コードは、さまざまな障害シナリオにマッピングできます。

結果オブジェクトの使用

resultオブジェクトには、次のパラメータが含まれています。

パラメータ	デフォルト	説明
構成	構成が空です	このパラメータを使用すると、設定パラメータをサーバに返送できます。プラグインで更新するパラメータを指定できます。この変更が SnapCenter サーバの構成に実際に反映されるかどうかは、設定の APP_CONF_PERSISTENCE = Y または N パラメータに依存します。
終了コード	0	処理のステータスを示します。「0」は、操作が正常に実行されたことを示します。その他の値はエラーまたは警告を示します。
標準出力	リストが空です	これは、stdout メッセージを SnapCenter サーバに返送するために使用できます。
標準エラー	リストが空です	このオプションを使用すると、stderr メッセージを SnapCenter サーバに返送できます。
メッセージ	リストが空です	このリストには、プラグインがサーバに返すすべてのメッセージが含まれています。SnapCenterサーバは、これらのメッセージをCLIまたはGUIに表示します。

SnapCenterエージェントは、すべての結果タイプに対してビルダーを提供し ("[ビルダパターン](#)")ます。これにより、非常に簡単に使用できます。

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

たとえば、終了コードを0に設定し、stdoutとstderrのリストを設定し、configパラメータを設定し、サーバに返送されるログメッセージを追加します。すべてのパラメータが必要ない場合は、必要なパラメータのみを送信してください。各パラメータにはデフォルト値があるため、以下のコードから.withExitCode(0)を削除して

も、結果は影響を受けません。

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

VersionResult

VersionResultは、SnapCenterサーバーにプラグインのバージョンを通知します。また、result から継承されるため、config、exitCode、stdout、stderr、および messages パラメータが含まれます。

パラメータ	デフォルト	説明
メジャー	0	プラグインのメジャーバージョンフィールド。
マイナー	0	プラグインのマイナーバージョンフィールド。
パッチ	0	プラグインのパッチバージョンフィールド。
構築	0	プラグインのビルドバージョンフィールド。

例：

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

コンテキストオブジェクトの使用

contextオブジェクトには、次のメソッドがあります。

コンテキストメソッド	目的
文字列 getWorkflowId();	現在のワークフローで SnapCenter サーバによって使用されているワークフロー ID を返します。

コンテキストメソッド	目的
<code>config getConfig();</code>	SnapCenter サーバからエージェントに送信されている設定を返します。

ワークフローID

ワークフロー ID は、実行中の特定のワークフローを SnapCenter サーバが参照するために使用する ID です。

構成

このオブジェクトには、ユーザが SnapCenter サーバの設定で設定できるパラメータのほとんどが含まれます。ただし、セキュリティ上の理由から、これらのパラメータの一部はサーバ側でフィルタリングされる場合があります。次に、Config にアクセスしてパラメータを取得する例を示します。

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

""//MyParameter"に、SnapCenterサーバ上のconfigから読み込まれたパラメータが含まれるようになりました。configパラメータキーが存在しない場合は、空の文字列("")を返します。

プラグインのエクスポート

SnapCenterホストにインストールするには、プラグインをエクスポートする必要があります。

Eclipseで次のタスクを実行します。

1. プラグインのベースパッケージを右クリックします（この例では `com.netapp.snapcreator.agent.plugin.TutorialPlugin`）。
2. 「* Export * → * Java * → * JAR File *」を選択します
3. 「* 次へ *」をクリックします。
4. 次のウィンドウで、インストール先の jar ファイルのパスを指定します。 `tutorial_plugin.jar` プラグインのベースクラスは `TutorialPlugin.class` という名前で、同じ名前のフォルダにプラグインを追加する必要があります。

プラグインが他のライブラリに依存している場合は、次のフォルダを作成できます。 `lib/`

プラグインが依存するjarファイル（データベースドライバなど）を追加できます。SnapCenter は、プラグインをロードすると、このフォルダ内のすべての jar ファイルを自動的に関連付けて、クラスパスに追加します。

SnapCenterのカスタムプラグイン

SnapCenterのカスタムプラグイン

Java、Perl、またはネイティブ形式で作成したカスタムプラグインをSnapCenterサーバを使用してホストに

インストールし、アプリケーションのデータ保護を有効にすることができます。このチュートリアルで説明する手順に従って、プラグインをエクスポートしてSnapCenterホストにインストールしておく必要があります。

プラグイン記述ファイルの作成

作成するプラグインごとに、説明ファイルが必要です。定義ファイルには、プラグインの詳細が定義されています。ファイル名はPlugin_descriptor.xmlである必要があります。

プラグイン記述子ファイルの属性と重要度の使用

属性	説明
名前	プラグインの名前。英数字を使用できます。例： DB2、MySQL、MongoDB ネイティブ形式で作成されたプラグインの場合は、ファイルの拡張子を指定しないでください。たとえば、プラグイン名がMongoDB.shの場合は、名前をMongoDBと指定します。
バージョン	プラグインのバージョン。メジャーバージョンとマイナーバージョンの両方を含めることができます。 例：1.0、1.1、2.0、2.1
DisplayName	SnapCenter サーバに表示されるプラグインの名前。同じプラグインの複数のバージョンが書き込まれている場合は、表示名がすべてのバージョンで同じであることを確認してください。
プラグインタイプ	プラグインの作成に使用した言語。サポートされている値は、Perl、Java、およびNativeです。ネイティブプラグインタイプには、UNIX/Linuxシェルスクリプト、Windowsスクリプト、Python、またはその他のスクリプト言語が含まれます。
OSNAME	プラグインがインストールされているホストOSの名前。有効な値は Windows と Linux です。1つのプラグインを複数のOSタイプ（Perlタイプのプラグインなど）に導入することができます。
osVersion	プラグインがインストールされているホストOSのバージョン。
リソース名	プラグインがサポートできるリソースタイプの名前。たとえば、データベース、インスタンス、コレクションなどです。

属性	説明
親	<p>場合、 ResourceName は階層的に別のリソースタイプに依存し、 Parent は親のリソースタイプを決定します。</p> <p>たとえば、DB2プラグインの場合、 ResourceName の「Database」には親の「Instance」があります。</p>
RequireFileSystemPlugin	YesまたはNo リストアウィザードにリカバリタブを表示するかどうかを指定します。
ResourceRequiresAuthentication	YesまたはNo ストレージの検出後にデータ保護処理を実行するために、自動検出されたリソースと自動検出されなかったリソースのどちらにクレデンシャルが必要かを指定します。
RequireFileSystemClone	YesまたはNo クローンワークフローでファイルシステムプラグインの統合が必要かどうかを指定します。

カスタムプラグインDB2のPlugin_descriptor.xmlファイルの例を次に示します。

```

<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ファイルの作成

プラグインが開発されて記述子ファイルが作成されたら、プラグインファイルと Plugin_descriptor.xml ファイルをフォルダに追加して zip する必要があります。

ZIPファイルを作成する前に、次の点を考慮する必要があります。

- スクリプト名はプラグイン名と同じにする必要があります。
- Perl プラグインの場合、ZIP フォルダにスクリプトファイルが格納されているフォルダと、記述ファイルがこのフォルダの外部にある必要があります。フォルダ名はプラグイン名と同じである必要があります。
- Perl プラグイン以外のプラグインを使用する場合は、ZIP フォルダに記述子とスクリプトファイルが含まれている必要があります。
- OSのバージョンは数字である必要があります。

例：

- DB2プラグイン：db2.pmファイルとPlugin_descriptor.xmlファイルを「db2.zip」に追加します。
- Java を使用して開発されたプラグイン：jar ファイル、依存する jar ファイル、 Plugin_descriptor.xml ファイルをフォルダに追加して zip ファイルを保存します。

プラグインのZIPファイルのアップロード

プラグインを目的のホストに導入できるように、プラグインの ZIP ファイルを SnapCenter サーバにアップロードする必要があります。

UIまたはコマンドレットを使用してプラグインをアップロードできます。

- UI : *
- プラグインの ZIP ファイルを * Add * または * Modify Host * ワークフローウィザードの一部としてアップロードします
- [選択] をクリックしてカスタムプラグインをアップロードします。 *
- PowerShell : *
- Upload-SmPluginPackageコマンドレット

例：PS>Upload -SmPluginPackage -AbsolutePath c:\DB2_1.zip

PowerShell コマンドレットの詳細については、SnapCenter のコマンドレットのヘルプを使用するか、コマンドレットのリファレンス情報を参照してください。

["SnapCenter ソフトウェアコマンドレットリファレンスガイド"](#)です。

カスタムプラグインの導入

アップロードしたカスタムプラグインを、* Add * および * Modify Host * ワークフローの一環として、目的のホストに導入できるようになりました。SnapCenter サーバに複数のバージョンのプラグインをアップロードして、特定のホストに導入するバージョンを選択できます。

プラグインをアップロードする方法の詳細については、[を参照してください。"ホストを追加してリモートホストにプラグインパッケージをインストールする"](#)

著作権に関する情報

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。