

選取要使用的 .NET 版本

發行項 • 2022/09/22

本文說明用來選取版本的 .NET 工具、SDK 和執行時間所使用的原則。這些原則在執行使用指定版本的應用程式，以及輕鬆升級開發人員和終端使用者電腦之間提供平衡。這些原則會啟用：

- 簡單且有效率的 .NET 部署，包括安全性和可靠性更新。
- 使用最新的工具和命令，且不受目標執行階段影響。

版本選取發生於下列情況：

- 當您執行 SDK 命令時，[SDK 使用最新安裝的版本](#)。
- 當您繫結組件時，[目標 Framework Moniker 定義建置時間 API](#)。
- 當您執行 .NET 應用程式時，會 [向前復原目標 framework 相依的應用程式](#)。
- 當您發佈獨立應用程式時，[獨立部署包含選取的執行階段](#)。

本文件的其餘部分會說明這四個案例。

SDK 使用最新安裝的版本

SDK 命令包含 `dotnet new` 和 `dotnet run`。 .NET CLI 必須為每個 `dotnet` 命令選擇 SDK 版本。根據預設，它會使用電腦上最新安裝的 SDK，即使：

- 專案的目標是舊版的 .NET 執行時間。
- .NET SDK 的最新版本是預覽版本。

您可以利用最新的 SDK 功能和增強功能，同時以較早的 .NET 執行階段版本為目標。您可以使用相同的 SDK 工具，以不同的 .NET 執行階段版本為目標。

在罕見的情況下，您可能需要使用舊版的 SDK。您可以在 [global.json 檔案](#) 中指定該版本。「使用最新的」原則表示您只會使用 `global.asax` 來指定比最新安裝版本更早的 .net SDK 版本。

`global.json` 可能放在檔案階層中的任何地方。CLI 會從專案目錄向上搜尋，以找到第一個 `global.json`。您可以根據指定的 `global.json` 在檔案系統中的位置，來控制其所套用的專案。 .NET CLI 會從目前的工作目錄向上反覆巡覽路徑，以搜尋 `global.json` 檔案。第一個找到的 `global.json` 檔案指定所使用的版本。如果已安裝該 SDK 版本，則會使用該版本。如果找不到在 `global.json` 中指定的 `sdk`，則 .net CLI 會使用比對 [規則](#) 來選取相容的 `sdk`，如果找不到，則會失敗。

下列範例示範 `global.json` 語法：

JSON

```
{
  "sdk": {
    "version": "5.0.0"
  }
}
```

選取 SDK 版本的過程如下：

1. dotnet 會從目前的工作目錄向上反覆反向巡覽路徑，以搜尋 dotnet 檔案。
2. dotnet 使用第一個找到的 dotnet 中指定的 SDK。
3. 如果找不到 dotnet，dotnet 會使用最新安裝的 SDK。

如需 SDK 版本選取的詳細資訊，請參閱[global.asax 總覽](#)文章中的比對規則和向前復原章節。

目標 Framework Moniker 定義建置時間 API

您可以針對目標 Framework Moniker (TFM) 中定義的 API 建置專案。您會在專案檔中指定目標 Framework。在您的專案檔中設定 TargetFramework 項目，如下列範例所示：

XML

```
<TargetFramework>net5.0</TargetFramework>
```

您可以針對多個 TFM 建置專案。設定多個目標 Framework 對程式庫較常見，但也可透過應用程式完成。您會指定 TargetFrameworks 屬性 (TargetFramework 的複數形式)。目標 Framework 是以分號分隔，如下列範例所示：

XML

```
<TargetFrameworks>net5.0;netcoreapp3.1;net47</TargetFrameworks>
```

指定的 SDK 支援一組固定的架構，限制為其隨附執行階段的目標 Framework。例如，.NET 5 SDK 包含 .NET 5 執行時間，這是目標 framework 的實 net5.0 作為目標。 .Net 5 SDK 支援、netcoreapp2.1、netcoreapp3.0 等，但不 net6.0 支援 netcoreapp2.0 (或更高的)。您要安裝 .NET 6 SDK 以建立 net6.0。

.NET Standard

.NET Standard 是一種將 API 介面設為目標的方式，該介面是由不同的 .NET 執行所共用。從 .NET 5 版本（即 API 標準本身）開始，.NET Standard 沒有關聯性，但有一個案例除外：當您想要以 .NET 和 .NET Framework 為目標時，.NET Standard 會很有用。
.NET 5 會執行所有的 .NET Standard 版本。

如需詳細資訊，請參閱 [.net 5 及 .NET Standard](#)。





與 Framework 相依的應用程式向前復原

當您使用 `dotnet run` 從來源執行應用程式、使用 `dotnet run` 執行應用程式，或使用從 **架構相依可執行檔** 執行應用程式時，可執行檔會是該應用程式的主機。

該主機會選擇電腦上最新安裝的修補程式版本。例如，如果您在專案檔中指定 `net5.0`，且 `5.0.2` 是最新安裝的 .NET 執行階段，則會使用 `5.0.2` 執行階段。

如果找不到可接受的 `5.0.*` 版本，則會使用新的 `5.*` 版本。例如，如果您指定 `net5.0` 並只安裝 `5.1.0`，應用程式會使用 `5.1.0` 執行階段來執行。這種行為稱為「次要版本向前復原」。也不會考慮較低的版本。若未安裝可接受的執行階段，應用程式將不會執行。

如果您將目標設為 `5.0`，則有幾個使用範例會示範行為：

- 指定了  `5.0`。 `5.0.3` 是已安裝的最高修補程式版本。使用 `5.0.3`。
-  指定 `5.0`。未安裝 `5.0.*` 版本。 `3.1.1` 是已安裝的最高執行時間。顯示錯誤訊息。
- 指定了  `5.0`。未安裝 `5.0.*` 版本。 `5.1.0` 是安裝的最高執行階段版本。使用 `5.1.0`。
-  指定 `3.0`。未安裝 `3.x` 版。 `5.0.0` 是安裝的最高執行時間。顯示錯誤訊息。

次要版本向前復原有一個可能會影響終端使用者的副作用。請考慮下列案例：

- 應用程式會指定需要 `5.0`。
- 但是，在執行時，不會安裝 `5.0.*` 版，但 `5.1.0` 是。將會使用版本 `5.1.0`。
- 之後，使用者會安裝 `5.0.3`，並再次執行應用程式，現在將會使用 `5.0.3`。

`5.0.3` 和 `5.1.0` 可能會有不同的行為，特別是針對序列化二進位資料等案例。

控制項向前復原行為

您可以使用四種不同的方式來設定應用程式的向前復原行為：

- 藉由設定 `<RollForward>` 屬性來 Project 層級設定：

XML

```
<PropertyGroup>
  <RollForward>LatestMinor</RollForward>
</PropertyGroup>
```

2. *.runtimeconfig.json 檔案。

當您編譯應用程式時，就會產生這個檔案。 <RollForward> 如果已在專案中設定屬性，則會在檔案中 *.runtimeconfig.json 重制為 rollForward 設定。使用者可以編輯此檔案來變更您的應用程式行為。

JSON

```
{
  "runtimeOptions": {
    "tfm": "net5.0",
    "rollForward": "LatestMinor",
    "framework": {
      "name": "Microsoft.NETCore.App",
      "version": "5.0.0"
    }
  }
}
```

3. dotnet 命令的 --roll-forward <value> 屬性。

當您執行應用程式時，您可以透過命令列來控制向前復原行為：

.NET CLI

```
dotnet run --roll-forward LatestMinor
dotnet myapp.dll --roll-forward LatestMinor
myapp.exe --roll-forward LatestMinor
```

4. DOTNET_ROLL_FORWARD 環境變數。

優先順序

當您的應用程式執行時，向前復原行為是依下列順序設定，優先順序較低的專案會優先于較低的編號專案：

1. *.runtimeconfig.json 首先會評估設定檔。
2. 接下來， DOTNET_ROLL_FORWARD 會考慮環境變數，覆寫先前的檢查。
3. 最後，任何 --roll-forward 傳遞至執行中應用程式的參數都會覆寫其他所有專案。

值

不過，您可以設定向前復原設定，請使用下列其中一個值來設定行為：

值	描述
Minor	如果未指定，則為 預設值 。 如果遺漏要求的次要版本，則向前復原到最低的次要版本。 如果要求的次要版本存在，則會使用該 LatestPatch 原則。
Major	如果遺漏要求的主要版本，則向前復原到下一個可用的較高主要版本，以及最低次要版本。 如果要求的主要版本存在，則會使用該 Minor 原則。
LatestPatch	向前復原到最高的修補程式版本。 此值會停用次要版本向前復原。
LatestMinor	向前復原到最高的次要版本，即使有要求的次要版本存在也一樣。
LatestMajor	向前復原到最高的主要和最高次要版本，即使有要求的主要存在。
Disable	不要向前復原，只系結至指定的版本。 此原則不建議用於一般用途，因為它會停用向前復原到最新修補程式的能力。 只有測試時才建議使用這個值。

獨立部署包含選取的執行階段

您可以將應用程式發佈為**獨立散發**。 這種方法會將 .NET 執行時間和程式庫與您的應用程式結合在一起。 獨立部署不會相依於執行階段環境。 執行階段版本選取發生於發佈時，而不是執行時。

發行時所發生的 **還原** 事件會選取指定執行時間系列的最新修補程式版本。 例如，如果是 .NET 5 執行時間系列中的最新修補程式版本， dotnet publish 則會選取 .net 5.0.3。 目標 Framework (包括最新安裝的安全性修補程式) 會封裝於應用程式。

如果未滿足針對應用程式指定的最小版本，就會發生錯誤。 dotnet publish 會繫結至最新的執行階段修補程式版本 (指定的主要.次要版本系列內)。 dotnet publish 不支援 dotnet run 的向前復原語意。 如需修補程式和獨立式部署的詳細資訊，請參閱部署 .NET 應用程式中的 **執行時間修補程式選項**。

獨立部署可能需要特定修補程式版本。 您可以覆寫專案檔中的最低執行階段修補程式版本 (改為較高或較低版本)，如下列範例所示：

XML
<pre><PropertyGroup> <RuntimeFrameworkVersion>5.0.7</RuntimeFrameworkVersion> </PropertyGroup></pre>

`RuntimeFrameworkVersion` 元素會覆寫預設的版本原則。針對獨立部署，

`RuntimeFrameworkVersion` 會指定「確切」`RuntimeFrameworkVersion`的執行階段架構版本。針對架構相依應用程式，`RuntimeFrameworkVersion` 會指定所需的「最低」

`RuntimeFrameworkVersion` 執行階段架構版本。

另請參閱

- [下載並安裝 .net](#)。
- [如何移除 .Net 執行時間和 SDK](#)。