

自動インストールの使い方 (インストールマニフェスト)



この資料の主な目標

- この資料では、製品インストール後の構成設定、ソースコードのインポートとコンパイル、初期実行などを自動的に処理できる「インストールマニフェスト」の記述方法を習得するため、以下の内容をご説明します。
 - インストールマニフェストとは
 - クラスの作り方
 - <Manifest>タグ
 - <Manifest>内で使用できる変数
 - 自動実行のための引数／コマンドプロパティ

製品インストールの自動実行については、Windowsの例のみを記載しています。
他OSの実行例については、ドキュメント([IRIS](#)／[Caché](#))をご参照ください



インストールマニフェストとは

- インストールマニフェストとは、製品インストール後の構成定義、セキュリティ設定、ソースコードのインポートや初期実行などを自動化するために記述する定義です。
 - %SYSネームスペースにクラス定義を1つ用意し、XDataブロックに専用タグを使用して定義します。
 - マニフェストをインストール後自動的に、または手動で呼び出すことで 手作業で行っていた設定やソースコードのインポート、初期実行などを、自動化することができます。
- インストールマニフェストは、%Installerパッケージで構成されています。
 - クラスコンパイル時 <Manifest> タグの定義内容を基に、%Installerパッケージのクラスを使用してロジックを生成します。



例: <Manifest>タグ

```
4  XData InstallManifest [ XMLNamespace = INSTALLER ]
5  {
6    <Manifest>
7      <!-- Manifestで使用する変数の設定 ${変数名}で操作可 インストール時に同名でパラメータとして渡すことも可-->
8      <Var Name="Namespace" Value="Training"/>
9      <!-- インストールファイルがあるディレクトリ/App をアプリケーションファイルがあるディレクトリとして設定-->
10     <If Condition='#{$length("${SourceDir}")}=0' >
11       <!-- インストール後に実行している場合のSourceDirの指定 -->
12       <Var Name="SourceDir" Value="c:/kit/Source"/>
13     </If>
14     <Var Name="AppSourceDir" Value="${SourceDir}/App"/>
15     <!--ライセンスキーの反映 -->
16     <CopyFile Src="${SourceDir}/cache.key" Target="${MGRDIR}/cache.key"/>
17     <Invoke Class="%SYSTEM.License" Method="Upgrade"/>
18     <!-- ネームスペース、データベース Training を作成 -->
19     <Namespace Name="${Namespace}" Create="yes" Code="${Namespace}" Data="${Namespace}">
20       <!--ネームスペース名と同じ名称のデータベースを作成する-->
21       <If Condition='##class(%File).Exists("${MGRDIR}/Training/CACHE.DAT")=0'>
22         <Configuration>
23           <Database Name="${Namespace}" Create="yes" Dir="${MGRDIR}/Training" Resource="%DB_%DEFAULT"/>
24         </Configuration>
25       </If>
26       <!-- /csp/ネームスペース名 の作成
27          パスワード認証でログインページを設定、自動コンパイルOff、CSPインポート後ソースコード消去
28          EventClassはWebAddOnライセンス使用時に必要 -->
29       <Var Name="CSPTTrainingDir" Value='${CSPDIR}#{ZCVT("${Namespace}","L")}'/>
30       <Var Name="CSPTTrainingPath" Value='/csp/#{ZCVT("${Namespace}","L")}'/>
31       <CSPApplication Url="${CSPTTrainingPath}"
32         Directory="${CSPTTrainingDir}"
33         Recurse="true"
34         AuthenticationMethods="32"
35         AutoCompile="0"
36         IsNamespaceDefault="true"
37         LoginClass='${CSPTTrainingPath}/Employee/emplogin1.csp'
38         EventClass="Training.WebAddOnSessionEvents"
39       />
40       <!-- ソースコードのインポート -->
41       <Import File="${AppSourceDir}/Source.xml" Flags="ck"/>
42       <Import File="${AppSourceDir}/Source-CSP.xml" Flags="ck"/>
43       <!-- WebAdon用イベントクラスのインポート-->
44       <Import File="${AppSourceDir}/Training.WebAddOnSessionEvents.xml" Flags="ck"/>
45       <!-- CSPファイルの削除 -->
46       <Invoke Class="%File" Method="Delete">
```

XDataブロックに用意する<Manifest>

XData **MyInstall** [XMLNamespace = **INSTALLER**]

```
<Manifest>  
  <Namespace>  
    <Configuration>  
      <Database />  
      <GlobalMapping />  
      <ClassMapping />  
    </Configuration>  
    <CSPApplication />  
    <Import />  
    <Invoke />  
  </Namespace>  
  <CopyFile />  
</Manifest>
```

マニフェストの名称で実行時の指定名として利用します。

INSTALLER
の指定で
<Manifest>
タグ以下の入力
候補が出せます。

タグ詳細やこの他のタグ
についてはドキュメントを
ご参照ください。



インストール自動化用クラスの作成方法

- %SYSネームスペースに Z から始まる名前のクラスを作成します。
 - %SYS上で起動するユーティリティを使用しない場合は任意のネームスペースで作成します。

The image shows a screenshot of the ZInstaller Wizard in a development environment. The wizard is titled "Server Template %Installer Wizard" and contains the following fields:

- パッケージ名: * ZInstaller (必須です。新しいクラスのバ)
- クラス名: * MyInstall (必須です。新しいクラスの名前)
- マニフェストの名前: * MyManifest (必須です。%Installer マニフェ)

The wizard's output is shown in the code editor window "ZInstaller.MyInstall.cls *". The code is as follows:

```
Include %occInclude

/// %Installer Manifest ZInstaller.MyInstall
Class ZInstaller.MyInstall
{

  /// マニフェスト定義.
  [Data:MyManifest [ XMLNamespace = INSTALLER ]
  <Manifest>
  <Namespace>
  <Configuration>
  <Database>
    <!-- Your Manifest code here -->
  </Database>
  </Configuration>
  </Namespace>
</Manifest>
}

/// これは XGL により生成されたメソッド・ジェネレーターです。
[ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3, pInstaller As %I
{
  #: XGL ドキュメントでこのメソッドのコードを生成する。
  Quit ##class(%Installer.Manifest).%Generate(%compiledclass,
    | %code, "MyManifest")
}
```

A red speech bubble on the right side of the code editor contains the following text:

ウィザードから作成すると必要な定義一式を作成します。ウィザードを使用しない方法は本資料末尾をご覧ください。

<Manifest>タグで使用できる式 \$(変数名)

- **\$(変数名)**は、マニフェスト内で使用できる事前定義変数やユーザ定義の変数の指定に利用できます。
 - マニフェスト内で変数を定義することもできます。

【マニフェスト内での変数定義】

```
<Var Name="Namespace"  
    Value="Training"/>  
<Namespace  
    Name="$(Namespace)"  
    Create="yes"  
    Code="$(Namespace)"  
    Data="$(Namespace)">
```

- マニフェストに変数リストを渡す方法もあります(後述します)。

事前定義の変数名	説明
SourceDir	(インストーラ実行中にのみ使用可能) インストール (setup_xxx.exe または irisinstall または cinstall) の実行元となるディレクトリ。
ISCUUpgrade	(インストーラ実行中にのみ使用可能) 新規インストールまたはアップグレードであることを示します。この変数は 0 (新規インストール) または 1 (アップグレード) のいずれかになります。
CFGDIR	"INSTALLDIR" を参照してください。
CFGNAME	インスタンス名。
CPUCOUNT	オペレーティング・システムの CPU の数。
CSPDIR	CSP ディレクトリ。
HOSTNAME	ホスト・サーバの名前。
HTTPPORT	Web サーバのポート。
INSTALLDIR	インストール先ディレクトリ。
MGRDIR	管理者用 (mgr) ディレクトリ。
PLATFORM	オペレーティング・システム。
PORT	スーパーサーバ・ポート。
PROCESSOR	プロセッサ・チップ。
VERSION	バージョン番号。

<Manifest>タグで使用できる式 \$(変数名) つづき

- マニフェスト実行用setup()メソッドの第1引数に、マニフェスト内で使用する変数を配列変数で渡すことができます。

【1】 setup()実行前に以下の変数を作成

```
set val("Namespace")="Training"  
set val("SourceDir")="C:¥kit¥source"
```

【2】 setup()の実行

```
set st=##class(ZMyClass.Installer).setup(.val,LogLevel)
```

【3】 マニフェストでの変数指定例

```
<Var Name="AppSourceDir" Value="{SourceDir}/App"/>  
<Namespace Name="{Namespace}" Create="yes"  
Code="{Namespace}" Data="{Namespace}">
```

```
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,pInstaller As %Installer.Installer,  
pLogger As %Installer.AbstractLogger) As %Status [ CodeMode = objectgenerator, Internal ]  
{  
Quit ##class(%Installer.Manifest).%Generate(%compiledclass,%code,"MyInstall")  
}
```

<Manifest>タグで使用できる式

#{評価式} つづき

- マニフェスト内で ObjectScript の式を指定できます。
- #{ } の中に ObjectScript で記述できる式を指定できます。
- スクリプトで指定する二重引用符の数にご注意ください。

【例】変数UnitTestRootに設定されたディレクトリが存在していない時に真となる条件式

```
<If Condition='# { ##class(%File).DirectoryExists("${UnitTestRoot}") } =0' >  
  <Invoke Class="%File" Method="CreateDirectory"  
    Return="RetVal" CheckStatus="true">  
    <Arg Value="${UnitTestRoot}"/>  
  </Invoke>  
</If>
```



マニフェストの実行

手動実行

1. 必要であれば、マニフェスト内で使用する変数を設定します。

```
set val("Namespace")="Training"  
set val("SourceDir")="c:¥kit¥source"
```

2. setup()メソッドを実行します。

```
set st=##class(ZMyClass.MyInstaller).setup(.val)
```

- setup()メソッドの第1引数に渡す変数が多い場合は、マニフェストクラスに実行用クラスメソッドを用意すると便利です。
- ログ出力を指定する場合は、第2、第4引数を使用します。
 - 第2引数 : ログレベルの指定 (1~3)
 - 第4引数 : マニフェスト実行結果をファイル出力したい場合、ログファイル用オブジェクト(%Installer.FileLogger のインスタンス)を指定します。

```
set pLogger = ##class(%Installer.FileLogger).%New(1,"ログファイルのフルパス")  
set st=##class(ZMyClass.MyInstaller).setup(.val,3,,pLogger)
```



マニフェストの実行

自動実行(マニフェスト用プロパティ)

- コマンド行パラメータでインストールマニフェストに関係したプロパティを指定できます。

※ドキュメントもご参照ください

- **INSTALLERMANIFESTPARAMS**

インストールマニフェストクラスのsetup()メソッドの第1引数に渡される変数と値のペア(name=value)を指定します。

- **INSTALLERMANIFEST**

インストールマニフェストクラスのエクスポートファイル名(XML)を指定します。

- **INSTALLERMANIFESTLOGLEVEL**

ログレベルを指定します(setup()メソッドの第2引数の指定する数値)。

- (※2016.1以降) **INSTALLERMANIFESTLOGFILE**

マニフェスト実行時のログメッセージを出力するファイル名をフルパスで指定します(ログのファイル出力について注意事項を後述しています)。

```
setup.exe /instance TEST /qn UNICODE=1 SUPERSERVERPORT=56774  
WEBSERVERPORT=57774 INSTALLDIR="c:¥intersystems¥test"  
/l c:¥kit¥Source¥install.log
```

```
INSTALLERMANIFESTPARAMS="SourceDir=c:¥kit¥Source"
```

```
INSTALLERMANIFESTLOGFILE="c:¥kit¥source¥installer_log.txt"
```

```
INSTALLERMANIFESTLOGLEVEL="3"
```

```
INSTALLERMANIFEST="C:¥kit¥Source¥ZMyClass.Installer.xml"
```

Windowsでの例

マニフェストの実行 自動実行(その他プロパティ)

- 自動インストールのコマンド行プロパティは以下の通りです。

パラメータ/プロパティ	指定内容
/instance	インスタンス名を指定 (※2012.2以降で利用できます) [メモ] 2012.1以前では、INSTANCENAME=XXX を使用します。このプロパティを使用すると2つ目以降のインストールで、インスタンスを選択するダイアログが表示されます。
/qn	サイレントインストール用(インストールの進捗画面が表示されません)
/l	ログファイルの指定
SUPERSEVERPORT	スーパーサーバポート
WEBSERVERPORT	Webサーバポート
INSTALLDIR	インストールディレクトリ
UNICODE	UNICODEモードの指定(日本語環境の場合必須)

※ドキュメントもご参照ください

```
setup.exe /instance TEST /qn UNICODE=1 SUPERSEVERPORT=56774  
WEBSERVERPORT=57774 INSTALLDIR="c:¥intersystems¥test"  
/l c:¥kit¥Source¥install.log  
INSTALLERMANIFESTPARAMS="SourceDir=c:¥kit¥Source"  
INSTALLERMANIFESTLOGFILE="c:¥kit¥source¥installer_log.txt"  
INSTALLERMANIFESTLOGLEVEL="3"  
INSTALLERMANIFEST="C:¥kit¥Source¥ZMyClass.Installer.xml"
```

Windowsでの例

INSTALLERMANIFESTLOGFILE または setup()メソッド第4引数指定の《注意事項》

- 製品インストール時のコマンドプロパティの指定で
INSTALLERMANIFESTLOGFILE (Unix/Linuxの場合は
ISC_INSTALLER_LOGFILE) や、setup()メソッドの第4引数に
%Installer.FileLoggerクラスのインスタンスを指定すると、マニフェスト
実行時のメッセージを指定ファイルに出力できます。
- マニフェストは%SYSネームスペースで実行されるため、ログファイル用
インスタンス(%Installer.FileLogger)は、%SYSネームスペース
で出力するログのみ有効となります。
 - 特定のネームスペースで生成したインスタンスは、別ネームスペースへ引き継
げないため、マニフェスト実行中に%SYS以外のネームスペースでのログ出
力がある場合、インスタンスが存在しないため UNDEFINEDエラー が発生し
ます。
- マニフェスト実行時にログ出力を選択していて、%SYSネームスペース以
外での処理が含まれる場合は、対象となるネームスペースの中だけで閉
じたログ出力にするなど工夫が必要です。



ご参考:

自動インストールの事前準備 (Windows)

- 以下のパッケージをダウンロードし事前にインストールします。
 - 2015.2以降

Visual Studio 2013 の Visual C++ 再頒布可能パッケージ

- 2015.1まで

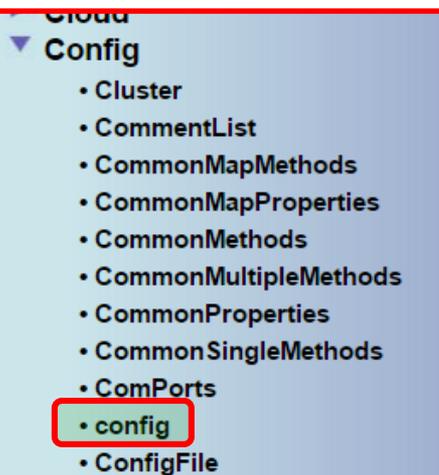
Visual Studio 2008 の Visual C++ 再頒布可能パッケージ

- 64bitシステムにインストールする場合は、
vcredist_x86.exe ファイルと vcredist_x64.exe ファイル
の両方をインストールする必要があります。



その他タグ: <SystemSettings> Configパッケージ以下クラスのプロパティ名を指定

```
<!-- データベースキャッシュサイズの設定-->  
<SystemSetting Name="Config.config.globals8kb" Value="2048"/>  
<!-- ルーチンキャッシュサイズの設定-->  
<SystemSetting Name="Config.config.routines" Value="256"/>  
<!-- 一般メモリヒープサイズの設定-->  
<SystemSetting Name="Config.config.gmheap" Value="262144"/>  
<!-- ロックサイズの設定-->  
<SystemSetting Name="Config.config.locksiz" Value="16777216"/>  
<!-- プロセスあたりの最大メモリサイズの設定-->  
<SystemSetting Name="Config.config.bbsiz" Value="393216"/>  
<!-- プライマリジャーナルディレクトリの設定-->  
<SystemSetting Name="Config.Journal.CurrentDirectory" Value="c:¥jrn¥journals"/>  
<!-- プライマリジャーナルディレクトリの設定-->  
<SystemSetting Name="Config.Journal.AlternateDirectory" Value="c:¥jrn¥journals"/>  
<!-- シャットダウン時SYSLOGをコンソールログに出力する設定-->  
<SystemSetting Name="Config.Miscellaneous.ShutDownLogErrors" Value="1"/>  
<!-- SYSLOGのエントリ数の設定-->  
<SystemSetting Name="Config.config.errlog" Value="1000"/>
```



Summary			
プロパティ			
CPFFile	Comments	ConsoleFile	Flags
LibPath	LineRecallBuffer	LineRecallEntries	LockSharedMemory
LockTextSegment	MaxServerConn	MaxServers	Name
VMSConsoleTerminal	ZFSize	ZFString	bbsiz
console	errlog	globals	globals16kb
globals32kb	globals4kb	globals64kb	globals8kb
gmheap	history	ijcbuff	ijcnum
locksiz	memlock	netjob	nlstab
overview	pjkdir	routines	udevtabsiz
userresidentmem	vectors	wjkdir	

<SystemSettings>を INSTALLERMANIFESTPARAMSで設定する

- <SystemSettings>の一部の設定は、**INSTALLERMANIFESTPARAMS**で指定でき、マニフェスト実行前に設定を有効化できます。
 - 設定できる項目は以下の通りです。
bbsiz、globals4kb、globals8kb、globals16kb、globals32kb、globals64kb、gmheap、LibPath、locksiz、MaxServerConn、Path、routines、ZFSize、ZFString

```
INSTALLERMANIFESTPARAMS="globals8kb=2048,routines=256,gmheap=262144,  
locksiz=16777216,bbsiz=393216
```

```
<!-- データベースキャッシュサイズの設定-->  
<SystemSetting Name="Config.config.globals8kb" Value="2048"/>  
<!-- ルーチンキャッシュサイズの設定-->  
<SystemSetting Name="Config.config.routines" Value="256"/>  
<!-- 一般メモリヒープサイズの設定-->  
<SystemSetting Name="Config.config.gmheap" Value="262144"/>  
<!-- ロックサイズの設定-->  
<SystemSetting Name="Config.config.locksiz" Value="16777216"/>  
<!-- プロセスあたりの最大メモリサイズの設定-->  
<SystemSetting Name="Config.config.bbsiz" Value="393216"/>  
<!-- プライマリジャーナルディレクトリの設定-->  
<SystemSetting Name="Config.Journal.CurrentDirectory" Value="c:¥jrn¥journals"/>  
<!-- プライマリジャーナルディレクトリの設定-->  
<SystemSetting Name="Config.Journal.AlternateDirectory" Value="c:¥jrn¥journals"/>  
<!-- シャットダウン時SYSLOGをコンソールログに出力する設定-->  
<SystemSetting Name="Config.Miscellaneous.ShutDownLogErrors" Value="1"/>  
<!-- SYSLOGのエントリ数の設定-->  
<SystemSetting Name="Config.config.errlog" Value="1000"/>
```

この項目を
INSTALLERMANIFESTPARAMS
で指定できます



セキュリティ設定

- セキュリティ設定はテスト環境などで作成した定義をXMLファイルでエクスポートしマニフェストの中でインポートします。
- インポートは、**Security**パッケージの設定対象クラス(ユーザ定義なら**User**クラス)の**Import()**メソッドを使用します。

```
<Invoke Class="Security.Users"  
  Method="Import" CheckStatus="true">  
  <Arg Value="{SourceDir}/Users.xml" />  
</Invoke>
```

- セキュリティ設定のエクスポートは ^SECURITYルーチンの各メニューから行うか、Securityパッケージ以下設定対象クラスのExport()メソッドから実行します(%SYSで実行)。

```
%SYS>set st=##class(Security.Users).Export(  
  "c:¥kit¥Source¥Users.xml")
```

```
%SYS>do ^SECURITY
```

```
1) User setup  
2) Role setup  
3) Service setup  
4) Resource setup  
5) Application setup  
6) Auditing setup  
7) Domain setup  
8) SSL configuration setup  
9) Mobile phone service provider setup  
10) OpenAM Identity Services setup  
11) Encryption key setup  
12) System parameter setup  
13) X509 User setup  
14) Exit
```

```
Option? 1
```

```
1) Create user  
2) Edit user  
3) List users  
4) Detailed list users  
5) Delete user  
6) Export users  
7) Import users  
8) Exit
```

```
Option? █
```

セキュリティ: <User>タグのパスワード

- <User>タグを利用したユーザ設定では、**PasswordVar**パラメータを使用してパスワードを設定します。
 - 【注意】パスワードを直接設定する項目ではありません。

```
<!-- Userタグでのユーザ情報登録-->  
<If Condition='#{$length("${test1PWD}")}=0'>  
  <Var Name="test1PWD" Value="test1"/>  
</If>  
<User Username="test1"  
  PasswordVar="test1PWD"  
  Enabled="true" Roles="%Developer"/>
```

PasswordVarで設定した**変数の値**は、例のように<Var>で定義しても、インストール時のコマンドパラメータ **INSTALLERMANIFESTPARMS** で設定するなど様々な方法が検討できます。

```
INSTALLERMANIFESTPARAMS="test1PWD=test1"
```



ご参考

- 古いバージョンでのインストールマニフェストクラスの作成方法



ご参考:

%Installerマニフェスト用ウィザードがないバージョンでの作成方法

- %SYSネームスペースに **Z** から始まる名前のクラスを作成します。
 - 何も継承しないクラスを作成します。
 - インクルードルーチン: %occInclude をインクルードします。

何も継承しないクラスの作成には、新規クラスウィザードのクラスタイプ選択画面でExtendsを選択し、スーパークラスに何も指定せず「完了」ボタンを押下します。

Class	(Summary)
名前	値
ClassType	
ClientDataType	VARCHAR
ClientName	

ZMyClass.Installer.IncludeCode

インクルードファイルを選択してください。順序を右側のリストで指定できます:

- NapiOBIND
- NapiXSQL
- Nassdbg
- Nassert
- Ncallout
- NcmtInclude
- NcspBuild
- NcspGatewayRegistry
- NcspInclude
- NcspMgr
- NDeepSee
- NDeepSee.CA

IncludeCode %occInclude

ご参考:

%Installerマニフェスト用ウィザードがないバージョンでの作成方法 つづき

- インストールの詳細を定義するXDataブロックを記述します。

The screenshot shows the Visual Studio IDE with the context menu open for the 'XData' element in the 'ZMyClass.MyInstaller.cs' file. The '追加(A)' (Add) option is highlighted. To the right, the '新規XDataウィザード' (New XData Wizard) dialog is open, showing the 'この新規 XData の名前を入力してください:' (Enter name for this new XData) field with 'MyInstall' entered. A red arrow points from the 'XData' menu item to the wizard dialog.

The screenshot shows the Visual Studio IDE with the 'ZMyClass.MyInstaller.cs' file open. The code shows the following XData block:

```
6 XData MyInstall [ XMLNamespace = INSTALLER ]  
7 {  
8   <Manifest>  
9     < /Manifest>  
10 }  
11 }  
12 }  
13 }
```

A green callout box points to the 'XMLNamespace = INSTALLER' attribute, containing the text: **XMLNameSpaceにINSTALLERを指定することで、<Manifest>タグの入力候補が出せるようになります。**

To the right, the 'XData Properties' window is open, showing the 'MyInstall' XData block. The 'XMLNamespace' property is highlighted with a green box and set to 'INSTALLER'.

名前	値
Name	MyInstall
Data	False
SchemaSpec	False
text/xml	text/xml
XMLNamespace	INSTALLER

ご参考:

%Installerマニフェスト用ウィザードがないバージョンでの作成方法 つづき

- `setup()`メソッドをサンプルからコピーします。
 - %Installer.Manifestクラスの%Generate()メソッドの第3引数には、XDataブロック名を指定します。

```
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,  
  pInstaller As %Installer.Installer,  
  pLogger As %Installer.AbstractLogger)  
  As %Status [ CodeMode = objectgenerator, Internal ]  
{  
  Quit ##class(%Installer.Manifest).%Generate(%compiledclass,%code,  
                                               "MyInstall")  
}
```

インストールマニフェスト
を実行するために
**setup()メソッドを全て
小文字で定義** します。
製品インストール時に
インストールマニフェスト
を指定すると、インストール
後に自動でsetup()メ
ソッドが呼び出されます。

```
1 Include %occInclude  
2  
3 Class ZMyClass.MyInstaller  
4 {  
5  
6 XData MyInstall [ XMLNamespace = INSTALLER ]  
7 {  
8   <Manifest>  
9   </Manifest>  
10 }  
11 ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3, pInstaller As %Installer.Inst  
12  
13   #; Let XGL document generate code for this method.  
14   Quit ##class(%Installer.Manifest).%Generate(%compiledclass,%code, "MyInstall")  
15 }  
16 }  
17 }
```

<記載例>