.. raw:: html

The DNN Manifest Schema

Overview

The **DNN manifest** is an XML file (e.g., MyDNNExtension.dnn) that specifies how each file in the extension package must be processed during installation.

Only the files declared in the manifest would be installed. This includes files inside a zip file specified in `component type="ResourceFile"`. Nonexistent files mentioned in the manifest will cause an error message.

The manifest file extension must be `.dnn`. You can add the DNN version at the end; e.g., `MyDNNExtension.dnn8`.

Save the manifest file in the base folder of your package and include it when zipping your package files.

Schema

```
<dotnetnuke type="Package" version="8.0">
    <packages>
        <package name="MyCompany.SampleModule" type="Module" version="1.0.0">
            <friendlyName>My Sample Module</friendlyName>
            <description>My Sample Module is a demonstration module.</description>
            <iconFile>MyIcon.png</iconFile>
            <owner>
                <name>MyCompany or MyName</name>
                <organization>MyCompany Corporation</organization>
                <url>www.example.com</url>
                <email>support@example.com</email>
            </owner>
            <license src="MyLicense.txt" />
            <releaseNotes src="MyReleaseNotes.txt" />
            <azureCompatible>true</azureCompatible>
            <dependencies>
                <dependency type="coreVersion">08.00.00</dependency>
                ...
            </dependencies>
            <components>
                <component type="Module">
                    ...
                </component>
                ...
            </components>
        </package>
    </packages>
</dotnetnuke>
```

package

```
<package name="MyCompany.MySampleModule" type="Module" version="1.0.0">
...
</package>
```

- **name** must be unique. To ensure your package's uniqueness, add your company as the prefix.
- **type** can be one of the following:
  - **Authentication System**
  - **Container**
  - **Dashboard Control**
  - Language Pack: (**CoreLanguagePack**, **ExtensionLanguagePack**)
  - **Library**
  - **Module**
  - **Provider**
  - **Skin**
  - **Skin Object**
  - other custom extension types
- **version** holds the version of your extension.

Each package represents a DNN extension. You can install multiple extensions using a single DNN manifest by creating a **package** section for each extension inside the **packages** tag.

However, when packaging a theme, containers must be defined under a separate section, even though the containers are installed as part of the theme.

Extensions are installed in the order they appear in the manifest.

Only the information about the *first* package is displayed during installation. This includes the package name, description, owner, license, and release notes.

friendlyName and description

```
<friendlyName>My Sample Module</friendlyName>
<description>My Sample Module is a demonstration module.</description>
```

The **friendlyName** can contain spaces and up to 250 characters. The **description** can hold up to 2000 characters.

iconFile

```
<iconFile>MyIcon.png</iconFile>
```

Optional. The icon is displayed in the DNN Control Panel's dropdown list and in the Extensions page. The **.png** format is preferred.

Note: The **iconFile** section must be between the **description** and the **owner** sections.

owner

```
<owner>
    <name>MyCompany or MyName</name>
    <organization>MyCompany Inc.</organization>
    <url>www.example.com</url>
    <email>support@example.com</email>
</owner>
```

Optional, but encouraged. Information about the owner or creator of the extension.

license and releaseNotes

```
<license src="MyLicense.txt" />
<releaseNotes src="MyReleaseNotes.txt" />
```

Optional, but encouraged. These text files are displayed during the installation. The user is prompted to accept or decline the license. The release notes is displayed during the installation. The actual text can also be embedded within the tag without the **src** attribute.

azureCompatible

```
<azureCompatible>true</azureCompatible>
```

Optional. Default is **false**. Set to **true** if the module is compatible with SQL Azure.

dependencies

```
<dependencies>
    <dependency type="coreVersion">08.00.00</dependency>
    ...
</dependencies>
```

Dependencies can be any of these types (case-insensitive):

- **coreVersion**. Minimum DNN version required by the extensions being installed. Example:

  ```
  <dependency type="coreVersion">08.00.00</dependency>
  ```

- **dependency list item**. Example:

  ```
  <dependency type="dependency list item">???</dependency>
  ```

- **managedPackage**. Example:

  ```
  <dependency type="managedPackage">???</dependency>
  ```

- **package**. A package required by the extensions being installed. It must already listed in the core Packages table. Example:

  ```
  <dependency type="package">AnotherPackageRequiredByThisComponent</dependency>
  ```

- **permission**. Example:

  ```
  <dependency type="permission">???</dependency> <!-- TODO for Joe for DNN permissions and to find out custom permissions.-->
  ```

- **type**. Example:

  ```
  <dependency type="type">...</dependency> <!-- Name of namespace.class in .NET, in a DNN library, or a third-party library. -->
  ```

components

```
<components>
    <component type="..." />
    <component type="..." />
    ...
</components>
```

A component type can be one of the following:

- **Assembly**. Assemblies to be installed in the main `\bin` folder of the installation.

```
<component type="Assembly">
    <assemblies>
        <assembly>
            <path /> <!-- Path of the assembly to install. Relative to the \bin folder of the DNN installation. -->
            <name /> <!-- Name of the assembly to install. -->
            <version /> <!-- Version of the assembly to install. -->
        </assembly>
        <assembly action="Unregister">
            <path /> <!-- Path of the assembly to remove. Relative to the \bin folder of the DNN installation. -->
            <name /> <!-- Name of the assembly to remove. -->
            <version /> <!-- Version of the assembly to remove. -->
        </assembly>
        ...
    </assemblies>
</component>
```

- **AuthenticationSystem**. Authentication providers used by the module, such as **OpenId**, **LiveId**, and **ActiveDirectory**.

```
<component type="AuthenticationSystem">
    <authenticationService>
        <type>OpenID</type>
        <settingsControlSrc />
        <loginControlSrc />
        <logoffControlSrc />
    </authenticationService>
    <authenticationService />
    ...
</component>
```

- **Cleanup**. List of files that must be deleted when the module is uninstalled. Also see:
    - Component type **Script** for data provider scripts that must be uninstalled, and
    - Component type **Config** to update configuration files during uninstall.

```
<!-- You can list the files individually in the manifest.  -->
<component type="Cleanup" version="07.40.00">
    <files>
        <file>
            <path>bin</path>
            <name>MyFile.dll</name>
        </file>
        <file />
        ...
    </files>
</component>

<!-- You can also list the files with their paths in a text file instead.  -->
<component type="Cleanup" version="07.40.00" fileName="ListOfFilesToDelete.txt" />
```

- **Config**. Changes to do on the specified config file..

```
<component type="Config">
    <config>
        <configFile>web.config</configFile> <!-- Name of config file, including its path relative to the root of the DNN installation. -->
        <install>
            <configuration>
                <nodes>
                    <!-- For information on the "node" attributes and child nodes, see http://www.dnnsoftware.com/wiki/manifest-xml-merge. -->
                    <node path="/configuration/system.webServer/handlers" action="update" key="path" collision="overwrite">
                        ...
                    </node>
                    <node />
                    ...
                </nodes>
            </configuration>
        </install>
        <uninstall>
            <configuration>
                <nodes />
            </configuration>
        </uninstall>
    </config>
    <config />
    ...
</component>
```

- **Container**. Containers to be installed.

```
<component type="Container">
    <containerFiles>
        <basePath /> <!-- Target base folder for the component installation. Relative to the root of the DNN installation. -->
        <containerName />
        <containerFile>
            <path /> <!-- Target file folder. Relative to basePath. -->
            <name />
        </containerFile>
        <containerFile />
        ...
    </containerFiles>
</component>
```

- **DashboardControl**.

```
<component type="DashboardControl">
    <dashboardControl>
        <key />
        <src />
        <localResources />
        <controllerClass />
        <isEnabled />
        <viewOrder />
    </dashboardControl>
    <dashboardControl />
    ...
</component>
```

- **File**. Files to be installed. Only files with allowed file extensions are installed. To view or modify the list of file extensions, go to **Host** > **Host Settings** > **Other Settings** > **Allowable File Extensions** in your DNN installation.

```
<component type="File">
    <files>
        <basePath /> <!-- Target base folder for the component installation. Relative to the root of the DNN installation. -->
        <file>
            <path /> <!-- Target file folder. Relative to basePath. -->
            <name />
            <sourceFileName /> <!-- Must include the path (relative to basePath) and the filename. -->
        </file>
        <file />
        ...
    </files>
</component>
```

- **CoreLanguage**. Used for packaging the language pack to be used by the DNN Platform installation. For the list of supported language codes, see the .NET CultureInfo class.

```
<component type="CoreLanguage">
    <languageFiles>
        <code />
        <displayName />
        <fallback /> <!-- Code for the alternative language. Used if a resource is not found in the current language pack. -->
        <languageFile>
            <path /> <!-- Target file folder. Relative to the root of the DNN installation. -->
            <name />
        </languageFile>
        <languageFile />
        ...
    </languageFiles>
</component>
```

- **ExtensionLanguage**. Used for packaging the language pack used by the extension. For the list of supported language codes, see the .NET CultureInfo class.

```
<component type="ExtensionLanguage">
    <languageFiles>
        <code />
        <package /> <!-- Name of another package that contains the extension that this language pack is intended for. -->
        <basePath /> <!-- Target base folder for the component installation. Relative to the root of the DNN installation. -->
        <languageFile>
            <path /> <!-- Target file folder. Relative to basePath. -->
            <name />
        </languageFile>
        <languageFile />
        ...
    </languageFiles>
</component>
```

- **Module**. Only one component with type="Module" is allowed within a **package** section. To install a set of modules as a unit, create one **package** section per module in the same manifest.

```
<component type="Module">
    <desktopModule>
```

```
            <moduleName />
            <foldername />
            <businessControllerClass />
            <codeSubdirectory />
            <isAdmin />
            <isPremium />
            <supportedFeatures> <!-- Requires a value for businessControllerClass. -->
                <supportedFeature type="Portable" /> <!-- The module has import/export capabilities using the IPortable interface. -->
                <supportedFeature type="Searchable" /> <!-- The module can be indexed or searched using the ISearchable interface. -->
                <supportedFeature type="Upgradeable" /> <!-- The module can be upgraded using the IUpgradeable interface. -->
                ...
            </supportedFeatures>
            <moduleDefinition>
                <friendlyName />
                <defaultCacheTime />
                <moduleControls>
                    <moduleControl>
                        <controlKey />
                        <controlSrc />
                        <supportsPartialRendering />
                        <controlTitle />
                        <controlType />
                        <iconFile />
                        <helpUrl />
                        <viewOrder /> <!-- TODO: Joe will check what this is used for. -->
                    </moduleControl>
                    <moduleControl />
                    ...
                </moduleControls>
                <permissions>
                    <!-- In <permission>,
                        "code" is the code for the module,
                        "key" is the code for the permission, and
                        "name" is the user-friendly name for the permission.
                    -->
                    <permission code="..." key="..." name="..." />
                    <permission code="..." key="..." name="..." />
                    ...
                </permissions>
            </moduleDefinition>
        </desktopModule>
        <eventMessage>
            <processorType />
            <processorCommand />
            <attributes>
                <node>value</node>
            </attributes>
        </eventMessage>
    </component>
```

- **Provider**. Extends the list of allowed file extensions.

```
    <component type="Provider" />
```

- **ResourceFile**. Zip files to be expanded during installation. Can be used instead of type "File" to simplify the manifest for packages that contain many files.

```
    <component type="ResourceFile">
        <resourceFiles>
            <basePath /> <!-- Target folder where the contents of the zip file will be installed. Relative to the root of the DNN installation. -->
            <resourceFile>
                <name /> <!-- Name of zip file. -->
            </resourceFile>
            <resourceFile />
            ...
        </resourceFiles>
    </component>
```

- **Script**. Database scripts to upgrade the tables or to install stored procedures that the module needs.

  The following scripts are handled differently:

  - **install.<dataprovidertype>** (e.g., `install.SqlDataProvider`) is executed *before* all other scripts, if the package is being installed for the first time.
  - **upgrade.<dataprovidertype>** (e.g., `upgrade.SqlDataProvider`) is executed *after* all regular scripts.

```
    <component type="Script">
        <scripts>
            <basePath /> <!-- Target base folder for the component installation. Relative to the root of the DNN installation. -->
            <script type="Install" >
                <path /> <!-- Target file folder. Relative to basePath. -->
                <name /> <!-- Must be named "<componentversion>.<dataprovider>". Example: 01.00.00.SqlDataProvider -->
                <version /> <!-- Version of script file to be installed. -->
            </script>
            <script type="UnInstall" >
                <path /> <!-- Location of script file. Relative to basePath. -->
                <name /> <!-- Must be named "uninstall.<dataprovidertype>". Example: uninstall.SqlDataProvider -->
                <version /> <!-- Version of script file to be removed. -->
            </script>
            ...
        </scripts>
    </component>
```

- **SkinObject**. Custom theme objects.

```
    <component type="SkinObject">
        <moduleControl>
            <controlKey /> <!-- Token of the skin object within square brackets []; e.g., [COPYRIGHT] -->
            <controlSrc /> <!-- Target file folder for the theme object's ASCX file. -->
            <supportsPartialRendering /> <!-- "true" if the theme object supports partial rendering through an MS AJAX update panel wrapper. Default
        </moduleControl>
    </component>
```

- **Skin**. All files related to the theme. The installer needs to parse the main theme files at installation time to replace relative folder names; therefore, every file of type **ASCX**, **HTML**, or **CSS** must be declared as a **skinFile**. Other files (i.e., images and scripts) can be packaged using component type "ResourceFile" to simplify the complexity of the theme manifest.

```
    <component type="Skin">
        <skinFiles>
            <basePath /> <!-- Target base folder for the component installation. Relative to the root of the DNN installation. -->
            <skinName />
            <skinFile>
                <path /> <!-- Target file folder. Relative to basePath. -->
                <name />
            </skinFile>
            <skinFile />
            ...
        </skinFiles>
    </component>
```

- **URL Provider**. Custom URL provider to be used with the Advanced URL Management System (AUM).

```
    <component type="URL Provider">
        <name />
        <type />
        <settingsControlSrc />
        <redirectAllUrls />
        <replaceAllUrls />
        <rewriteAllUrls />
        <desktopModule />
    </component>
```

Example

Download a sample theme manifest **MyThemeManifest.dnn8**.

See Also

- Top 5 DotNetNuke Manifest file Module Packaging Tips by Bruce Chapman
- DNN Community blog: DAL 2 - A New DotNetNuke Data Layer for a New Decade by Charles Nurse
- Contents of a Theme Package
- Contents of a Module Package

Sources

- DNN Wiki: Manifests
- DNN Community blog: The New Extension Installer Manifest - Part 1, Introduction by Charles Nurse