World of Tanks: Mod Packages

version 0.5, 2017-10-12

World of Tanks 9.20.1

Author		Contact information
Anton Bobrov	Wargaming.net	
Mikhail Paulyshka	XVM team	mixail@modxvm.com
Andrey Andruschyshyn	Individual	
The Koreanrandom.com community		

1. General Information

A package is a method of arranging modification files. According to the method, all content of a particular modification is packed into a single file.

In case of using the old file distribution scheme, modifications are installed into the following folder `<WoT_game_folder>/res_mods/<WoT_version>/`. According to this scheme, files of different modifications are located in the same folders, thus it is rather difficult to find files of a particular modification.

The package method will make arrangement of modification files far less complicated: to install a modification, a player simply needs to copy a package to the folder `<WoT_game_folder>/mods/<WoT_version>/`, or remove the same file to uninstall the modification.

2. Package Structure

A package is a **zip**-archive with the following features:

- no compression
- extension: `.wotmod`
- the maximum archive size: 2 Gb - 1 byte (2 147 483 647 bytes)

NOTE: compressed archives are not supported in the current version of World of Tanks, thus set the compression level to “without compression” when creating archives.

NOTE: 2 Gb archives and larger are not supported in the current version of World of Tanks, thus they should be split into archives smaller than 2 Gb - 1 byte.

A package contains the following:

- required: the `/res/` folder. A modification content is located in this folder, i.e., the very files that used to be installed in the following folder: `<WoT_game_folder>/res_mods/<WoT_version>`
- optional: utility file `meta.xml` (view **section 5**)
- optional: file `LICENSE` containing a license agreement
- optional: any other content that a modification developer might need: link to the modification web page, documents, change list, etc.

Example of a package structure:

```
/package.wotmod
               /meta.xml
               /README.md
               /LICENSE
               /res
                   /scripts
                           /client
                                  /gui
                                      /mods
                                           /mod_example.pyc
```

3. Installing a Package

Packages are to be installed in the following folder: `<WoT_game_folder>/mods/<WoT_version>`. They can either be copied manually, or installed via a special installer file that contains a particular modification or a pack of modifications.

If required, packages can be split into sub-folders, which allows developers to arrange files in particular groups:

```
mods/
    0.9.17.1/
            MultiHitLog_2.8.wotmod
            DamagePanel/
                        Some_common_library_3.14.5.wotmod
                        DamagePanel_2.6.wotmod
                        DamagePanel_2.8.wotmod
                        DamagePanel_2.8_patch1.wotmod
```

4. Recommendations on Naming Packages

We recommend using the following scheme when creating a package identifier (hereinafter `package_id`):

```
package_id = author_id.mod_id
```

Where:

- `author_id`: a developer identifier. It can be either a developer's inverted domain (`com.example`) or the developer's nickname (`noname`)
- `mod_id`: a modification identifier. It is selected at developer's discretion.

Package identifier is used in the `<id>` field of the `meta.xml` file (view **section 5**) and as a part of the package file name..

Examples of package identifiers:

- `com.example.coolmod`;
- `noname.supermod`.

A package name is created according to the following scheme:

```
<author_id>.<mod_id>_<mod_version>.wotmod
```

Where:

- `mod_version`: modification version, specified by the modification developer in the `<version>` field of the `meta.xml` file (view **section 5**).

Examples of file names:

- `com.example.coolmod_0.1.wotmod`;
- `noname.supermod_0.2.8.wotmod`.

5. Metadata File `meta.xml`

The `meta.xml` optional file contains special fields for describing a modification.

Example:

```
<root>
    <!-- Package identifier -->
    <id>noname.crosshair</id>

    <!-- Package version -->
    <version>0.2.8</version>

    <!-- Package name clear for players -->
    <name>Crosshair</name>

    <!-- Package description -->
    <description>New cool Crosshair with feature1.....N</description>
</root>
```

Values specified in the `<id>` and `<version>` fields are used for determining the order of loading packages. Values specified in the `<name>` and `<description>` fields will subsequently be used in the modification management system.

6. Loading Packages

6.1 Order of Loading

All packages located in the `<WoT_game_folder>/mods/<WoT_version>/` folder are sorted by the `<id>` node value specified in the `meta.xml` file and are loaded according to this order. If the `meta.xml` file is missing, the file name will be used as the package identifier.

The `load_order.xml` file can be used for changing the order of loading. It should be located in the abovementioned folder.

If all packages are specified in the `load_order.xml` file, they are loaded according to the order set in the file.

If some packages are not specified in the `load_order.xml` file, packages specified in `load_order.xml` are loaded first. The rest of the packages are loaded in alphabetical order.

6.2 Using Packages Together with the `res_mods` Folder

From the point of the game client, the virtual system root is formed of:

- `/res_mods/<WoT_version>`
- `/mods/<WoT_version>/<package_name>.wotmod/res/`
- `/res/packages/*.pkg/`
- `/res/`
- Other locations specified in the `<WoT_game_folder>/paths.xml` file

These paths are listed descending by priority. I.e., files located in the `/res_mods/<WoT_version>/` folder have higher priority regardless of the `load_order.xml` file.

6.3 Resolving Conflicts that Occur upon Loading

Generally, the package method does allow a situation, when identical files are located within different packages in the `res/` folder. Such situations are considered to be conflicts.

If a conflict is detected, the conflicted package is not loaded, a corresponding is displayed to the user.

In other words, if both packages `a.wotmod` and `b.wotmod` contain the `res/scripts/entities.xml` file, the `a.wotmod` package will be loaded successfully, while the `b.wotmod` package will cause a conflict and thus will not be loaded.

Use the following to handle conflicts:

1. The `load_order.xml` file

The `load_order.xml` file should be located in the following folder: `<WoT_game_folder>/mods/<WoT_version>/`. It is formed in the following way:

```
<root>
    <Collection>
        <pkg>package1_name.wotmod</pkg>
        <pkg>package2_name.wotmod</pkg>
        <!-- ... -->
        <pkg>packageN_name.wotmod</pkg>
    </Collection>
</root>
```

Packages specified in this file are not regarded as conflicts. They are loaded without checking for identical names. A package file specified at the end has the highest priority.

2. Values of the `<id>` and `<version>` nodes from `meta.xml`

If the `<id>` node is specified in the `meta.xml` file, names of package files are not considered in the loading order. Packages that have identical `<id>` values, are regarded as different versions or parts of one modification. Conflicts between such elements are not considered. They are loaded in the version order (versions are specified in the `<version>` node).

Packages versions are compared by characters according to the ASCII table. The behavior is similar to the behavior of the following function: strcmp():

- version `9.0.0` has a higher priority than version `10.0.0`;
- version `b` has a higher priority than version `B`;
- version `c<any characters>` has a higher priority than version `c`;
- if versions are identical, packages are loaded in the alphabetical order.

If different packages contain files with identical names, and the conflicts they cause are resolved with the `load_order.xml` or `meta.xml` files, the file from the most recently added package has a higher priority.

6.4 Executing Python Code

After adding all packages and resolving conflicts, all .pyc files with names starting from `mod_` located the `/scripts/client/gui/mods/` folder are executed in alphabetical order.

Within a package, this file should be located here:

```
<author_id>.
<mod_id>_<version>.wotmod/res/scripts/client/gui/mods/mod_<anything>.pyc
```

7. Recommended Paths for Modification Files

7.1 Configuration Files

Modification configuration files are recommended to be located here:

```
<WoT_game_foler>/mods/configs/<author_id>.<mod_id>/
```

Where:

- `author_id` и `mod_id` - identifiers described in **section 4** of this document.

7.2 Log Files

Apart from the `python.log` standard file, it is recommended to use the following path:

```
<WoT_game_folder>/mods/logs/<author_id>.<mod_id>/
```

Where:

- `author_id` и `mod_id` - identifiers described in **section 4** of this document.

7.3 Temporary Files

Temporary files are recommended to be located here:

```
<temp>/world_of_tanks/<author_id>.<mod_id>/
```

Where:

- `temp` - path to a folder containing temporary files for a current user in the OS;
- `author_id` и `mod_id` - identifiers described in **section 4** of this document.

7.4 Other Modification Files

Use the following path to store content that should be accessible in the game client:

```
<package_name>.wotmod/res/mods/<author_id>.<mod_id>/
```

Where:

- `author_id` и `mod_id` - identifiers described in **section 4** of this document.

8. Working with Files within Packages

Use the `ResMgr` module for working with files within packages.

8.1 Standard Operations

8.1.1 Reading a File from a Package

```
#import
import ResMgr

#function
def read_file(vfs_path, read_as_binary=True):
    vfs_file = ResMgr.openSection(vfs_path)
    if vfs_file is not None and ResMgr.isFile(vfs_path):
        if read_as_binary:
            return str(vfs_file.asBinary)
        else:
            return str(vfs_file.asString)
    return None

#example
myscript = read_file('scripts/client/gui/mods/mod_mycoolmod.pyc')
```

8.1.2 Obtaining a List of Elements in a Folder

```
#import
import ResMgr

#function
def list_directory(vfs_directory):
    result = []
    folder = ResMgr.openSection(vfs_directory)

    if folder is not None and ResMgr.isDir(vfs_directory):
        for name in folder.keys():
            if name not in result:
                result.append(name)

    return sorted(result)

#example
content = list_directory('scripts/client/gui/mods/')
```

8.1.3 Copying a File from a Package to a Folder

```
#import
import os
import ResMgr

#function
def file_copy(vfs_from, realfs_to)
    realfs_directory = os.path.dirname(realfs_to)
    if not os.path.exists(realfs_directory):
        os.makedirs(realfs_directory)

    vfs_data = file_read(vfs_from) #view 8.1.1
    if vfs_data:
        with open(realfs_to, 'wb') as realfs_file:
            realfs_file.write(vfs_data)

#example
file_copy('scripts/client/gui/mods/mod_my.pyc','res_mods/0.9.17.1/scripts/client/gui/mods/mod_my.pyc')
```

9. Known Issues

9.1 Executing `.py` files

Issue description

Currently, '.py' files located inside a package cannot be executed.

Temporary solution

A package should contain both `.py` files and compiled into bytecode `.pyc` files.

9.2 Partial support of the ZIP format

Issue description

Currently, it is impossible to use `.wotmod` files that do not have the `ZIPDIRENTRY` and `ZIPFILERECORD` structures for all folders inside the archive.

Temporary solution

Use compatible archivers for creating archives, for example:

- 7-Zip http://7-zip.org ;
- Info-ZIP http://info-zip.org/ .

Attachment A. Change-list

v 0.5 (2017-10-12)

- Points 9.2 and 9.3 removed (fixed in World of Tanks 9.20.1)
- Added the description of the issue related to partial support of the ZIP format.

v 0.4 (2017-05-04)

- reworked description of resolving package conflicts with the `load_order.xml` file.

v 0.3 (2017-05-03)

- added information on restrictions to the `.wotmod` file formats;
- supplemented description of resolving conflicts for packages with identical identifiers

v 0.2 (2017-04-10)

- reworked design: new layout, separation into articles
- reworked description of the package naming scheme
- reworked description of the package loading order
- added recommendations concerning the locations of logs and configuration files
- added examples of the source code for working with files within packages
- added description of currently known issues

v 0.1 (2017-01-13)

- First version.