World of Tanks: Mod Packages

version 0.2, 2017-04-10

- Anton Bobrov, Wargaming.net
- Mikhail Paulyshka, XVM team
- Andrey Andruschyshyn, Independent
- 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 without compression with the `.wotmod` extension.

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.

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 the following scheme of naming a package (hereinafter `package_id`):

```
package_id = author_id.mod_id
```

Where:

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

This name is used in the `meta.xml` file (view **section 5**) and as a part of the package file name.

Example of package names:

```
com.example.coolmod
noname.supermod
```

A file name is formed in the following way:

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

Where:

- `mod_version`: modification version, set by the modification developer in `meta.xml` (view **section 5**).

Examples:

```
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>
    <!-- Techical MOD ID -->
    <id>noname.crosshair</id>

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

    <!-- Human readable name -->
    <name>Crosshair</name>

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

Values specified in these 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.

NOTE: currently, it is rather difficult to use the `load_orders.xml` file (view **section 9.4**).

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 entirely excluded from processing. In such a manner, a corresponding notification 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` will cause a conflict and thus will not be loaded.

Use the following to handle conflicts:

1. The `load_order.xml` file. Packages specified in this file are not regarded as conflicts. They are loaded without checking for identical names.
2. The `<id>` node value from `meta.xml`. Packages that have identical `<id>` values, are regarded as different versions or parts of one modification. Conflicts between such elements are not considered.

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

```python
#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 Case-Sensitive File Names

Issue Description

Currently, when adding files to the virtual file system:

- files from packages are added in the lower-case
- files from the `<WoT_game_folder>/res_mods/` folder are added as they are

As a result, if a file is located both in a package and in the `res_mods` folder, and the file name contains at least one upper-case letter, the file may load twice.

Temporary Solution

Use only lower-case letters for names of files and folders located in the `<WoT_game_folder >/res_mods` folder.

9.2 Working with the GNU Gettext Files

Issue Description

Currently it is impossible to assign the `.mo` files in a package instead of the `.mo` files located in the `<WoT_game_folder>/res/text/LC_MESSAGES/` folder.

Temporary Solution

Use net.openwg.vfsgettext as a temporary solution:

http://openwg.net/download/vfsgettext/net.openwg.vfsgettext_1.0.0.wotmod

9.3 Executing the `.py` Files

Issue Description

Currently, the `.py` files located in a package cannot be executed.

Temporary Solution

Add the compiled into byte code `.pyc` files to a package, in addition to the `.py` files.

9.4 Changing the Order of Loading Packages

Issue Description

Currently, it is impossible to change the order of loading packages using the `load_order.xml` file.

Temporary Solution

There is no temporary solution to the issue. The issue is going to be resolved soon.

Attachment A. Change-list

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