

Improved Linux Download Scripts

by Hartmut Buhrmester

Introduction

I like to introduce a complete rewrite of the Linux download scripts for the project WSUS Offline Update. These scripts offer many improvements over the legacy script DownloadUpdates.sh:

- Separation of a frontend and backend script

The script update-generator.bash is used to interactively select the update, language and download options. The script download-updates.bash fetches the selected updates without any user interaction. This separation makes the structure of both files more straightforward.

- Highly modular approach

Both scripts are further split into libraries, common tasks, setup tasks and download tasks. Each script does one task only in the most straightforward manner. This resembles the flow of control and makes the scripts easily expandable and more maintainable.

- Unified language settings

There is no distinction between default languages, custom languages and update languages.

Users can specify *one* language on the command line, and then they will get downloads for the specified language only, and nothing more.

- Verification of downloaded files

SHA-1 hashes are embedded into the filename of all security updates, as a number of 40 hexadecimal digits. These are compared to the checksums, which are calculated by hashdeep.

The verification of digital file signatures with Sysinternals Sigcheck running under wine was tried, but it doesn't really work without the necessary root certificates.

- Compatibility

The download script uses the same algorithms for calculating superseded and dynamic updates as the Windows script DownloadUpdates.cmd. The compliance with the Windows scripts can be tested with the scripts compare-integrity-database.bash and compare-update-tables.bash.

- Desktop integration

Obsolete updates are not deleted immediately, but moved into the trash. GNOME and most other GTK+ based desktop environments use GVFS to handle the trash. The package trash-

cli can be used with other desktop environments or window managers. trash-cli should also work without any graphical environment.

- Self updates of WSUS Offline Update

Both the setup and the download script check for new versions of WSUS Offline Update. They also handle updates of the configuration files in the static and exclude directories.

- Same day rules

Same day rules are used to prevent the repeated evaluation of the same tasks in adjacent runs of the download script.

- Documentation

There is even a complete documentation.

Compatibility

The scripts are only tested on:

- Debian 7 Wheezy
- Debian 8 Jessie
- Debian 9 Stretch, the current *stable* since June 2017

Other Linux distributions should just work fine, if the needed applications are installed (see below).

FreeBSD may work, because it seems to use many GNU utilities. For example, the manual page for *grep* refers to the GNU Project. The command *readlink* in FreeBSD seems to be compatible with that in Linux.

- <https://www.freebsd.org/cgi/man.cgi>

Mac OS X (macOS) is different: it uses some GNU utilities, but they all stay at GPL v2. Everything with GPL v3 is shunned by Apple. This means, that the bash will stay at version 3.2.53 forever. This version should still be sufficient, though, because the scripts don't use more recent features. Also, developing for a plain POSIX shell is not the answer, since this affects all other commands as well. For example, the command *readlink* in Mac OS X is *not* compatible with the same command in Linux, and there are long discussions in the Internet about this one utility:

- <https://stackoverflow.com/questions/1055671/how-can-i-get-the-behavior-of-gnus-readlink-f-on-a-mac>

Requirements

The download script uses some additional applications, which can usually be installed from the repositories of the Linux distribution.

Required applications

- **cabextract** is used to extract the file package.xml from the WSUS catalog file.
- **XMLStarlet** is used to extract information from the file package.xml, to calculate dynamic and superseded updates. It is provided by the package xmlstarlet.

Note, that the installed binary may be /usr/bin/xmlstarlet in Debian and Red Hat, or /usr/bin/xml in distributions, which use the unmodified upstream source code from <http://xmlstar.sourceforge.net/> directly. Despite such possible differences, the package name should always be *xmlstarlet*, and there never was a package *xml*.

- **wget** is the standard interactive download utility for Linux. It should be installed by default in Linux distributions, but this may not be true for Mac OS X and other BSDs.
- **hashdeep** is used to create an integrity database of the downloaded files. This allows a simple integrity check: For most security updates, the SHA-1 hash is embedded into the filename as a hexadecimal number of 40 digits. For example, if the filename is ndp35sp1-kb958484-x64_e69006433c1006c53da651914dc8162bbdd80d41.exe, then the SHA-1 hash of the file is e69006433c1006c53da651914dc8162bbdd80d41. After calculating the hashes with hashdeep, they can be easily compared to the expected values.

The application hashdeep is provided by the package md5deep or hashdeep, depending on your distribution. See the *Installation* section later in this document.

Recommended applications

- **gvfs-trash** or **trash-put** can be used to move old files into the trash, rather than deleting them directly.

The virtual file system GVFS is used by GNOME and other GTK+ based desktop environments. gvfs-trash is provided by the package gvfs-bin in Debian.

trash-put is provided by the package trash-cli. These are Python scripts, which implement the FreeDesktop.org Trash specification. They can be used with other desktop environments and window managers, and they should even work without any graphical user interface.

The local trash directory is \$HOME/.local/share/Trash/.

Creating a trash directory on external drives requires sufficient rights. It will be an invisible directory .Trash-1000 at the root level of the drive. The number 1000 is the user ID of the first regular user on Debian. It will be .Trash-500 on Fedora.

Optional applications

- **Aria2** features multiple connections, to speed up the download of large files. The time stamping feature of Aria2 usually works better than that of Wget 1.16 and lower.

Wget 1.16 always uses *two* queries for each download: a HEAD query to get the file size and modification date of the remote file, and a GET query to download the file if it is newer than the local file. Wget 1.16 also downloads a file again, whenever the file size changes, regardless of the file modification date. In a content delivery network, and with files which change very often like the virus definition files, this may lead to all kinds of strange behavior.

Aria2 only uses *one* GET query, along with a conditional header If-Modified-Since. Then the server can decide, if the server file is newer than the local file.

Wget 1.18 in Debian 9 Stretch uses the same approach as Aria2 for time stamping.

Note, that the application Aria2 is provided by the package `aria2` in Debian, but the installed binary is `/usr/bin/aria2c`.

If you like to use Aria2, change the search order in the file preferences.bash to:

```
supported_downloaders="aria2c wget"
```

- **wine** can be used to run Sysinternals Sigcheck on Linux. This could be used to validate digital file signatures, but it doesn't work so far. See a discussion below in the chapter *Validation of downloaded files*.
- **rsync** is used by the optional script `70-synchronize-with-target.bash`, to synchronize the client directory with another directory, for example on a USB drive.

Download of the current version

The current version is 1.0.

The installation archive, a corresponding hashdeep checksum file, and the results of a virus scan at VirusTotal are available at:

<http://downloads.hartmut-buhrmester.de/>

Installation

WSUS Offline Update already includes the new Linux download scripts. You don't need to install the Linux scripts separately, as it was necessary for the first beta versions, but you should review the needed packages from your Linux distribution.

Install the required and recommended packages

For Debian and Debian-derived distributions, you need to distinguish between the packages *md5deep* and *hashdeep*.

The upstream developers moved their project from SourceForge to GitHub, and they renamed their project from *md5deep* to *hashdeep*:

<http://md5deep.sourceforge.net/>
<https://github.com/jessek/hashdeep/>

Debian followed this move and renamed the package md5deep to hashdeep, starting with Debian 8 Jessie-Backports in summer 2015. The general rule for Debian and Debian-derived distributions then is: Install the package md5deep, if the distribution was released before 2015. Install the package hashdeep for all recent distributions.

- For Debian 7 Wheezy:

```
su -  
aptitude install cabextract md5deep wget xmlstarlet trash-cli
```

- For Debian 8 Jessie-Backports and newer:

```
su -  
aptitude install cabextract hashdeep wget xmlstarlet trash-cli
```

- For Ubuntu 14.04LTS Trusty:

```
sudo apt-get install cabextract md5deep wget xmlstarlet trash-cli
```

- For Ubuntu 16.04LTS Xenial and newer:

```
sudo apt-get install cabextract hashdeep wget xmlstarlet trash-cli
```

Other distributions, which are not Debian-based, seem to stay with the package name md5deep.

Note, that both packages md5deep and hashdeep install a series of related applications: hashdeep, md5deep, sha1deep, sha256deep, tigerdeep, and whirlpooldeep. Throughout WSUS Offline Update, you always need the application *hashdeep*, regardless of the package name.

Download and unpack the wsusoffline archive

Download the newest wsusoffline archive from the download page <http://download.wsusoffline.net/> and unpack it. Note, that the zip archive comes with an accompanying hashes files. You can use it to verify the download with:

```
hashdeep -a -v -v -l -k wsusoffline1111_hashes.txt wsusoffline1111.zip
```

The new Linux scripts are included in the "sh" subdirectory. Due to the packaging on Windows, the scripts are not yet executable. Run the included script fix-file-permissions.bash once as:

```
bash fix-file-permissions.bash
```

to make the scripts update-generator.bash, download-updates.bash, get-all-updates.bash, and some others executable.

You can then use the script update-generator.bash to interactively select one update, language and the available options to download. This script doesn't allow multiple selections for update and language, though.

You can also use the script `get-all-updates.bash` as a template: This script downloads all updates with all available options for the default languages German and English. But it is also meant for customization – you can simply comment out or delete all items you don't need.

Notes

The new Linux scripts don't work alone – they need the configuration files from the `wsusoffline` installation. Also, the Linux download scripts can only replace the Windows download scripts, e.g. `DownloadUpdates.cmd`. To install the updates, you surely need the files in the client directory, e.g. the application `UpdateInstaller.exe`.

Therefore, you should not download the Linux scripts separately, as it was necessary for the first beta versions. Just get the latest `wsusoffline` archive and find the Linux scripts in the "sh" subdirectory.

If you need to copy or move the `wsusoffline` directory, please make sure to keep the modification date of all files. You could use `"cp --archive"` or `"cp --preserve"` instead of just `"cp"`. This is important for all files throughout WSUS Offline Update.

You can find the complete documentation in the subdirectory *documentation*.

Configuration

The download scripts don't need an initial configuration. You can, however, edit some permanent settings in the file `preferences.bash`:

- The preferred download utility: GNU Wget or Aria2
- Proxy servers can be set in the file `preferences.bash`, but there are more ways to do so: Desktop environments like GNOME and KDE may use their own settings for proxy servers. You can also define them as environment variables in the file `~/.profile` or edit the preferences files for the download utilities Wget and Aria2.
- If you don't need Silverlight, you can set the option `include_win_glb` to *disabled*.

Monthly quality versus security-only updates

By default, WSUS Offline Update downloads and installs the full monthly *quality* update rollups for Windows 7 and Windows Server 2008 R2, Windows Server 2012, Windows 8.1 and Windows Server 2012 R2. To prefer *security-only* update rollups, change the option `prefer_seconly` to *enabled* in the file `preferences.bash`.

The switch from *quality* to the *security-only* updates uses two sets of configuration files. The files

```
../client/exclude/HideList-seconly.txt  
../client/exclude/custom/HideList-seconly.txt
```

exclude the monthly *quality* updates from download, while the files

```
../client/static/StaticUpdateIds-w61-seconly.txt
../client/static/StaticUpdateIds-w62-seconly.txt
../client/static/StaticUpdateIds-w63-seconly.txt
../client/static/custom/StaticUpdateIds-w61-seconly.txt
../client/static/custom/StaticUpdateIds-w62-seconly.txt
../client/static/custom/StaticUpdateIds-w63-seconly.txt
```

include the *security-only* update rollups in download and installation. The files in the directories `../client/exclude` and `../client/static` are maintained by the author of WSUS Offline Update. They can be replaced at any time. Their counterparts in the directories `../client/exclude/custom` and `../client/static/custom` can be created for manual customization.

Overviews for the *quality* and *security-only* update rollups can be found at Microsoft:

- [Windows 7 SP1 and Windows Server 2008 R2 SP1 update history](#)
- [Windows Server 2012 update history](#)
- [Windows 8.1 and Windows Server 2012 R2 update history](#)

These Windows update rollups include updates for the .NET Frameworks, which are preinstalled in the respective Windows versions.

If other versions of the .NET Frameworks are installed manually, they must be updated separately. Since October 2016, these .NET updates are also released as full *quality* or *security-only* updates. So far, such updates have been released in [October 2016](#), [December 2016](#), [April 2017](#), and [May 2017](#).

- [.NET Blog](#)
- [.NET Framework Monthly Rollups Explained](#)

Viewing progress with wget

With GNU Wget 1.16 and lower, all messages are written to the log file. There is no progress indicator in the terminal window, in which the script is run. It is recommended to open another terminal window and view the progress with:

```
tail -F ../log/download.log
```

With GNU Wget 1.18, you can use the option `--show-progress` to display a progress bar in the terminal window, while the rest of the output is written to the log file. This option must be manually added to the configuration variable `wget_common_options` in the file `40-configure-downloaders.bash`.

Usage

New users should just run the script `update-generator.bash` to interactively set up the download. This script doesn't have any command-line options. Just run it as:

```
./update-generator.bash
```

After selecting the update, language and optional downloads, the setup script shows the download command for confirmation, and then passes execution to the download script.

The script `download-updates.bash` is meant to run without any user interaction. It will ask for confirmation, if there are new versions of WSUS Offline Update or the Linux download scripts available, but this questions default to *no* after 30 seconds. This answer can be changed to *yes* by setting the option `unattended_updates` to *enabled* in the preferences file.

Once you are familiar with the different settings, you could customize the script `get-all-updates.bash` to get the downloads you need.

Language settings in the Windows scripts

The language settings for the Windows script `DownloadUpdates.cmd` are quite complicated, as it distinguishes between *default* languages, *custom* languages, and *update* languages.

All Windows versions since Windows Vista are considered to be *global*, but they still include localized installation files for:

- Internet Explorer 9 on Windows Vista
- Internet Explorer 11 on Windows 7
- .NET Framework language packs for languages other than English
- Microsoft Security Essentials (MSSE)

By default, WSUS Offline Update downloads these files in the two most often used languages, German and English. Therefore, the supposedly global file `StaticDownloadLinks-dotnet-x64-glb.txt` just contains one German language pack. Other languages can be added as *custom languages*.

Handling these languages requires at least four additional scripts:

```
AddCustomLanguageSupport.cmd  
RemoveCustomLanguageSupport.cmd  
RemoveEnglishLanguageSupport.cmd  
RemoveGermanLanguageSupport.cmd
```

But this turned out to be quite complicated for both users and other developers.

A unified approach for language settings

The new Linux scripts use a unified approach:

1. The default languages German and English are removed from the global static download files on the first run.
2. Users must always specify one real language like *deu* or *enu* on the command line; the placeholder *glb* is not allowed for any update.

Since version 1.0-beta-4, you can also join several languages to a comma-separated list like *deu,enu*.

3. These languages are used wherever a language setting is needed: They are used like the *default* and *custom* languages to include localized downloads for Internet Explorer, .NET Frameworks and Security Essentials. For Office 2007 – 2013, they are used as the *update* languages.
4. This way, users get downloads for the specified languages only, and nothing else.

So far, the setup script `update-generator .bash` does not support multiple selections. If you use this script to select all options, you may end up with downloading several languages in turn.

If different languages are downloaded in turn, then previous downloads in languages other than the selected one must be preserved between runs. The cleanup function handles this by treating the complete static directory as an additional white list. Technically, this is just a recursive `grep` for the filename. Files, which are not in the current download set, but which can still be found in the static directory, are reported as *valid static files*. These files are never automatically deleted. If they are not needed anymore, they must be manually deleted once, and then they won't get downloaded again.

While the concept of *valid static files* was introduced to preserve localized downloads between runs, the same mechanism also protects some other files:

- If service packs have been downloaded before, and the option `-includesp` is not used, the files are still preserved.
- 64-bit Office downloads are not deleted, if the corresponding 32-bit downloads are selected.

Again, if these files are not needed anymore, they must be manually deleted.

Combining different languages to a comma-separated list will overcome such considerations, and it allows a faster evaluation of Windows downloads by avoiding unnecessary repetitions. For Office downloads, it is merely a convenience, but it doesn't make the script run faster.

Command-line options

The command-line options for the download script are:

`download-updates.bash`: Download updates for Microsoft Windows and Office

USAGE

```
./download-updates.bash UPDATE LANGUAGE[,LANGUAGE...] [OPTIONS]
```

UPDATE

w60	Windows Server 2008, 32-bit
w60-x64	Windows Server 2008, 64-bit
w61	Windows 7, 32-bit
w61-x64	Windows 7 / Server 2008 R2, 64-bit
w62-x64	Windows Server 2012, 64-bit
w63	Windows 8.1, 32-bit
w63-x64	Windows 8.1 / Server 2012 R2, 64-bit

w100	Windows 10, 32-bit
w100-x64	Windows 10 / Server 2016, 64-bit
o2k10	Office 2010, 32-bit
o2k10-x64	Office 2010, 32-bit and 64-bit
o2k13	Office 2013, 32-bit
o2k13-x64	Office 2013, 32-bit and 64-bit
o2k16	Office 2016, 32-bit
o2k16-x64	Office 2016, 32-bit and 64-bit

LANGUAGE

deu	German
enu	English
ara	Arabic
chs	Chinese (Simplified)
cht	Chinese (Traditional)
csy	Czech
dan	Danish
nld	Dutch
fin	Finnish
fra	French
ell	Greek
heb	Hebrew
hun	Hungarian
ita	Italian
jpn	Japanese
kor	Korean
nor	Norwegian
plk	Polish
ptg	Portuguese
ptb	Portuguese (Brazil)
rus	Russian
esn	Spanish
sve	Swedish
trk	Turkish

Note: Multiple languages can be joined to a comma-separated list like "deu,enu".

OPTIONS

- includesp
Include Service Packs
- includecpp
Include Visual C++ runtime libraries
- includedotnet
Include .NET Frameworks: localized installation files and updates
- includewddefs
Virus definition files for Windows Vista and 7. These virus definition files are only compatible with the original Windows Defender, which was included in Windows Vista and 7.
- includemsse
Microsoft Security Essentials: localized installation files and virus definition updates. Microsoft Security Essentials is an optional installation for Windows Vista and 7.

- includewddefs8
Virus definition files for Windows 8 and higher. These are the same virus definition updates as for Microsoft Security Essentials, but without the localized installers.

COMPATIBILITY

- The option -includesp can be used with all updates.
- The options -includecpp and -includedotnet can be used with all Windows updates.
- The options -includewddefs and -includemsse can be used with Windows 7, Windows Server 2008 and 2008 R2 (w60, w60-x64, w61, w61-x64).
- The option -includewddefs8 can be used with Windows 8.1 and 10, Windows Server 2012, 2012 R2 and 2016 (w62-x64, w63, w63-x64, w100, w100-x64).

NOTES

Windows Vista and higher are multilingual, but WSUS Offline Update still needs the correct language settings to get:

- localized installers for Internet Explorer on Windows 7, Windows Server 2008 and 2008 R2
- language packs for .Net Frameworks, for languages other than English
- localized installers for Microsoft Security Essentials

In the Linux download scripts, all needed languages must be given on the command-line. For convenience, you can join several languages to a comma-separated list, to allow a faster evaluation of Windows 7, Windows Server 2008 and 2008 R2, .Net Frameworks, and Microsoft Security Essentials.

EXAMPLES

To get updates for Windows 7 with all optional downloads in German and English, you could either use:

```
./download-updates.bash w61 deu -includesp -includecpp \  
-includedotnet -includewddefs -includemsse  
./download-updates.bash w61 enu -includesp -includecpp \  
-includedotnet -includewddefs -includemsse
```

or:

```
./download-updates.bash w61 deu,enu -includesp -includecpp \  
-includedotnet -includewddefs -includemsse
```

This should get the same updates as the Windows version in its default configuration, using the default languages German and English.

To get updates for Windows 8.1 with all optional downloads in French and Spanish, you could use:

```
./download-updates.bash w63 fra,esn -includesp -includecpp \  
-includedotnet -includewddefs8
```

To get the same results with the Windows version, you need to run four additional scripts first:

```
RemoveGermanLanguageSupport.cmd  
RemoveEnglishLanguageSupport.cmd  
AddCustomLanguageSupport.cmd fra  
AddCustomLanguageSupport.cmd esn
```

See the script `get-all-updates.bash` for more examples. This script may also serve as a template for customization.

This description is also available in the file `usage.txt` and at the top of the download script itself.

Comparing the results on Windows and Linux

Selecting the same options in Windows and Linux should result in the same downloads. Optimally, files should not just be compared by their name, but also by their content. This would take a long time for two directories of about 30 GB each. But fortunately, most of this work has already been done by creating the *integrity database* of hashdeep files in the `client/md` directory.

- Each hashdeep file corresponds to one download directory.
- Each line of a hashdeep file is a fingerprint of one downloaded file. It consists of the file size, MD5, SHA-1 and SHA-256 hashes, and the relative file path.

Thus, comparing two directories of small text files is enough for a deep comparison of all downloaded files. This can be easily done with `diff`. Comparing the hashes files also ensures, that these files are in the correct format to be used by the Windows script `DoUpdate.cmd` during installation.

The script `compare-integrity-database.bash` in the directory `comparison-linux-windows` is meant for this comparison. The file `example-results-md.txt` shows typical results: The four virus definition files are usually different, because they change every two hours, but the other files should be the same.

The script `compare-update-tables.bash` does a similar comparison of the `*.csv` files, which are used for the installation of Office updates.

Validation of downloaded files

There are at least three different approaches to validate downloaded files:

Comparing file hashes

For all security updates extracted from the WSUS catalog file `wsusscn2.cab`, the SHA-1 hash is embedded into the filename. It can be easily compared to the calculated hash of the file.

Unfortunately, this doesn't work for the WSUS catalog file itself and for the virus definition files. But these files create most problems.

Testing the file integrity with cabextract

The integrity of the WSUS catalog file `wsusscn2.cab` is tested with `cabextract -t`. This ensures, that all files in the archive can be expanded. This is not necessary for the other cab archives, because they can be tested by using the SHA-1 hash, which is embedded in the file name, as the reference value.

Validating digital file signatures with Sysinternals Sigcheck

Sysinternals Sigcheck does run under wine, but there are a few drawbacks:

- The built-in wine library `CRYPT32.dll` doesn't seem to provide the functionality to really validate file signatures. With this library, Sigcheck can only tell, if a file is *signed* or *unsigned*. That much actually works, but it is not enough to detect subtle problems with the downloaded files. If a file is digitally signed, but the file is damaged for some reason, it is still reported as *signed*. The correct result should be that the signature could not be verified.
- The utility `winetricks` can replace the built-in wine library with a native Windows library. But without the necessary root certificates and complete certificate chains, Sigcheck shows a generic error message for every single file.

Thus, although a preliminary implementation for wine and Sigcheck exists, it needs more work, especially to transfer the root certificates from Windows to Linux.

Validating digital file signatures with chktrust

The .NET Framework on Windows and the Mono Framework on Linux provide similar utilities for the same tasks.

The .NET Framework on Windows provides a Certificate Manager for the Microsoft Management Console, which can be launched as:

```
mmc.exe certmgr.msc
```

The Certificate Verification tool (`chktrust.exe`) and the Sign tool (`signtool.exe`) can be used to verify digital signatures.

- <https://msdn.microsoft.com/en-us/library/z045761b%28v=vs.100%29.aspx>
- <https://msdn.microsoft.com/en-us/library/8s9b9yaz%28v=vs.110%29.aspx>

The Mono framework for Linux provides two similar utilities: `certmgr` and `chktrust`, but the command `chktrust` can only verify executable files, not cab archives.

- <http://www.mono-project.com/docs/tools+libraries/tools/>

Still, this might be the easiest way to transfer certificates from Windows to Linux. Since wine seems to integrate with the Mono framework, these certificates might even work with Sigcheck.

Missing functionality

- Creation of ISO images

WSUS Offline Update was once designed to create custom update CDs/DVDs. Therefore, the contents of the client directory can be written to an ISO image file. The ISO image can then be burned to a writable CD or DVD. The file `client/autorun.inf` ends up in the root directory of the CD/DVD and starts the update, when the disk is inserted.

But are optical media still used anymore? The client directory can be copied to an external USB drive instead. If an archive of the client directory is required, then I would recommend an *uncompressed* tar archive. Just don't use a compressed format like `*.tar.gz`, because this won't achieve anything, if the input are huge, already highly compressed files.

- Download from local WSUS servers

Implementing and testing this function would require a WSUS server, which I don't have.

Also, downloading from a local WSUS server may not work as expected: Only *dynamic* download links, which are extracted from the WSUS catalog file `wsusscn2.cab`, can be redirected to a local WSUS server. *Static* download links are still downloaded from the Microsoft download sites. This includes the catalog file `wsusscn2.cab` itself and all other download links from the static download files in the `wsusoffline/static` directory. The domain `wsusoffline.net` will also be contacted to check for updates of the configuration files.

So, the *download part* of WSUS Offline Update does not really work without an Internet connection, and it was never meant to do so. The *installation part* of WSUS Offline Update does work without Internet connections.

Local WSUS servers may still be useful to restrict the downloads to *approved* updates only.

File version

This file was last modified on 2018-01-19.