Versions

OTP Version

As of OTP release 17, the OTP release number corresponds to the major part of the OTP version. The OTP version as a concept was introduced in OTP 17. The version scheme used is described in detail in Version Scheme.

OTP of a specific version is a set of applications of specific versions. The application versions identified by an OTP version corresponds to application versions that have been tested together by the Erlang/OTP team at Ericsson AB. An OTP system can, however, be put together with applications from different OTP versions. Such a combination of application versions has not been tested by the Erlang/OTP team. It is therefore *always preferred to use OTP applications from one single OTP version*.

Release candidates have an `-rc<N>` suffix. The suffix `-rc0` is used during development up to the first release candidate.

Retrieving Current OTP Version

In an OTP source code tree, the OTP version can be read from the text file `<OTP source root>/OTP_VERSION`. The absolute path to the file can be constructed by calling `filename:join([code:root_dir(), "OTP_VERSION"])`.

In an installed OTP development system, the OTP version can be read from the text file `<OTP installation root>/releases/<OTP release number>/OTP_VERSION`. The absolute path to the file can by constructed by calling `filename:join([code:root_dir(), "releases", erlang:system_info(otp_release), "OTP_VERSION"])`.

If the version read from the `OTP_VERSION` file in a development system has a `**` suffix, the system has been patched using the `otp_patch_apply` tool. In this case, the system consists of application versions from multiple OTP versions. The version preceding the `**` suffix corresponds to the OTP version of the base system that has been patched. Note that if a development system is updated by other means than `otp_patch_apply`, the file `OTP_VERSION` can identify an incorrect OTP version.

No `OTP_VERSION` file is placed in a target system created by OTP tools, because one can easily create a target system where it is hard to even determine the

base OTP version. However, it is allowed to place such a file there if one knows the OTP version.

OTP Versions Table

The text file `<OTP source root>/otp_versions.table`, which is part of the source code, contains information about all OTP versions from OTP 17.0 up to the current OTP version. Each line contains information about application versions that are part of a specific OTP version, and has the following format:

```
<OtpVersion> : <ChangedAppVersions> #
```

`<OtpVersion>` has the format `OTP-<VSN>`, that is, the same as the git tag used to identify the source.

`<ChangedAppVersions>` and `<UnchangedAppVersions>` are space-separated lists of application versions and has the format `<application>-<vsn>`.

- `<ChangedAppVersions>` corresponds to changed applications with new version numbers in this OTP version.
- `<UnchangedAppVersions>` corresponds to unchanged application versions in this OTP version.

Both of them can be empty, but not at the same time. If `<ChangedAppVersions>` is empty, no changes have been made that change the build result of any application. This could, for example, be a pure bug fix of the build system. The order of lines is undefined. All white-space characters in this file are either space (character 32) or line-break (character 10).

By using ordinary UNIX tools like `sed` and `grep` one can easily find answers to various questions like:

- Which OTP versions are `kernel-3.0` part of?

  ```
  $ grep ' kernel-3\.0 ' otp_versions.table
  ```

- In which OTP version was `kernel-3.0` introduced?

  ```
  $ sed 's/#.*//;/ kernel-3\.0 /!d' otp_versions.table
  ```

The above commands give a bit more information than the exact answers, but adequate information when manually searching for answers to these questions.

Application Version

As of OTP 17.0 application versions use the same version scheme as the OTP version, except that application

versions never include the `-rc<N>` suffix. Also note that a major increment in an application version does not necessarily imply a major increment of the OTP version. This depends on whether the major change in the application is considered a major change for OTP as a whole or not.

Version Scheme

> **Change**
>
> The version scheme was changed as of OTP 17.0. A list of application versions used in OTP 17.0 is included at the end of this section.

Normally, a version is constructed as `<Major>.<Minor>.<Patch>`, where `<Major>` is the most significant part. However, versions with more than three dot-separated parts are possible.

The dot-separated parts consist of non-negative integers. If all parts less significant than `<Minor>` equals `0`, they are omitted. The three normal parts `<Major>.<Minor>.<Patch>` are changed as follows:

- `<Major>` - Increases when major changes, including

incompatibilities, are made.

- `<Minor>` - Increases when new functionality is added.
- `<Patch>` - Increases when pure bug fixes are made.

When a part in the version number increases, all less significant parts are set to `0`.

An application version or an OTP version identifies source code versions. That is, it implies nothing about how the application or OTP has been built.

Order of Versions

Version numbers in general are only partially ordered. However, normal version numbers (with three parts) as of OTP 17.0 have a total or linear order. This applies both to normal OTP versions and normal application versions.

When comparing two version numbers with a defined order, one compares each part as standard integers, starting from the most significant part and moving towards the less significant parts. The order is determined by the first parts of the same significance that differ. A larger OTP version encompasses all changes present in a smaller OTP version. The same principle applies to application versions.

Versions can have more than three parts, resulting in partial ordering. Such versions are only used when branching off from another branch. When an extra part (apart from the normal three parts) is added to a version number, a new branch of versions is made. The new branch has a linear order against the base version. However, versions on different branches have no order, and therefore one can only conclude that they all include what is included in their closest common ancestor. When branching multiple times from the same base version, `0` parts are added between the base version and the least significant `1` part until a unique version is found. Versions that have an order can be compared as described in the previous paragraph.

An example of branched versions: The version `6.0.2.1` is a branched version from the base version `6.0.2`. Versions of the form `6.0.2.<X>` can be compared with normal versions smaller than or equal to `6.0.2`, and other versions on the form `6.0.2.<X>`. The version `6.0.2.1` will include all changes in `6.0.2`. However, `6.0.3` will most likely *not* include all changes in `6.0.2.1` (note that these versions have no order). A second branched version from the base version `6.0.2` will be version `6.0.2.0.1`, and a third branched version will be `6.0.2.0.0.1`.

Releases and Patches

When a new OTP release is released it will have an OTP version on the form `<Major>.0` where the major OTP version number equals the release number. The major version number is increased one step since the last major version. All other OTP versions with the same major OTP version number are patches on that OTP release.

Patches are either released as maintenance patch packages or emergency patch packages. The only difference is that maintenance patch packages are planned and usually contain more changes than emergency patch packages. Emergency patch packages are released to solve one or more specific issues when such are discovered.

The release of a maintenance patch package usually imply an increase of the OTP `<Minor>` version, while the release of an emergency patch package usually imply an increase of the OTP `<Patch>` version. However, this is not always the case, as changes in OTP versions are determined by actual code modifications rather than whether the patch was planned or not. For more information see Version Scheme.

OTP Versions Tree

All released OTP versions can be found in the OTP Versions Tree, which is automatically updated whenever we release a new OTP version. Note that each version number explicitly determines its position in the version tree. All that is required to build the tree are the version numbers themselves.

The root of the tree is OTP version 17.0 which is when we introduced the new version scheme. The green versions are normal versions released on the main track. Old OTP releases will be maintained for a while on `maint` branches that have branched off from the main track. Old `maint` branches always branch off from the main track when the next OTP release is introduced into the main track. Versions on these old `maint` branches are marked blue.

Apart from the green and blue versions, there are also gray versions. These denote versions established on branches to resolve a particular issue for a specific customer based on a specific base version. Branches with gray versions will typically become dead ends very quickly if not immediately.

OTP 17.0 Application Versions

The following list details the application versions that were part of OTP 17.0.

If the normal part of an application version number is smaller than the corresponding application version in the list, the version number does not adhere to the versioning scheme introduced in OTP 17.0. Consequently, it is not regarded as having an order against versions used from OTP 17.0 onwards.

- `asn1-3.0`
- `common_test-1.8`
- `compiler-5.0`
- `cosEvent-2.1.15`
- `cosEventDomain-1.1.14`
- `cosFileTransfer-1.1.16`
- `cosNotification-1.1.21`
- `cosProperty-1.1.17`
- `cosTime-1.1.14`
- `cosTransactions-1.2.14`
- `crypto-3.3`
- `debugger-4.0`
- `dialyzer-2.7`
- `diameter-1.6`
- `edoc-0.7.13`
- `eldap-1.0.3`
- `erl_docgen-0.3.5`

- `erl_interface-3.7.16`
- `erts-6.0`
- `et-1.5`
- `eunit-2.2.7`
- `gs-1.5.16`
- `hipe-3.10.3`
- `ic-4.3.5`
- `inets-5.10`
- `jinterface-1.5.9`
- `kernel-3.0`
- `megaco-3.17.1`
- `mnesia-4.12`
- `observer-2.0`
- `odbc-2.10.20`
- `orber-3.6.27`
- `os_mon-2.2.15`
- `ose-1.0`
- `otp_mibs-1.0.9`
- `parsetools-2.0.11`
- `percept-0.8.9`
- `public_key-0.22`
- `reltool-0.6.5`
- `runtime_tools-1.8.14`
- `sasl-2.4`
- `snmp-4.25.1`
- `ssh-3.0.1`
- `ssl-5.3.4`
- `stdlib-2.0`
- `syntax_tools-1.6.14`
- `test_server-3.7`
- `tools-2.6.14`

- `typer-0.9.6`
- `webtool-0.8.10`
- `wx-1.2`
- `xmerl-1.3.7`