



Version format ``<V>.<S>.<B>.<R>`` where V for major version, S for servicing number, B for build number, and R is the build revision number.  
Explain what are "major version", "servicing number", "build number", and "build revision number"



In software development, the process of versioning is critical for organizing and keeping track of changes to the software. It also provides users with an understanding of how the software has evolved.



Given the format you've provided (``<V>.<S>.<B>.<R>``), here's what each segment typically signifies:

- Major Version (``V``):** This is the most significant number in the versioning schema. It changes when there are significant jumps in functionality or a rehaul of the software that changes its operation fundamentally. For instance, a change in the major version number might represent a complete redesign of the software's user interface, or the addition of a set of features that drastically change how users interact with the software.
- Servicing Number (``S``):** Also often called the "minor version", it typically indicates that new features have been added or significant bugs have been fixed since the last major version. The changes are substantial but not enough to warrant a change in the major version number. For instance, a minor version update might include the addition of new functionalities that don't alter the overall operation of the software.
- Build Number (``B``):** The build number is usually incremented with every release that is made for testing or for release to customers. This might include small feature enhancements, bug fixes, or other incremental changes that don't significantly change the overall functionality of the software.
- Revision Number (``R``):** This is often used for very minor changes, often related to specific bug fixes or minor modifications. Sometimes, it's used to represent the number of commits or changes since the last build. This number can also be used to reflect patches, hotfixes, or other small changes that are not substantial enough to warrant a new build number.

Please note that these are general interpretations, and the exact definitions can vary depending on the organization or the software development team.



How to decide that I should increase a build number or revision number?



The decision to increase a build number versus a revision number depends on the type and scope of changes made to the software. However, specific criteria can vary depending on the project's practices, or the standards followed by your organization or team. Here is a general guideline:



**Increase the Build Number (``B``):**

- You've made significant code changes such as adding new features, enhancing existing features, or removing obsolete ones.
- There are substantial bug fixes or performance improvements that affect multiple areas of the software.
- Any change that results in a new release that you want to distribute to your users, testers, or stakeholders. This often indicates a milestone in the software development process and can be tied to specific objectives for a development sprint or phase.

**Increase the Revision Number (``R``):**

- There are minor bug fixes, security patches, or hotfixes that don't significantly change the software's functionality.
- You've made small changes to the software that aren't visible to the end user, such as code cleanup or documentation improvements.
- Any change that is not substantial enough to warrant a new build. Often, these are changes that might be essential but are relatively inconsequential in the grand scheme of the application's evolution.

Ultimately, it's about effectively communicating the scale and impact of the changes in your software. If the update is minor and does not significantly affect the software's functionality, a revision number increase is probably sufficient. If the changes are more substantial, or you're planning to release a new version of the software, increasing the build number is more appropriate. The key is to maintain a consistent strategy that aligns with your team's goals and makes sense to your users.



How to decide that I should increase a servicing number or build number?



Deciding whether to increment the servicing number (often referred to as the "minor version") or the build number depends on the nature and impact of the changes made in the software. However, the exact criteria can differ based on the project or the standards followed by your organization or team. Here are some general guidelines:



**Increase the Servicing Number (``S``):**

- Addition of new features that are substantial but not game-changing to the extent that it requires a major version increment.
- Significant changes or enhancements are made to existing features but the overall functionality and user experience of the software remain largely the same.
- Significant bug fixes or performance improvements that impact many areas of the software.
- Any change that is backward-compatible, meaning it doesn't break the software's compatibility with its previous version.

**Increase the Build Number (``B``):**

- A new release of the software is made available to testers or customers. This might include small feature enhancements, bug fixes, or other incremental changes that don't significantly alter the overall functionality of the software.
- There are smaller, iterative improvements or minor bug fixes that don't warrant an increment in the servicing or major version number.
- Non-functional changes are made, such as improving the efficiency of the code, refactoring, improving error handling, etc.

Again, these guidelines can vary and should be tailored to fit your project's needs and your team's versioning strategy. The key objective is to effectively communicate the scale and impact of the changes in your software to the end-users and stakeholders. If you are using semantic versioning, the definitions may be stricter.