



DIAVOLO V3

Versions numbering scheme



This document will discuss Diavolo versions numbering scheme in order to provide a consistent way to handle versioning and compatibility. This could provide a key feature given the higher number of sub-system and external components Diavolo features when compared to Rainbow Portal.

Main Diavolo version

To determine version of your portal you should always look at main *Diavolo.Core* assembly because that always represent version of core portal, the very same way *Rainbow.dll* assembly held main Rainbow Portal version. To determine current portal version, Diavolo code will always try to load a type from that assembly and then check its version. However, things starts to be more difficult when you look at sub-systems as Rainbow Portal only had one main assembly while Diavolo can have many sub-systems which are external to main core.

Versions scheme is the same that Rainbow Portal used and which is common to assembly versioning:

MAJOR.MINOR.BUILD.REVISION

where all parts are named according to their functionality. It must be noted that REVISION number is what will be used by Diavolo to identify release version, that is current portal version. This scheme is the same Rainbow Portal used.

First Diavolo version will be 3.0.0.1882 because last official Rainbow Portal version is (to date) 2.0.0.1881.

As you probably noted, release number (i.e. *revision*) is progressive and starts from 1882. There hasn't been any 3.0.0.1 release.

Other parts of version number are currently unused and won't be used by code. Only revision number will be used to manage updates.

Sub-systems versioning

In order to limit chances to mix incompatible versions of various sub-systems, it's better to develop a scheme to number different versions to help users to identify compatible components. Given *Diavolo.Core* assembly version numbering, sub-systems will use a similar numbering but BUILD number will be used to set minimum compatible Core version while release number will be used by sub-system developers. So, for example, users management sub-system can have a version number like this:

1.0.1883.5



Major and minor version number will be set according to developers needs, so the aforementioned number will mark v1.0 of that component. Build number (1883) will mark the minimum required Core version, that is at least 3.0.0.1883 will be required to run that component. Notice that this scheme won't be enforced by main code itself so nothing will prevent portal administrators to drop v1.0.1883.5 while still using Diavolo 3.0.0.1882, for example.

We believe that enforcing that version compatibility could be premature since this scheme hasn't been tested enough to be considered as flexible enough and thus enforcing that scheme in code could result in future incompatibilities: we'd like to avoid that at this stage.

However, do notice that nothing prevents component itself, during initialization for example, to perform that check and raise a *DiavoloIncompatibleException* if a non-compatible version will be found. In the example shown above, users management component could raise an exception if Core v3.0.0.1882 will be found during initialization. That behavior is definitely recommended.

Finally, revision number (5 in our last example), is left available to developers to mark minor changes to their components.

Reason why we won't use revision number to match Core minimum required release number is we would like to avoid confusion. Since no actual checks for version compatibility will be enforced into code, that kind of numbering would make users believe that Core release number and component release number must match while this is not the case. That would also make it more difficult for developers to mark minor changes to their components.

By using build number, we can avoid confusion and make it easier to understand if a newer version of the same component has been released by developers.

Major and minor version matching

We evaluated the possibility of suggesting a match of major and minor version numbers of Core and components, that is releasing a component and mark its version as v3.0.1882.5 in order to let users know that component has been developed for Core v3.0. However, we believe that's not necessary as our scheme, as exposed in previous paragraphs, is flexible enough to mark suggested compatible version without the need to prevent developers from using major / minor version numbers for their own purposes. Build number matched to Core release number will provide enough flexibility and it's easy to understand and developers will be free to mark most-relevant changes to their code by using such numbers.



Version checking of components

In a future release, we plan to introduce a way to let Diavolo check if components added to an installation are compatible with core itself and between each other.

Such mechanism will be probably implemented as a manifest object to be embedded inside component itself or some other kind of public mechanism in order to let Diavolo check such objects during initialization.

While we're making changes to update system, we consider premature to embed such mechanism inside portal core now. Following these versions numbering guidelines will probably ensure that no major hassle will occur when such system will be implemented. We cannot exclude that some changes will be needed as system could evolve and need a more formal mechanism for compatibility among components. However, while that cannot be excluded, we don't anticipate that such changing will be needed as we would like to keep that area very simple and easy to manage, especially for users and non-developers administrators.

We don't expect such system to be implemented earlier than 20-25 releases because we want to be sure that once implemented, it won't be changed too soon and it will be flexible enough to provide a mechanism for a safer experience rather than a difficult stuff which would slow development.