SCMFViz manual

Document Identification	SCMFViz manual
Document Version	0.1.0
Document Type	Manual
Project Name	Software Configuration Management Framework Visualizer
Project Identification	SCMFViz
Document Owner	Sergii Shmarkatiuk

1. Contents

2. Application description

The main goal of **SCMFViz** (Software Configuration Management Framework Visualizer) application is to:

2.1. Graphically represent main principles and concepts of the Software Configuration Management Framework.

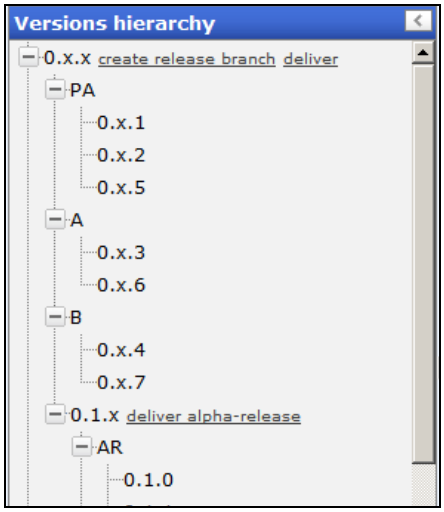
2.2. Allow user to see SCMF principles in action.

2.3. Automate manual tasks and configuration management activities (branches and tags creation, version-platforms mapping, etc).

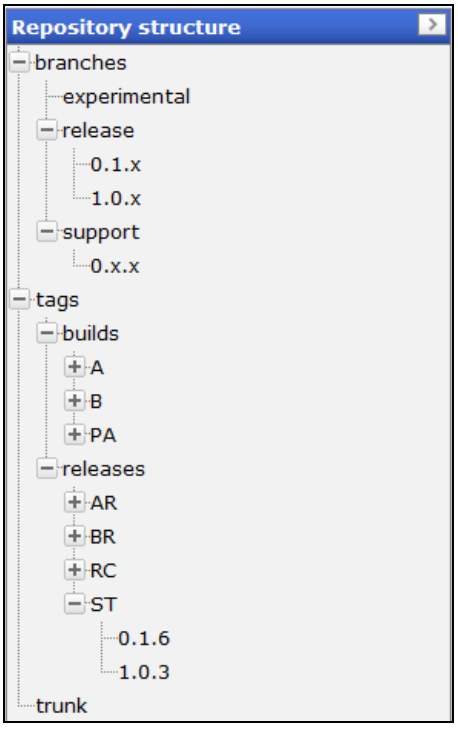
3. Conventions

- **SCMF** (Software Configuration Management Framework) – set of Software Configuration Management conventions, principles, concepts and practices used as a basis for the SCMFViz application features and functionality.

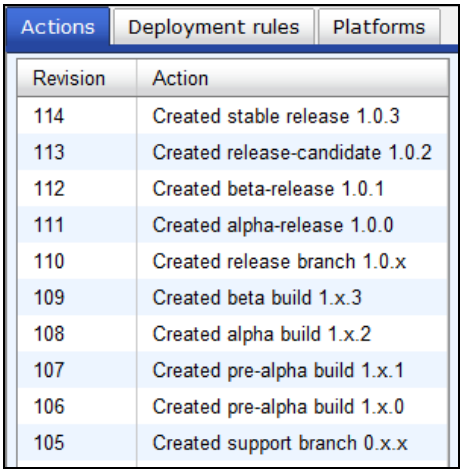
- **Versions hierarchy view** – part of the SCMFViz interface used for the purpose of displaying versions as a hierarchical structure without regard to repository structure. *Versions hierarchy view* is the simplified representation of streamline diagrams.



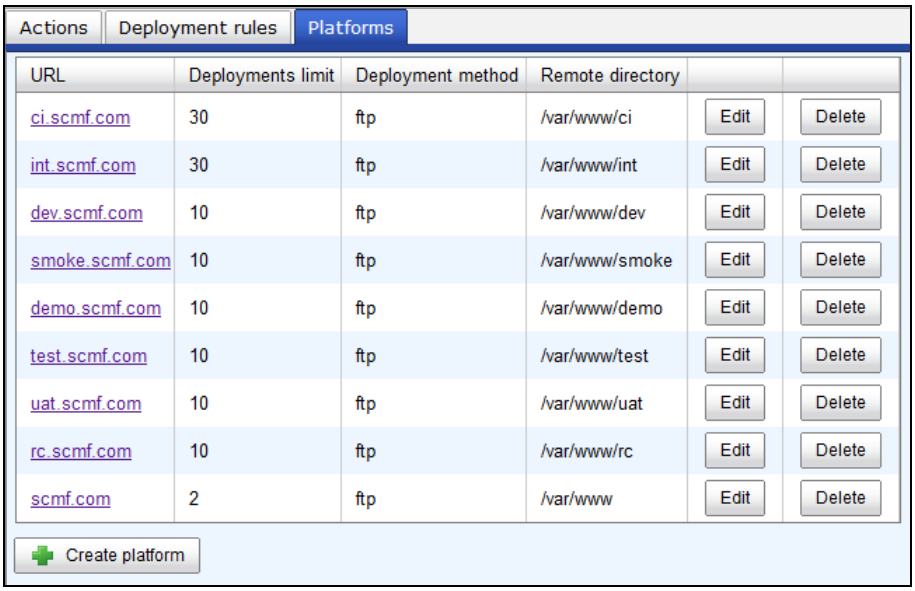
- **Repository structure view** – part of the SCMFViz interface used for the purpose of displaying repository structure and content of such repository directories as *branches* and *tags*.



- **Actions view** – part of the SCMFViz interface displaying sequence of actions performed using SCMFViz functionality.



- **Platforms view** – part of the SCMFViz interface which allows viewing and managing target deployment platforms and their properties

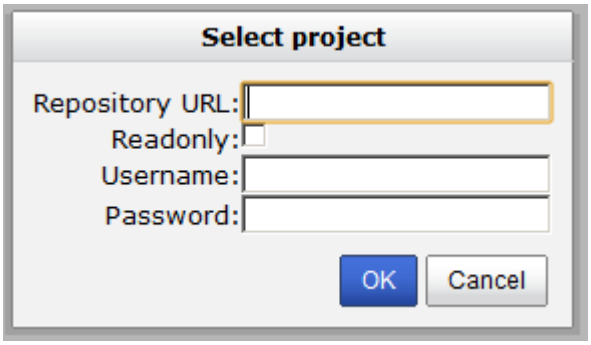


- **Deployment rules view** – part of the SCMFViz interface which allows viewing and managing deployment rules

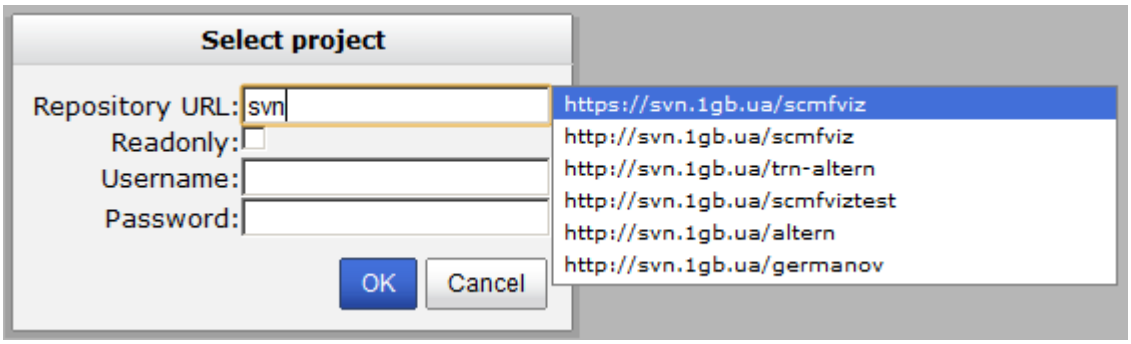
- **Version reference** – link to SCMFViz application (readonly access) from other applications (developed according to SCMF principles and using SCMF version numbering approach):

4. Repository selection

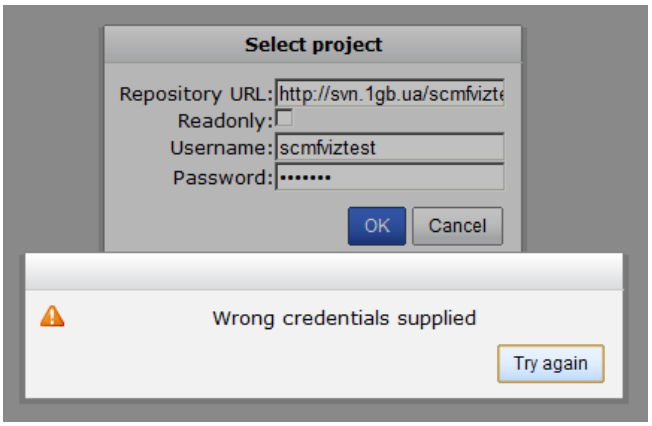
In order to start working with SCMFViz, you need to select version control system repository to work with using ‘Select project’ dialog:



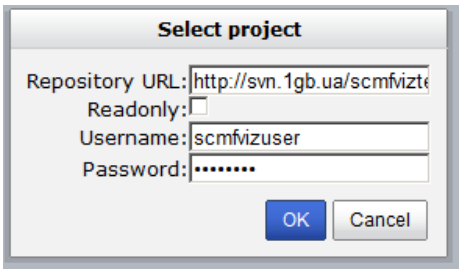
Once you start typing repository address into the 'Repository URL' field, several suggestions appear. Suggestions include URLs of repositories you selected before (stored in cookies) and URLs of repositories stored in a form of a project in SCMFViz database:

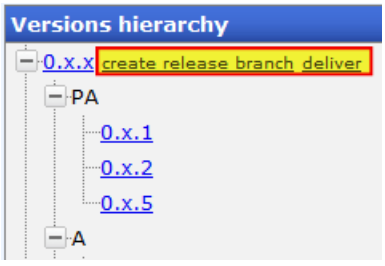


In order to be able to perform actions that include repository structure changes, you need to specify credentials for the VCS account having access with writable permissions. If you have supplied incorrect credentials or user has readonly access, following message will appear:

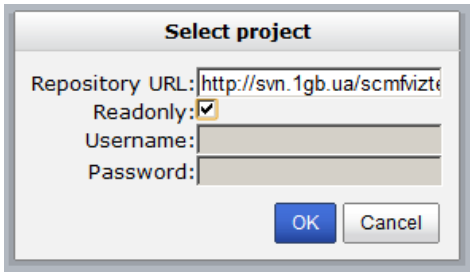


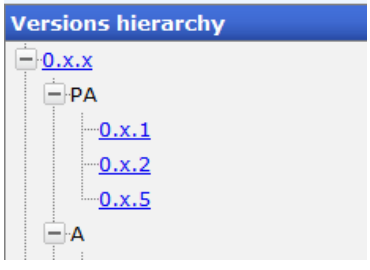
After you have entered correct credentials, you will be able to perform repository structure changes using SCMFViz functionality:





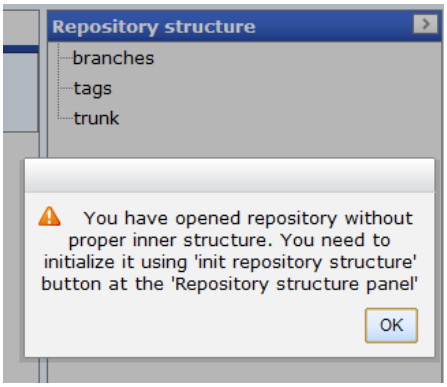
If 'Readonly' checkbox has been selected, SCMFViz will not be able to perform repository structure changes:





5. Initializing repository structure

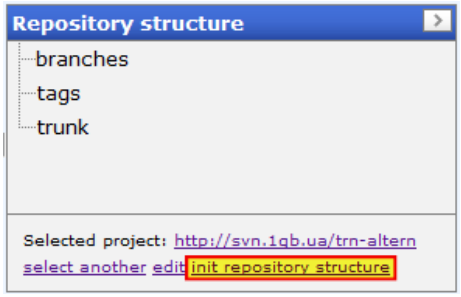
If you have selected repository without proper inner structure (according to SCMF), following message will appear:



It is supposed that repository has following initial directories structure:

```
/trunk
/tags
        /builds
                /PA
                /A
                /B
        /releases
                /AR
                /BR
                /RC
                /ST
/branches
        /experimental
        /support
        /release
```

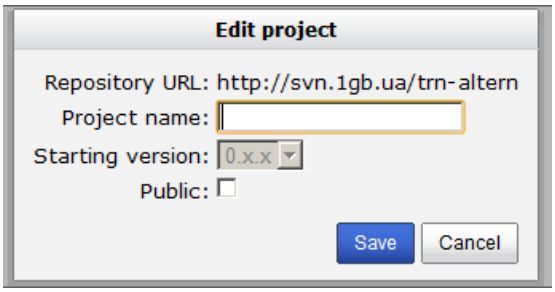
In the case you want to initialize repository structure using SCMFViz, press ‘init repository structure’ link at the *repository structure view*:



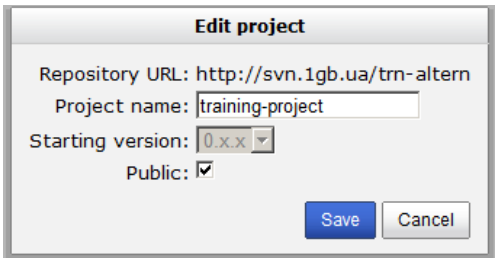
Properly initialized repository looks as follows:

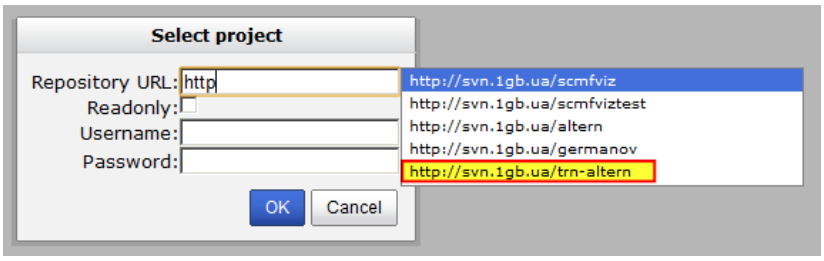
6. Editing project properties

Once you have opened repository in SCMFViz, you can save it as a project. You need to press ‘edit’ link at the *repository structure view:*



If you selected ‘Public’ checkbox before pressing ‘Save’, repository URL will appear in a suggestions list on the select project dialog:



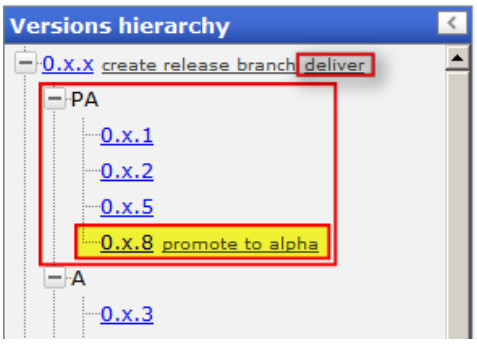


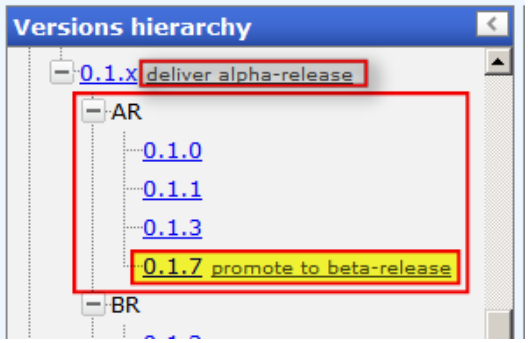
7. Repository structure changes

After you have selected version control system repository to work with and it has been checked that it has proper directories structure, you are able to perform repository structure changes: **deliver new version** or **create branch**. While **delivering new version** is associated with tagging (and creating new directory in `/tags/` directory), branch creation is associated with branching (and creating new directory in `/branches/` directory)

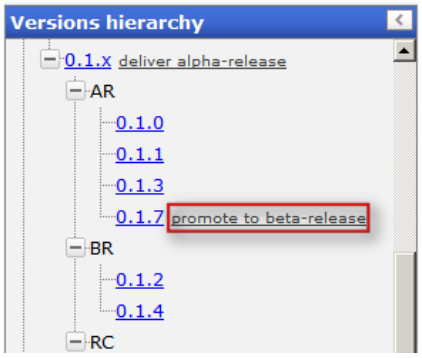
7.1. Delivering new version

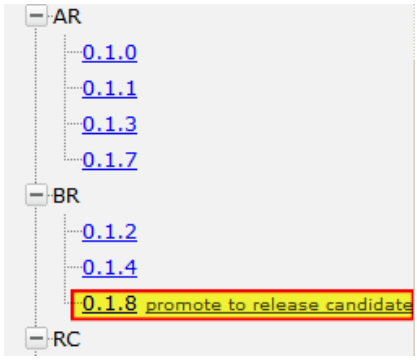
New version is considered to be delivered if corresponding subdirectory of `/tags/builds/PA` or `/tags/releases/AR` directory has been created:

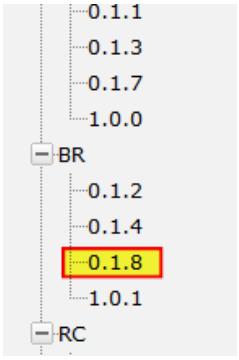




Then version can be promoted to another maturity level (`PA -> A -> B, AR -> BR -> RC -> ST`). Corresponding repository directories will be created:



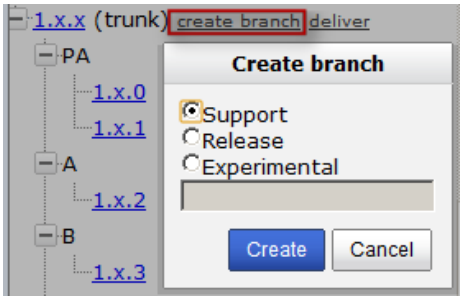




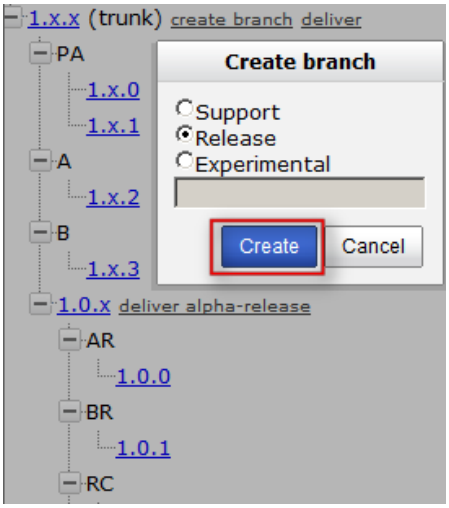
When version is being delivered/promoted, tagging is performed using corresponding branch codebase. For example, if version `0.1.7` is being delivered, latest state of branch `0.1.x` is to be tagged using following command: `svn copy /branches/releases/0.1.x /tags/releases/BR/0.1.8`

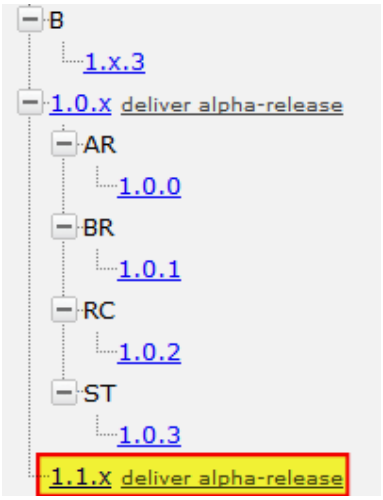
7.2. Branch creation

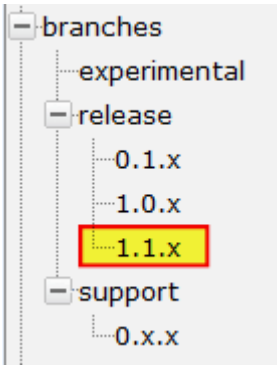
In general, it is possible to create branches of three main types: **support**, **release** and **experimental**:

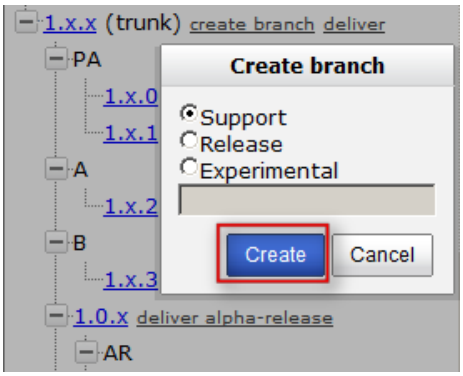


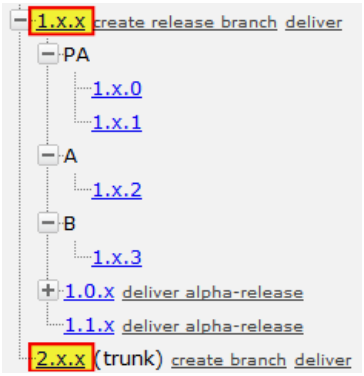
Different types of branches are being created in different repository directories:

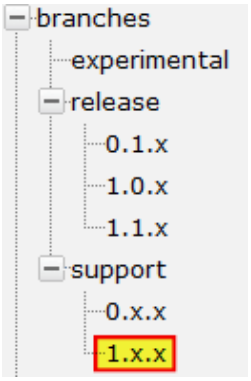


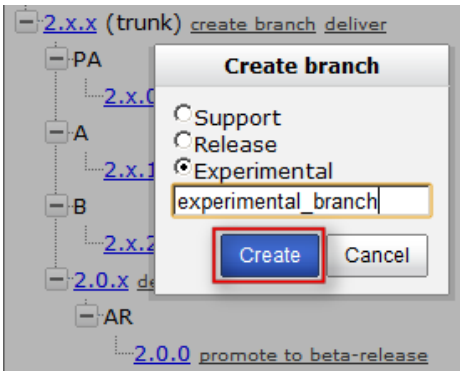


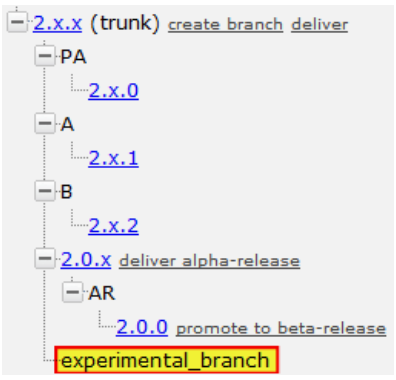


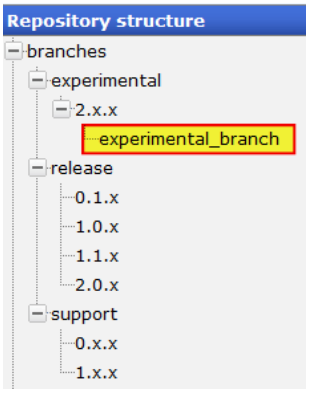






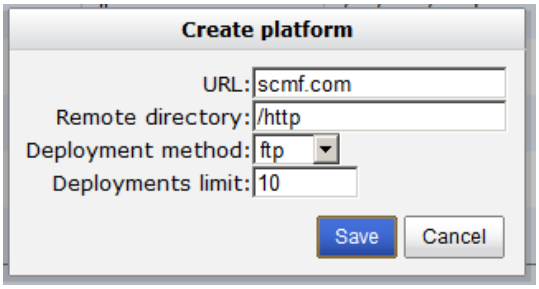






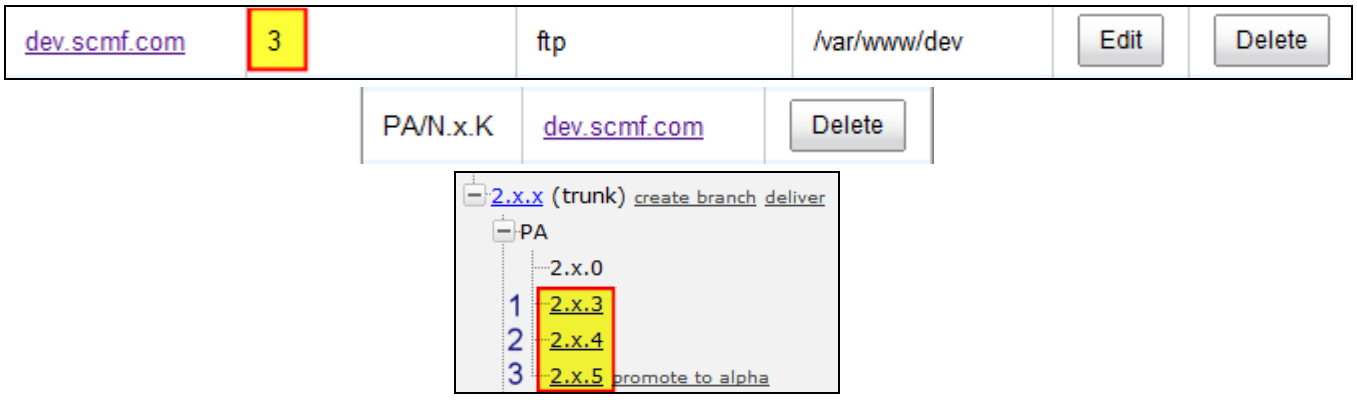
8. Adding platform

Platforms are used for the purpose of storing information about places where application artifacts can be deployed. In order to add platform, you need to press 'Create new platform' at the *platforms view.* You will see following dialog:



'Deployments limit' field is used for entering maximum number of deployments which will be stored at the platform. For example, if 'deployments limit' property is set to 2 for platform platform1 which is mapped to the `PA/N.M.K` version set and there are three PA versions `1.x.0`, `1.x.3` and `1.x.4`, there will be only two latest versions stored at the target platform1: `1.x.3` and `1.x.4`.

Another example of deployments limit usage:



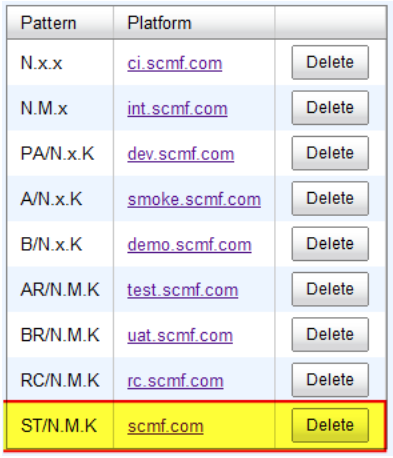
9. Adding deployment rule

Deployment rules are used for mapping specific *version sets* to platforms. In general, there are 9 version sets referenced by following patterns:

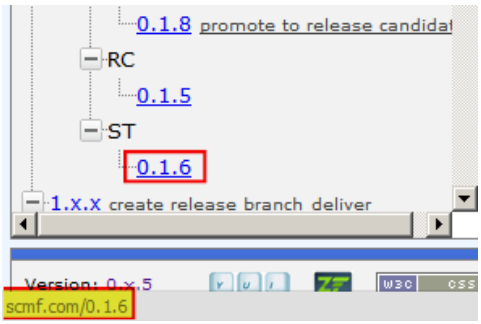
```
N.x.x
N.M.x
PA/N.M.K
A/N.M.K
B/N.M.K
AR/N.M.K
BR/N.M.K
RC/N.M.K
ST/N.M.K
```

In order to create deployment rule, you need to press ‘Create deployment rule’ button at the *deployment rules view*. You will see following dialog:

After pressing ‘Save’ button rule will become available at the deployment rules list at deployment rules view:

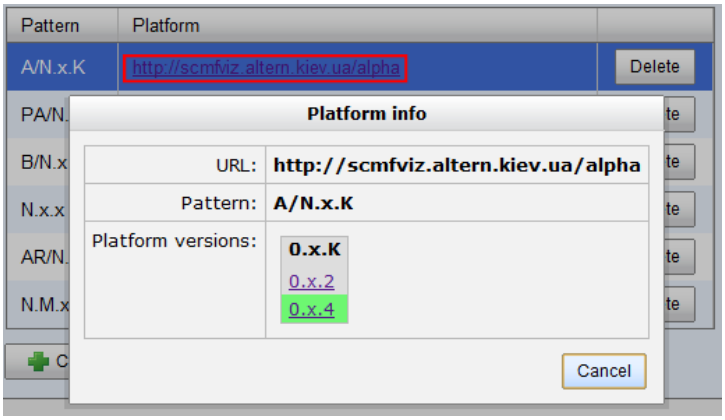


If version set is mapped to the platform, corresponding links become available at the *version hierarchy view*:

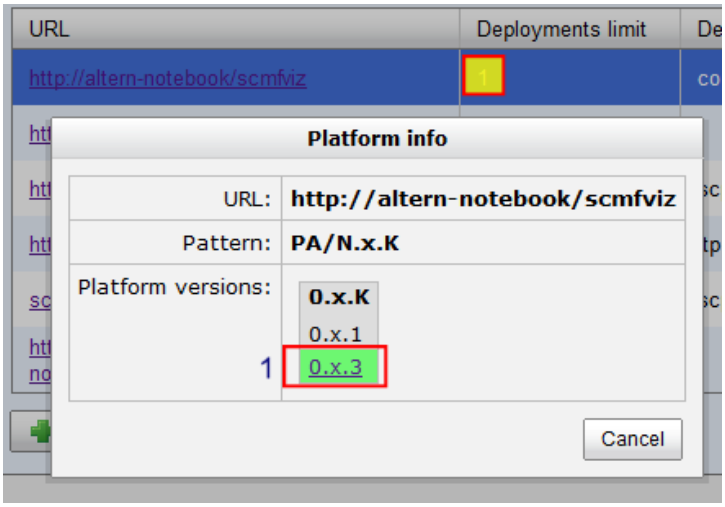


10. Platform information

If you will click on a platform link either from *deployment rules view* or *platforms view*, you will see ‘Platform info’ dialog containing information about versions deployed to the platform you have clicked on:

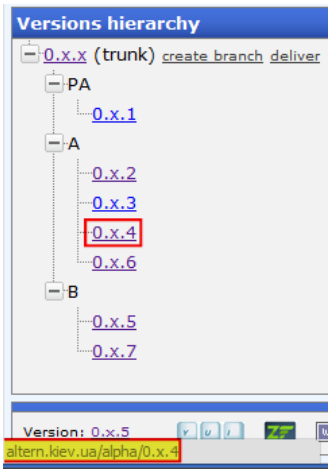


Latest version deployed to the platform is highlighted with green. If version fits into the deployment limit for platform, it will appear as a link:

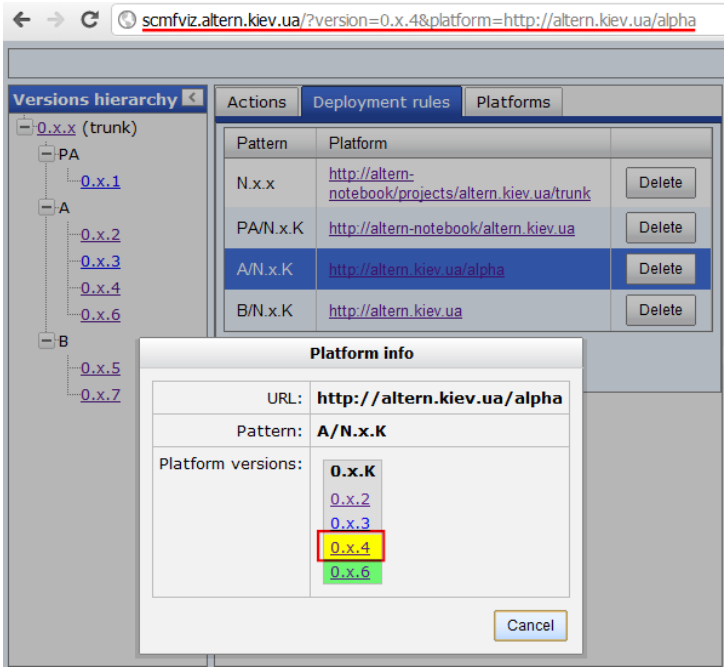


11. Version reference

It is supposed that every application which is being developed according to SCMF principles and practices has *version reference* in the form of a link to SCMFViz application. Every such link is formed in a way that allows referencing currently deployed version of the application by its version number and platform where it is actually deployed.



After you click on the *version reference*, SCMFViz application will be opened in a readonly mode. Referenced version will be highlighted with pulsating yellow in an opened ‘platform info’ dialog.



12. Goals of SCMFViz 0.1.x release

1. Main goal of SCMFViz 0.1.x release is to implement *version reference* functionality in order to be able to embed link to SCMFViz application into other applications (developed according to SCMF practices and principles).
2. Implement *versions hierarchy view* according to SCMF.
3. Implement functionality of repository changing actions according to SCMF (deliver, promote, create branch, etc).
4. Illustrate SCMF principles and concepts (versions numbering approach and repository structuring approach) visually.

13. Known issues of SCMFViz 0.1.x

1. Deployment rules can be mapped only in a way that platform has single mapping rule. It will be fixed in future versions.
2. There is a problem with escaping parentheses in commit messages. It might cause action records to disappear from *actions view* or render it with extra slashes. It will be fixed in future versions.
3. It is impossible to edit commit messages at the repositories currently used by SCMFViz for testing purposes. Actions list might contain (or lack) some records, which, in turn, might affect whether versions history is displayed properly in *actions view*.
4. Deployments should be performed manually.
5. There some properties and fields that are not used by SCMFViz functionality:
    a. ‘Deployment method’ field (platform properties)
    b. ‘Remote directory’ field (platform properties)
    c. ‘Starting version’ field (project properties)

    Those properties will be used when requirement of performing application deployment automatically will be implemented in future versions.

14. Features to be included into the future SCMFViz releases

14.1. Short-term plans

1. Deployment rules
    1.1. Possibility of mapping several version patterns to the same platform.
    1.2. Possibility of splitting pattern into the version subsets. For example, splitting `N.M.x` into `N.0.x`, `N.1.x`, `N.2.x`, etc.
2. Checking whether there were commits to the parent branch on version promotion. If yes, then increment version number. If not – leave it the same.

3. Make it possible to promote to any maturity level (with minor restrictions). Currently it is possible to promote only from previous maturity level.
4. Rewrite manual using some markup language (pandoc, Markdown, etc).
5. Provide inline description of maturity levels (via pop-ups and tips) and their primary goals.

14.2. Long-term plans

1. DVCS integration (git/mercurial).
2. Visualizing continuous integration builds according to SCMF.
3. Perform automated deployments and continuous delivery.
4. Implement requirements mapping and version numbering suggestions according to SCMF
5. Merge management functionality and visualization
6. Manage software quality using SCMFViz, develop recommendations in order to improve software quality during different SDLC stages and map it to maturity levels.
7. Provide dependencies management, binaries management, etc.
8. Provide list of SCM tasks for Project Managers in order to properly allocate time and resources.