Configuration Management Process

SOC 2 Document

esm_logo.jpg

Nexelus USA  
New York, NEW YORK (NY)  
646-558-1950 ext.128

**11/10/2022**

# Document Information

The following table shows the details for document creation, review, approval, and effective date.

| **Category** | **Information** |
| --- | --- |
| Work Product: | Configuration Management Process – SOC 2 Document |
| Project Name: | Configuration Management Process |
| Function Name: | SOC 2 Document |
| Version: | 1.2 |
| Status: | Draft |
| Author(s): | Tauseef Shahzad |
| Reviewer(s): | Asim Jameel |
| Approver(s): | Imran Rahman |
| Control Status: | CONTROLLED, PROTECTED |
| Disclaimer: | This document contains confidential information. Do not distribute this document without prior approval from Nexelus. |

# Revision History

The following table is used for revision details of this document.

| **Author(s)** | **Date** | **Version** | **Description of Change** |
| --- | --- | --- | --- |
| Tauseef Shahzad | October 10, 2021 | 1\_0 | Initial Draft |
| Tauseef Shahzad | October 31, 2022 | 1\_1 | Added description for Major and Minor releases |
| Tauseef Shahzad | November 10, 2022 | 1\_2 | Added Process for Production Rollout |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

Contents

[Document Information ii](#_Toc118970890)

[Revision History iii](#_Toc118970891)

[Table of Contents i](#_Toc118970892)

[Scope 1](#_Toc118970893)

[Configuration Management 2](#_Toc118970894)

[Software Configuration Management 3](#_Toc118970895)

[Configuration management in a development cycle 3](#_Toc118970896)

[Release Backlog and Issue Tracking 5](#_Toc118970897)

[Tasks in development and maintenance 5](#_Toc118970898)

[Configuration Identification 7](#_Toc118970899)

[Identification rules of configuration items 7](#_Toc118970900)

[Identification rules of documents 7](#_Toc118970901)

[Process for Production Rollout 8](#_Toc118970902)

[Pre-Requisites: 8](#_Toc118970903)

[Package Deployment at Client-UAT Environment: 10](#_Toc118970904)

[Package Deployment at Production Environment: 10](#_Toc118970905)

[Incremental Upgrades: 11](#_Toc118970906)

# Scope

Software Configuration Management(SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. The primary goal is to increase productivity with minimal mistakes. SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

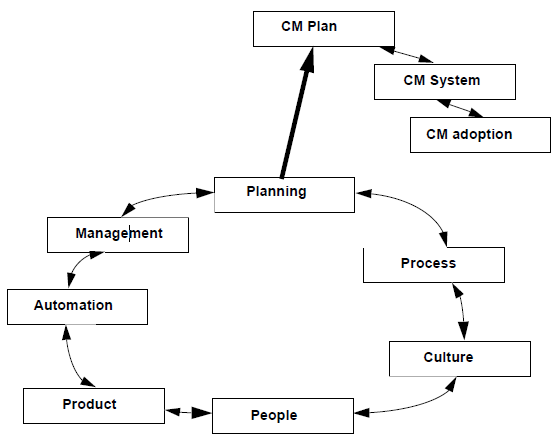
The scope of this document is to document Software Configuration Management Process as implemented for Nexelus Software Development.

**Reference**

SSAE-18 SOC 1 Type II – Requirements

# Configuration Management

There are ten elements which are the keys to solving the CM needs in an organization. Seven of these elements relate to the problem preparation and solving work and the other three are the results of seven element. All 10 elements are shown in Figure below:



## Software Configuration Management

At Nexelus, the software configuration management is done with Team Foundation Server, a SCM tool that integrates with Microsoft Visual Studio as well as provides a web interface/ Team Foundation Server provides Source Code Management as well as acts as primary bug tracking and task management tool.

For each project a root branch is created with template “\_<ProjectID>Trunk” as main development branch, called the Master, receives by default all software developments made by the software team. When a new major version is planned (for instance V1.0 or V2.0), a branch is created from the master. This branch is isolated to be tested, fixed, and finally delivered. A Minor branch is assigned an incremental decimal value such as (1.1, 1.2, …).

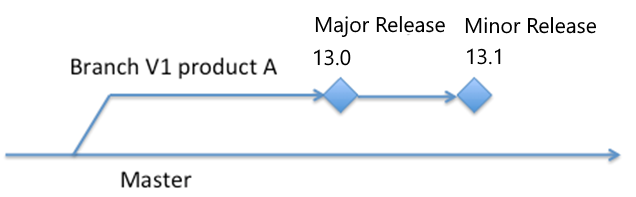


Figure Master and branches in Team Foundation Server

### Configuration management in a development cycle

The changes made by developers during a development cycle are managed by the following method.

Once the work on a branch is complete it is marked as “Read Only” and merged into Master branch.

Two new branches are created from Master branch:

1. New branch with new release version. All new development is maintained in new branch.
2. Hot Fix Branch with existing release ID and Hotfix prefix. All production issues related to current deployed release are checked into this branch.

Periodically the Hotfix branch is merged into Master branch and then from Master branch, merged into new Development Branch.

#### Major Upgrade: All new functionality

• Convention: 13.0.0.0

• Timing: Typically, every 7-12 Months based on Product Road Map

• Test Upgrade: Yes, with enough time given to clients to verify and plan new functionality usage before production upgrade

#### Minor Upgrade: standalone product enhancements/incremental changes

• Convention: 13.1.0.0

• Timing: Typically, 2-6 Months or based on Product Road Map

• Test Upgrade: Yes, with enough time given to clients to verify and plan new functionality usage before production upgrade

#### Service Pack: Bug fixes, along with minor enhancements which need to be deployed to prevent any data or user issue

• Convention: 13.1.1.0

• Timing: As needed based on Support priorities and Impact

• Test Upgrade: Yes, with enough time given to clients to verify the upgrade before production upgrade

•

#### Hot-Patch: Minor Enhancements and fixes which need to be deployed

• Convention: 13.1.1.1h

• “h” is added at end of release number for **Hot Patch**

• Timing: As needed based on Support priorities and Impact

#### Urgent Patch: Critical/Urgent Bug fixes which need to be deployed

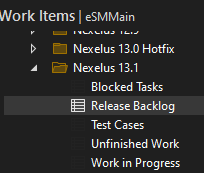
• Convention: 13.1.1.1u

• “u” is added at end of release number for **Urgent Items**

• Timing: As needed based on Support priorities and Impact

Note: Deployment process where production and UAT environments are on different versions, all Hot-Patch and Urgent-Patch will be provided separately for production and UAT environments with package naming conventions relevant to version deployed on each environment.

### Release Backlog and Issue Tracking



A new Iteration is created in TFS “Work Items” corresponding to Development release.

* All Backlog items are listed in this iteration status “New”.
* Once the Backlog item is assigned to a developer, it is marked as “approved”
* The developer marks the backlog item as Committed, after the domain Transfer session.
* The QA team starts entering test cases against Backlog items in parallel.
* When Developer starts development, he marks it as “In Progress”
* Once the Development is complete, the developer marks the Backlog item as “Review” for unit testing and review by Team Lead.
* After reviewing the backlog item, it is marked as Testing and assigned to QA Team.
* The QA Team tests the backlog item. If no issues are found, it is marked as “Demo”
* If some issues are found, the status is changed to “Bug Fixing” and assigned back to developer.

### Tasks in development and maintenance

The tasks depend on the phase of the software development project or of maintenance. The SCM Manager does the following operations, in the software life-cycle.

|  |  |  |
| --- | --- | --- |
| **Item** | **Event** | **Operation** |
| 1 | Launching the development of a new product | Creating the source folder structure in the master branch |
| 2 | Deciding to create a major version | Fork of a branch from the current state of the master branch |
| 3 | Releasing a major version | Tagging the current version in its branch.  Archiving the tagged version |
| 4 | Releasing a minor version or a patch | Adding a new tag to the current version in the branch.  Archiving the tagged version |
| 5 | Closing an iteration cycle | Adding a new tag to the current version in the master branch |

## Configuration Identification

### Identification rules of configuration items

The identification of configuration item is:

* <configuration item name>\_m.n

where:

* "m " is the major version of the configuration item,
* n is the minor version number,

The version number of the configuration item m.n starts at 1.0.

The number "m" of major version is incremented when substantial modifications are made to the device, for example:

* Updating of the intended use,
* Adding new modules / functionalities,

The number "n" of minor version is incremented when non-substantial modifications are made to the device, for example:

* Adding new functionalities to existing modules,
* Updating the GUI.

### Identification rules of documents

The identification of documents is described below:

XXX-<document number>\_<revision index>

where:

* XXX is an acronym to identify the project/module/functionality
* " document number " is a incremental in the project,
* " revision index " designates the approved iteration of the document. The revision index is 0 for draft version, 1 for the first approved revision, 2 for the second and so on.

## Process for Production Rollout

There are 3 Stages Deployment Process before GO-Live:

1. QA Environment

2. Release Environment

3. Client-UAT Environment

Note: NO DIRECT FIXING ON QA AND RELEASE ENVIRONMENT BY DEVELOPMENT TEAM; IN CASE OF ANY FIXES QA MANAGER/LEAD NEEDS TO GET LATEST PACKAGE FROM TFS AND EXECUTE IT ON REQUIRED DATABASES AND DEPLOY THE UPDATED WEB PACKAGE (IF REQUIRED).

### Pre-Requisites:

1. Every Development Project must contain a file named “Deployment Guide", which will include each configuration, rule and/or library changed or added.

[Responsible: Dev Manager]

2. Updated Web Package must be available at TFS and labeled with proper build number (e.g., Build # 12.0.0.0001) before handing over package to QA;

Package should be compiled on Build Server which is only accessible to Dev Manager/Lead

[Responsible: Dev Manager/Lead]

3. Updated DB Package must be available at TFS and labeled with proper build number (e.g., Build # 12.0.0.0001u) before handing over package to QA;

Package should be compiled on compilation machine [Responsible: Database Manager/Lead]

NOTE: Dev Manager/Lead and Database Manager/Lead Needs to generate an email at least a day before deployment that updated package along with the mentioned build number, deployment guide is updated in TFS

### Package Deployment at QA Environment:

1. QA needs to get the latest package/build (Web and DB) from TFS and make sure that all the pre-requisites got fulfilled. [Responsible: QA Manager/Lead]

In case of missing pre-requisite, QA Manager/Lead needs to raise the RED flag and inform CEO and CC to GM Development/Dev Manager and Database Manager and should not proceed with deployment until pre-requisites got fulfilled.

2. Continue with the Script Execution and/or Application Deployment [Responsible: QA Manager/Lead] and share the list of issues and reasons that surfaced during deployment.

3. Perform Process testing and share the list of issues and reasons that QA team encountered during process testing. [Responsible: QA Manager/Lead]

4. In case of issues/bugs; QA team needs to inform Dev team and get the issues fixed. [Dev Manager/Lead and Database Manager/Lead needs to provide all the fixes in updated package/build]

5. Once reported issues are fixed; QA Manager needs to get the latest package/build from TFS and then re-deploy the package with utility and perform his testing.

NOTE: No direct fixing on QA and Release environment by Development Team. If any fixes QA Manager/Lead needs to get latest package/build of dotnet and scripts from TFS and execute it on required databases. [Responsible: QA Manager/Lead]

6. Make sure to verify all the post deployment issues of previous release (Reported by client) must be intact in Current Released Version [Responsible: QA Manager/Lead]

7. For Incremental upgrades: For every build Database Manager/Lead and Dev Manager/Lead needs to provide a release document pertaining to the changes/fixes/additions/new items for that release.

For instance: We have deployed a release at client UAT which is supposed to be 12.0.0.0000 (if this is how it’s been maintained). Any subsequent releases (12.0.0.0001, 12.0.0.0002 etc.) will have the next number and a release note document and to be kept in TFS with script. [Responsible: Dev Manager/Database Manager/Lead]

8. Re-Do the deployment steps after every build [Responsible: QA Manager/Lead]

Package Deployment at Release Environment:

1. QA needs to get the latest package (Web and DB) from TFS and make sure that all the pre-requisites got fulfilled. [Responsible: QA Manager/Lead]

2. Continue with the Script Execution / and Application Deployment [Responsible: QA Manager/Lead]

and share the list of issues along with reason that QA Manager/Lead did face during deployment.

3. Perform Process testing and share the list of issues along with reason that QA team did face during process testing. [Responsible: QA Manager/Lead]

4. In case of issues/bugs; QA team needs to inform Dev team and get the issues fixed. [Dev Manager and Database Manager/Lead needs to provide all the fixes in updated package]

5. Once reported issues got fixed; QA Manager/Lead needs to get the latest package from TFS and then re-deploy the package with utility and perform his testing.

NOTE: no direct fixing on QA and Release Environment by Development team; in case of any fixes QA Manager/lead needs to get latest package from TFS and execute it on required databases. [Responsible: QA Manager/Lead]

6. Make sure to verify all the post deployment issues of previous release (Reported by client) must be intact in Current Released Version [Responsible: QA Manager/Lead]

### Package Deployment at Client-UAT Environment:

#### Pre-requisite:

Provided Package with the mentioned build number should be verified/executed by QA Manager/Lead on Release environment for selected clients before upgrading any client-UAT environment

Package should be error free and finalized by QA - [Responsible: QA Manager/Lead]

NOTE: Client-UAT upgrade should not be done until package got executed/verified on Release Environment

1. Release Manager needs to get the latest package with the provided build number (Web and DB) from TFS and make sure that all the pre-requisites got fulfilled. [Responsible: Release Manager];

NOTE: In case of missing pre-requisite; Release Manager needs to inform CEO and CC to QA Manager/Lead, Dev Manager/Lead and Database Manager/Lead and should not proceed with deployment until pre-requisites got fulfilled.

2. Continue with the Script Execution / and Application Deployment [Responsible: Release Manager]

and share the list of issues along with reason that occurred during deployment.

3. Perform Process testing and share the list of issues along with reason that occurred during process testing. [Responsible: Release Manager]

NOTE: QA resource can be involved if client have more than 2-3 companies.

4. In case of issues/bugs; Release Manager will move the issues to QA and then QA Manager/Lead will take a lead to re-visit the issues and get them fixed from dev team and inform Release Manager to get the latest package from TFS and re-deploy the package [Responsible: QA Manager/Lead]

5. On QA Manager/Lead’s confirmation; Release Manager will get the latest package from TFS and then re-deploy the package with utility and perform his testing [Responsible: Release Manager]

6. Make sure to verify all the post deployment issues of previous release (Reported by client) must be intact in Current Released Version [Responsible: Release Manager]

NOTE: Based on number of client-reported issues QA team member can be involved for its verification.

### Package Deployment at Production Environment:

#### Pre-requisite:

Final/Freeze Package with the mentioned build number should be deploy on Release environment and Client-UAT environment at least once so that we can make sure that we are deploying the error free and QA package at live environment- [Responsible: QA Manager]

1. Release Manager needs to get the latest package with the mentioned build number (Web and DB) from TFS and make sure that all the pre-requisites got fulfilled. [Responsible: Release Manager].

NOTE: In case of missing pre-requisite; Release Manager needs to inform CEO and CC to QA Manager/Lead, Dev Manager/Lead and Database Manager/Lead and should not proceed with deployment until pre-requisites got fulfilled.

2. Continue with the Script Execution / and Application Deployment [Responsible: Release Manager]

and share the list of issues along with reason that occurred during deployment.

3. Perform Process testing and share the list of issues along with reason that occurred during process testing. [Responsible: Release Manager]

NOTE: QA resource can be involved if client have more than 2-3 companies.

4. In case of issues/bugs; Release Manager will move the issues to QA and then QA Manager/Lead will take a lead to re-visit the issues and get them fixed from dev team and inform Release Manager to get the latest package from TFS and re-deploy the package [Responsible: QA Manager/Lead]

5. On QA Manager/Lead’s confirmation; Release Manager will get the latest package from TFS and then re-deploy the package with utility and perform his testing [Responsible: Release Manager]

NOTE: Post deployment issues of previous release (Reported by client) that we already verified on QA Environment, Release environment and Client-UAT environment are not part of testing during production upgrades.

### Incremental Upgrades:

#### Pre-requisite:

For every build dev/QA team need to provide a release document pertaining to the changes/fixes/additions/new items for that particular release/build. [Responsible: Dev Manager/Lead, Database Manager/Lead and QA Manager/Lead]

Same deployment process will follow for incremental upgrades (Summary is as follows);

1. Get latest package/build from TFS along with the above mentioned document [Responsible: QA Manager/Lead]

2. Deploy and verify the package/build on QA environment [Responsible: QA Manager/Lead]

3. Deploy and verify the package/build on Release environment [Responsible: QA Manager/Lead]

4. Deploy and verify the package/build on Client-UAT environment [Responsible: Release Manager]

NOTE:

1. Make sure before upgrading Client-UAT environment; first 2 stages of deployment must be upgraded with the same package/build and verified by QA.

2. Post deployment issues of previous release (Reported by client) that we already verified on QA Environment, Release environment and Client-UAT environment are not part of testing during incremental upgrades on CLIENT-UAT and Production environments.