|  |  |  |  |
| --- | --- | --- | --- |
| Abstract: | | | |
| This is the Configuration & Change Management Procedure applicable for MCAL/SMCAL projects configuration and change management software development and test development activities. | | | |
| Keywords: | | | |
| Configuration, Change, Management, Plan, Template, Quality Engineer, vob, branch, label | | | |
| ApprovAL TABLE | | | |
| Author | Sign-off Signature #1  Approval date | Sign-off Signature #2 &  Approval date |
| Simona Almajan | Marius Rotaru | Daniel Weyl/Marites Collier |
|  |  |  |

MCAL SW Configuration and Change Management Procedure

REVISION HISTORY

| Version | Date | Author | Comment |
| --- | --- | --- | --- |
| 1.0- D01 | 27 Nov 2009 | Simona Almajan | Initial version. Moved generic content of the existing project specific Configuration Management Plans |
| 1.0 | 29 Jan 2010 | Simona Almajan | Approved: 38131423 |
| 1.1 | 17 May 2010 | Simona Almajan | Added the frozen branch process. Updated the folders structure |
| 1.2 | 01 Oct 2010 | Simona Almajan | Added the Integration CRs usage, cloning process and track tags |
| 1.3 | 12 April 2011 | Simona Almajan | Updates to Integration CRs process, added QP label |
| 1.4 | 31 May 2011 | Simona Almajan | Added ASR4.0 Folders Structure, Quality Documentation Naming Rules |
| 1.5 | 5 August 2011 | Eugenia Neacsa | Updated Doc folder naming convention and FBR versioning |
| 1.6 | 25 January 2012 | Eugenia Neacsa | Changed MISRA reports name, add test reports naming convention for Cosmic compiler and added QP track tag usage. Small format/ style and chapter numbering changes |
| 1.7 | 22 February 2012 | Eugenia Neacsa | Updated the procedure according to ASR 3.2 working approach. |
| 1.8 | 07 May 2012 | Simona Almajan | Merged the MCAL3.0 and MCAL4.0 folders structure in one chapter due to 99% duplicated info,  Moved the FBR Process into FBR Procedure,  Updated Baselines Derivation rules,  Moved the HF Lables naming rules in HF Procedure,  Added the CRs and baselines schema from the Global Change Management Procedure,  Added Compass Folders Structure for process project documentation, |
| 1.9 | 26 July 2012 | Eugenia Neacsa | Updated MCAL3.2 development related rules for configuration management, as follows:   * Folder structure of Test folder; * Corrected branching schemes * Corrected branch naming convention   Updated change management section, as follows:   * Updated the change management process for MCAL   3.2 development (3.2 Config Affected will be used).   * Added management of Errata CR section. * Corrected IPV baseline derivation for MCAL 3.2   baseline IPVs.   * Added more details related to what should be   updated during CR analysis state.   * Updated CR Analysis template: removed “AutoSar   3.x version affected” section.   * Some updates on ICR chapter. |
| 1.10 | 13- July-2012 | Eugenia Neacsa | Updated naming convention for test configuration files and MISRA reports. |
| 1.11 | 09 –October-2012 | Eugenia Neacsa | Updated the change management process for MCAL 3.2 development in terms of baseline treatment.  Updated the CR analysis process.  Updated label naming convention and braches naming convention as a result of latest Autosar 3.2 process changes. |
| 1.12 | 05-Nov-2012 | Simona Almajan | Added SMCAL Vob and labels/baselines naming rules for SMCAL development |
| 1.13 | 17 – Jan -2013 | Eugenia Neacsa | Updated with the latest CQ tool changes (v 8.0.0.1);  Added more details for clone cases (chapter 3.6); |
| 1.14 | 21- Jan -2013 | Simona Almajan | Added MCAL tests configuration management process in vnv\_test VOB  Added MCAL CRs development in VV-TEST-MCAL BIN |
| 1.15 | 19-Mar-2013 | Simona Almajan | Added Chapter 4: MCAL/SMCAL Code Bases Synchronization |
| 1.16 | 09-April-2013 | Simona Almajan | Updated Chapter 4 and cloning section with ASR/IPs specifics |
| 1.17 | 22-May-2013 | Eugenia Neacsa | Added more details for Integration CRs –use cases (chapter 3.5), cloning a CR – use cases (chapter 3.6)  Added Errata Handling –Source Code (chapter 2.6)  Added Test Specification .cfg file naming rules for platforms with multiple derivatives. |

TABLE OF CONTENTS

[MCAL SW Configuration and Change Management Procedure 1](#_Toc357008853)

[1 Introduction 8](#_Toc357008854)

[1.1 Purpose 8](#_Toc357008855)

[1.2 Scope 8](#_Toc357008856)

[1.3 Audience Description 8](#_Toc357008857)

[1.4 References 8](#_Toc357008858)

[1.5 Definitions, Acronyms, and Abbreviations 9](#_Toc357008859)

[1.6 Document Location 10](#_Toc357008860)

[1.7 Problem Reporting Instructions 10](#_Toc357008861)

[2 Configuration Management Activities 11](#_Toc357008862)

[2.1 Overview 11](#_Toc357008863)

[2.2 ClearCase Folders Structure for MCAL 3.0 /3.2/4.0 and SMCAL Software Development 11](#_Toc357008864)

[2.2.1 JSW\_MCAL/XPC56xx 🡪 dev Folder 14](#_Toc357008865)

[2.2.1.1 JSW\_MCAL/XPC56xx / dev 🡪apps Folder 14](#_Toc357008866)

[2.2.1.2 JSW\_MCAL/XPC56xx / dev 🡪drivers Folder 15](#_Toc357008867)

[2.2.1.2.1 Autosar Folder 16](#_Toc357008868)

[2.2.1.2.1.1 Doc Folder 18](#_Toc357008869)

[2.2.1.2.2 IPVault Folder 24](#_Toc357008870)

[2.2.1.3 JSW\_MCAL/XPC56xx/dev 🡪Test Folder 27](#_Toc357008871)

[2.2.2 JSW\_MCAL/XPC56xx 🡪 output Folder 30](#_Toc357008872)

[2.2.3 JSW\_MCAL/XPC56xx 🡪 releases Folder 31](#_Toc357008873)

[2.2.4 JSW\_MCAL/XPC56xx 🡪 tools Folder 32](#_Toc357008874)

[2.2.5 vnv\_test/auto/asr/mcal 🡪Folder 32](#_Toc357008875)

[2.3 ClearCase Branching Strategy for MCAL/SMCAL Software 33](#_Toc357008876)

[2.3.1 Branch types 33](#_Toc357008877)

[2.3.2 Branches Naming Convention 34](#_Toc357008878)

[2.3.3 Branching Schemes 36](#_Toc357008879)

[2.3.3.1 For Autosar Drivers 37](#_Toc357008880)

[2.3.3.1.1 For Generic files 37](#_Toc357008881)

[2.3.3.1.2 For Specific files 38](#_Toc357008882)

[2.3.3.1.3 Doc Files 39](#_Toc357008883)

[2.3.3.1.4 For IPVault Drivers 40](#_Toc357008884)

[2.3.3.2 For Sample Applications Folder 42](#_Toc357008885)

[2.3.3.3 For Tests Folder 42](#_Toc357008886)

[2.3.3.4 For Tools Folder 43](#_Toc357008887)

[2.4 Clearcase Labeling Strategy for MCAL /SMCAL Software 43](#_Toc357008888)

[2.4.1 Component Labels 43](#_Toc357008889)

[2.4.1.1 For IPVault Drivers 44](#_Toc357008890)

[2.4.1.2 For Autosar Drivers – JSW, JSW\_ASR4, SASW VOBs 44](#_Toc357008891)

[2.4.1.3 For Autosar Drivers – vnv\_test VOB 45](#_Toc357008892)

[2.4.1.4 For Stubs Modules 45](#_Toc357008893)

[2.4.1.5 For Sample Applications 46](#_Toc357008894)

[2.4.1.6 For Test Folder 46](#_Toc357008895)

[2.4.2 Release Labels 46](#_Toc357008896)

[2.4.2.1 Official Release Labels – JSW, JSW\_ASR4, SASW VOBs 46](#_Toc357008897)

[2.4.2.2 Official Release Labels – vnv\_test VOB 47](#_Toc357008898)

[2.4.2.3 Patch Release Labels 47](#_Toc357008899)

[2.4.2.4 Quality Package Labels 48](#_Toc357008900)

[2.4.3 Private Labels 49](#_Toc357008901)

[2.4.4 Integration Labels - JSW, JSW\_ASR4, SASW VOBs 49](#_Toc357008902)

[2.5 Compass Folders Structure 50](#_Toc357008903)

[2.6 Errata Handling – Source Code 53](#_Toc357008904)

[3 Change Management Activities 55](#_Toc357008905)

[3.1 Project Initiation Activities (NPI Planning Phase) 56](#_Toc357008906)

[3.2 MCAL BINs 57](#_Toc357008907)

[3.3 Change Requests 79](#_Toc357008908)

[3.3.1 Change Request Lifecycle 79](#_Toc357008909)

[3.3.2 Submitting a Change Request (NEW State) 80](#_Toc357008910)

[3.3.3 Evaluating a Change Request (CCB and ANALYSIS States) 87](#_Toc357008911)

[3.3.4 Implementing a Change Request (OPEN State) 90](#_Toc357008912)

[3.3.5 Closing the Change Request (RESOLVED State) 91](#_Toc357008913)

[3.3.6 Other Fields (Resolved state) 95](#_Toc357008914)

[3.3.7 Accepting a Change Request 96](#_Toc357008915)

[3.3.8 Closing a Change Request (NO CHANGES State) 96](#_Toc357008916)

[3.3.9 Other Fields (not mandatory and not linked to any state) 98](#_Toc357008917)

[3.4 Inquiry CRs 100](#_Toc357008918)

[3.5 Integration Change Requests 101](#_Toc357008919)

[3.6 Forwarding and Cloning a Change Request 104](#_Toc357008920)

[3.7 Specific Track Tags Usage 107](#_Toc357008921)

[3.8 Change management rules for MCAL 3.2 projects 109](#_Toc357008922)

[3.9 Errata Handling – Clear Quest 110](#_Toc357008923)

[3.10 Baselines 111](#_Toc357008924)

[3.10.1 Baseline Lifecycle 111](#_Toc357008925)

[3.10.2 Baselines Introduction 111](#_Toc357008926)

[3.10.3 Planning a Baseline (“PLAN”) 113](#_Toc357008927)

[3.10.4 Opening a Baseline (“OPEN”) 117](#_Toc357008928)

[3.10.5 Staging a Baseline (“STAGE”) 118](#_Toc357008929)

[3.10.6 Testing a Baseline (“TEST”) 119](#_Toc357008930)

[3.10.7 Closing a Baseline (“FINALIZE”) 119](#_Toc357008931)

[3.10.8 Rejecting a Baseline (“TERMINATE”) 120](#_Toc357008932)

[3.10.9 Other Baseline related Fields/Tabs 121](#_Toc357008933)

[3.10.10 MCAL Baselines Naming Convention 121](#_Toc357008934)

[4 MCAL/SMCAL Code Bases Synchronization 127](#_Toc357008935)

[4.1 MCAL QM Code Bases Synchronization 127](#_Toc357008936)

[4.2 MCAL QM - SMCAL Code Bases Synchronization 128](#_Toc357008937)

**LIST OF TABLES**

[Table 1 References Table 9](#_Toc339877702)

[Table 2 Acronyms Table 10](#_Toc339877703)

[Table 3 Table of interdependencies 12](#_Toc339877704)

[Table 4 JSW/JSW\_ASR4/SASW VOBs Clearcase Folders Description 13](#_Toc339877705)

[Table 5 Autosar Modules Names 16](#_Toc339877706)

[Table 6 IPVault Modules Names 24](#_Toc339877707)

[Table 7 Branch types naming convention 35](#_Toc339877708)

[Table 8 External Requests Sources 52](#_Toc339877709)

[Table 9 Parts Affected Table – MCAL BIN 53](#_Toc339877710)

[Table 10 Parts Affected Table – MCAL4.0 55](#_Toc339877711)

[Table 10 Parts Affected Table – SMCAL 57](#_Toc339877712)

[Table 11 Configs Affected Table – MCAL BIN 60](#_Toc339877713)

[Table 12 Configs Affected Table – MCAL4.0 BIN 60](#_Toc339877714)

[Table 12 Configs Affected Table – SMCAL BIN 61](#_Toc339877715)

[Table 13 Permissions Table 67](#_Toc339877716)

[Table 14 BIN Tailoring Table 69](#_Toc339877717)

[Table 15 “New” State ClearQuest Fields 72](#_Toc339877718)

[Table 16 Severity Types 76](#_Toc339877719)

[Table 17 Priority Types Field 78](#_Toc339877720)

[Table 18 CR Priorities – more details 79](#_Toc339877721)

[Table 19 Analysis and CCB States Fields 80](#_Toc339877722)

[Table 20 Open State fields 82](#_Toc339877723)

[Table 21 Resolved State ClearQuest Fields 83](#_Toc339877724)

[Table 22 Root Cause Types Field Elements for Defects 85](#_Toc339877725)

[Table 23 Root Cause Types Field Elements for New Works 87](#_Toc339877726)

[Table 24 Resolved State ClearQuest Fields 88](#_Toc339877727)

[Table 25 No Changes State ClearQuest Fields for Defects 89](#_Toc339877728)

[Table 27 No Changes State ClearQuest Fields for New Works 89](#_Toc339877729)

[Table 27 No Changes State ClearQuest Fields for Inquiry Change Requests 90](#_Toc339877730)

[Table 28 Other Fields/Tabs (not mandatory and not linked to any state) 90](#_Toc339877731)

[Table 29 Plan State Fields (not mandatory and not linked to any state) 104](#_Toc339877732)

[Table 30Plan State Fields 108](#_Toc339877733)

[Table 31 Stage State Fields 109](#_Toc339877734)

[Table 32 Finalized State Fields 110](#_Toc339877735)

[Table 33 QA Designation Field Option 110](#_Toc339877736)

[Table 34 Other Fields (not mandatory and not linked to any state) 111](#_Toc339877737)

**LIST OF FIGURES**

[Figure 1 MCAL Development VOBs - Top Level Structure 13](#_Toc351485730)

[Figure 2 VnV Test VOB - Top Level Structure 13](#_Toc351485731)

[Figure 3 DEV Folder Structure 14](#_Toc351485732)

[Figure 4 Apps folder structure 14](#_Toc351485733)

[Figure 5 Drivers folder structure 15](#_Toc351485734)

[Figure 6 A detailed drivers folder 27](#_Toc351485735)

[Figure 7 Tests folder structure 28](#_Toc351485736)

[Figure 8 JSW\_ASR4.0 Test Folder Structure 30](#_Toc351485737)

[Figure 9 JSW(JSW\_ASR4)output folder structure 31](#_Toc351485738)

[Figure 10 JSW(JSW\_ASR4)releases folders structure 31](#_Toc351485739)

[Figure 11 JSW(JSW\_ASR4)tools folder structure 32](#_Toc351485740)

[Figure 12 Autosar drivers – Generic Files Branching scheme 38](#_Toc351485741)

[Figure 13 Autosar Drivers Specific Files Branches scheme 39](#_Toc351485742)

[Figure 14 Branching scheme for doc files 40](#_Toc351485743)

[Figure 15 IPVault Drivers branching scheme 41](#_Toc351485744)

[Figure 16 Sample Application Folder branching scheme 42](#_Toc351485745)

[Figure 17 Compass Folder Structure 51](#_Toc351485746)

[Figure 18 Change Request Lifecycle 77](#_Toc351485747)

[Figure 19 Inquiry Change Requests Lifecycle 99](#_Toc351485748)

[Figure 20 Integration CRs workflow 100](#_Toc351485749)

[Figure 21 Integration CRs example 101](#_Toc351485750)

[Figure 22 Integration CR Usage 1 102](#_Toc351485751)

[Figure 23 Integration CR Usage 2 103](#_Toc351485752)

[Figure 24 Baseline Lifecycle 108](#_Toc351485753)

[Figure 25 ClearQuest Baselines 109](#_Toc351485754)

[Figure 26 ClearQuest Platform Baselines 110](#_Toc351485755)

[Figure 27 Baselines Derivation 114](#_Toc351485756)

[Figure 28 QM-QM Code Bases Synchronization Matrix 124](#_Toc351485757)

[Figure 29 QM – SMCAL Synchronization Matrix 125](#_Toc351485758)

# 1 Introduction

This document provides the common configuration and change management activities performed for MSGMCAL projects covering software development for both safety and non-safety projects (independent test activities are included)

## Purpose

This document describes the common Configuration and Change Management activities applicable for non-JDP MCAL 3.0, 3.2, 4.0 and SMCAL projects.

Project specific Configuration and Change Management rules and activities are described in project Configuration Management Plan - will be created only if the current procedure cannot cover project specific information

## Scope

The scope of the plan is all non-JDP MCAL 3.0, 3.2, 4.0 and SMCAL Projects.

## Audience Description

The audience for this document is all people involved in the project, including Project Managers, Team Managers, Technical Leaders, Software Developers, Integrators, Testers, and QA team members.

## References

The documents referred across the plan can be found in the following table:

|  |  |
| --- | --- |
| Name | location |
| 1. Configuration Management Plan Template | http://compass.freescale.net/go/206938477 |
| 1. Clearcase Trainings and MCAL Configuration Management Strategy | <http://compass.freescale.net/go/219570339> |
| 1. Configuration & Change Management Compass Folder | http://compass.freescale.net/go/206940631 |
| 1. MCAL Hot Fix Procedure | http://compass.freescale.net/go/206932641 |
| 1. MCAL Frozen Branch Procedure | http://compass.freescale.net/go/206932641 |

Table 1 References Table

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Acronyms | |
| Term/Acronym | Definition |
| API | Application Programming Interface |
| ASR | Autosar |
| Baseline | Record in ClearQuest, as a change request, but with different state machine |
| Branch | A variant development path that diverges from the primary development path |
| Build | An operational version of a system or component that incorporates a specified subset of the capabilities that the final product will provide. |
| CCB | Change Control Board |
| EOL | End of Life |
| CM | Configuration Management |
| JDP | Joint Development Program |
| Configuration Item (CI) | An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process. |
| Configuration Management (CM) | A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. |
| CR | Change Request |
| Label | A string used to uniquely identify a version of a configuration item. |
| Lock | A marker placed on the configuration status of an item, which prevents the item from being modified. |
| Mainline | A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures. |
| MCAL | Microcontrollers Abstraction Layer |
| AUTOMOTIVE SW | Automotive Industrial Software Group |
| Merge | The activity of ‘joining’ modifications performed simultaneously to different copies of the same module.  Merge back: merge operation from a development path to the baseline.  Up-merge: merge operation from the baseline to a development path. |
| Parallel Development | The process of creating and maintaining multiple variants of a software system concurrently. |
| Release | The action whereby an item is made officially available for its intended use. |
| RRR | Release Readiness Review |
| SCM | Software Configuration Management |
| NPI | New Product Introduction |
| SCMP | Software Configuration Management Plan |
| SoC | System on Chip |
| SPICE | Software Process Improvement and Capability dEtermination |
| SQAP | Software Quality Assurance Plan |
| SQE | Software Quality Engineer |
| SRS | Software Requirements Specification |
| TL | Technical Leader |
| TMX | Traceability Matrix |
| TWG | Technical Working Group |
| Version | An identified and documented software product. |

Table 2 Acronyms Table

## Document Location

The document is located at the following address:

<http://compass.freescale.net/go/20693264>

## Problem Reporting Instructions

Problems or correction to this document should reported to the following email address: [Simona.Almajan@freescale.com](mailto:Simona.Almajan@freescale.com)

# Configuration Management Activities

This section contains the common configuration management activities and rules required to manage MCAL projects under existing configuration management system.

## Overview

* **NPI Planning Phase**

The following CM Activities will be performed in NPI Planning phase:

* Project SCMP Creation, review and approval – in needed
* The list of project documentation (process and documentation) will be established. Compass folder shall be created.
* CM and ClearCase Trainings will be delivered if necessary (new team members)
* **NPI Execution Phase**

The following CM Activities will be performed in NPI Execution phase:

* MCAL projects software development activities will be performed according to the current document and Project Configuration Management Plan (if applicable)

## ClearCase Folders Structure for MCAL 3.0 /3.2/4.0 and SMCAL Software Development

The main idea behind MCAL Folders Structure and Branching Strategy which will be presented in the next chapters is achieving the following goals:

* Maximum **code reusability** between different SoCs and different Autosar revisions
* Minimum **effort** in porting MCAL on different platforms
* Multi-core architecture scalability

Some prerequisites to be taken into account:

* MCAL drivers will be developed multi-site
* The need to establish a clear ownership for each MCAL or IP drivers
* The need to have a common baseline for all drivers

**Outcome:**

**Component Base Approach** is the only solution to have one common baseline/multi-site development/clear ownership.

In conclusion, the

* New MCAL drivers organization: Autosar drivers/IPVault drivers
  + Autosar specific drivers - Autosar dependent/ Platform dependent
  + IPVault drivers -: Autosar dependent/ IP dependent
* SoC/Compiler specific files to be grouped in one place and one component and used by all other apps/test.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Autosar Drivers (High Level Drivers)** | **IPVault Drivers (Low Level Drivers)** | **Applications** | **Test** | **Tools  (TPB)** | **Common (compiler/mmap/make files...)** |
| **Autosar version** | x | x | x | x | x |  |
| **Platform/SoC** | x |  | x | x |  | x |
| **IP Version/Type** |  | x |  |  |  |  |
| **Tresos Version** |  |  |  |  | x |  |

Table 3 Table of interdependencies

1. MCAL 3.0 and MCAL 3.2 Drivers will be developed in **JSW** vob:

/**JSW**/JSW\_MCAL/XPC56xx - contains MCAL 3.0/3.2 low-level & high level drivers (source code, libraries, makefiles, config files, tresos files, engineering documentation)

1. MCAL 4.0 Drivers will be developed in **JSW**\_**ASR4**vob:

/**JSW**\_**ASR4**/JSW\_MCAL/XPC56xx - contains MCAL 4.0 low-level & high level drivers (source code, libraries, makefiles, config files, tresos files, engineering documentation

1. SMCAL Drivers will be developed in the SASW vob:

/**SASW**/JSW\_MCAL/XPC56xx - contains SMCAL low-level & high level drivers (source code, libraries, makefiles, config files, tresos files, engineering documentation)

1. MCAL/SMCAL Tests – testing activities performed by the Independent Test Group – are developed in **vnv\_test** VOB

Note: Test cases from SASW VOBs have been transferred to vnv\_test in January 2013.

The next chapters cover the folders structure applicable for all four VOBs.

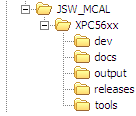


Figure 1 MCAL Development VOBs - Top Level Structure

| Folder name | Description |
| --- | --- |
| **JSW/JSW\_ASR4/SASW** | **Top level directory for MCAL 3.0 (JSW), MCAL4.0 (JSW\_ASR4) and SASW development** |
| JSW\_MCAL | Generic JDP project identifier. From historical anf folders structure migration reasons, the JDP identfied was kept for non-JDP development |
| XPC56xx | Automotive projects |
| dev | Development root |
| docs | Project specific documentation |
| output | Build releated output files (dir only) |
| releases | Software release artifacts |
| tools | Non-commercial tools to facilitate the build and release of software |

Table 4 JSW/JSW\_ASR4/SASW VOBs Clearcase Folders Description

TBD

Figure 2 VnV Test VOB - Top Level Structure

| Folder name | Description |
| --- | --- |
| **TBD** |  |
|  |  |
|  |  |
|  |  |

Table 5 vnv\_test VOB Clearcase Folders Description

### JSW\_MCAL/XPC56xx 🡪 dev Folder

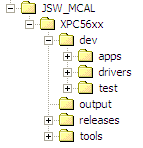


Figure 3 DEV Folder Structure

#### JSW\_MCAL/XPC56xx / dev 🡪apps Folder

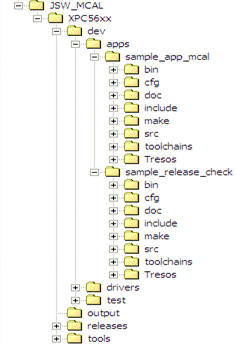
****

Figure 4 Apps folder structure

This repository shall contain sample applications or release checks for one or many Autosar modules. Each sample application will include a module specific configuration, source code and makefiles and eventually CW/GHS project settings.

* …/dev/apps/sample\_app\_mcal/**bin** folder contains application configuration files for the specific Autosar Module.
* …/dev/apps/sample\_app\_mcal/**cfg** folder contains application configuration files for the specific Autosar Module.
  + …/cfg/**include**  folder contains application header files of ASR Modules used by sample application (Example: Port\_cfg.h)
  + …/cfg/**src** folder contains application configuration source files of ASR Modules used by sample application. (Example: Port\_cfg.c)
* …/dev/apps/sample\_app\_mcal/**doc** folder contains sample application documentation like: UML Design and Integration Application document
* …/dev/apps/sample\_app\_mcal /**include** folder contains include files for the specific Autosar Sample Application. Example: main.h
* …/dev/apps/ sample\_app\_mcal /**make** folder contains application makefiles for the specific Autosar Sample Application. Example: CW, Lauterbach, ghs(compiler specific) makefiles. Examples: makefile, rules.mak
* …/dev/apps sample\_app\_mcal/**toolchains** folder contains configuration files for CW Startup files, Flash, Intram and OS
* …/dev/apps sample\_app\_mcal/**Tresos** folder contains source application files for the specific Autosar Sample Application.

#### JSW\_MCAL/XPC56xx / dev 🡪drivers Folder

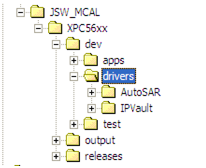
****

Figure 5 Drivers folder structure

The Drivers folder is split in two folders in terms of module category:

* **Autosar folder** containing Autosar Drivers
* **IPVault folder** containing IP specific Drivers

##### Autosar Folder

The list of Autosar Drivers is available in the following table:

|  |  |
| --- | --- |
| Name of the Autosar Module “asrmod” | Description |
| ADC | Analog to Digital Converter |
| BASE | Virtual Component containing platform settings and compiler abstraction layer |
| CAN | Controller Area Network |
| CANIF | CAN Interface Stub |
| DEM | Diagnostic Event Manager Stub |
| DET | Development Error Tracer Stub |
| DIO | Digital I/O |
| ECUM | ECU Manager Stub |
| FEE | Flash EEPROM Emulation |
| FLS | FLASH |
| FR | Flexray Module |
| FRIF | Flexray Interface Module |
| GPT | General Purpose Timer |
| ICU | Input Capture Unit |
| LIN | Local Interconnect Network |
| LINIF | LIN Interface Stub |
| AUTOMOTIVE SW | Microcontroller Unit |
| MEMIF | Memory Interface Stub |
| PORT | PORT |
| PWM | Pulse Width Modulation |
| RESOURCE | Resource Module |
| SCHM | Schedule Manager Stub |
| SPI | Serial Peripheral Interface |
| WDG | Watchdog Timer |
| WDGIF | Watchdog Interface Stub |
| MCEM | Non-Autosar Driver |
| CRC | Non-Autosar Driver |

Table 5 Autosar Modules Names

Notes:

* One Autosar Driver can be implemented using one or more IPVault Drivers.
* Autosar High Level APIs are SoC independent.
* In order to isolate/group all files needed for one Autosar MCAL Driver in one place, this folder should contain also the config files/ LLD which are SoC Dependent.

This folder contains one folder for each Autosar module, this last one containing documentation, includes, source files, configuration files, generated source code and includes. There are two specific modules BASE and RESOURCES which are not Autosar specific modules but there are used by all Autosar modules because they contain compiler specific defines and typedefs as well as platform dependant configuration information.

Description of all subfolders from “asrmodl”:

* …/drivers/Autosar/asrmodl/**doc** contains module engineering documentation like: detailed design docs, test plans, test reports, MISRA reports, LDRA reports, traceability matrices
  + **…/doc/design** contains ASR module design document and test specification
  + **…/doc/reports** contains test reports, MISRA2004 violation report file, LDRA reports, Traceability Matrix
  + **…/doc/user\_manuals** contains Autosar Driver User’s Manual and Integration Manual (Example: AUTOSAR\_MCAL\_DIO\_IM.doc)
* …/drivers/Autosar/asrmodl/**generic**/**include** contains the definition of APIs, specific defines, types and function declaration to be exported by Autosar drivers
* …/drivers/Autosar/asrmodl/**generic**/**src** contains source files for Autosar drivers (implementation of Autosar Modules Interface functions (APIs))
* …/drivers/Autosar/asrmodl/**specific** is the SoC specific folder
* …/drivers/Autosar/asrmodl/**specific**/**config** contains Tresos Config files
* …/drivers/Autosar/asrmodl/**specifc**/**generate** contains drivers templates for automatic code generation
* …/drivers/Autosar/asrmodl/**specific**/**generate**/**include** contains generated header files templates
* …/drivers/Autosar/asrmodl/**specific**/**generate**/**src** contains generated source file templates
* …/drivers/Autosar/asrmodl/**specific**/**include** contains headers for SoC and IP specific low level modules

The IP Specific files are symbolic linked from IPVault folder.

*Naming Convention for MCAL 3.0/3.2:*

***<ip1\_lld.h>,* <Reg\_eSys\_Ip2.h>, *<ip1\_lld\_CfgEx.h>***

***<ip2\_lld.h>,* <Reg\_eSys\_Ip2.h>, *<ip2\_lld\_CfgEx.h>***

Example: emios\_lld.h, Pit\_Gpt\_LLD\_CfgEx.h , Reg\_eSys\_LINFlex.h

Naming Convention for SoC Specific files:

**<asrmod\_lld.h>**

Example: adc\_lld.h

*Naming convention for MCAL4.0:*

**<asrmod1\_Ip1\_lld.h>, <Reg\_eSys\_Ip1.h>, <asrmodl>\_<ip1>\_lld\_CfgEx.h**

**<asrmod2\_Ip2\_lld.h>, <Reg\_eSys\_Ip2.h>, <asrmod2>\_<ip2>\_lld\_CfgEx.h**

Example: Gpt\_eTimer\_LLD.h, Reg\_eSys\_eTimer.h

Naming Convention for SoC Specific files:

**<asrmod\_lld.h>**

Example: pwm\_LLD.h

* …/drivers/Autosar/asrmodl/**specific**/**src** contains source files for SoC and IP specific low level modules.

*Naming Convention for MCAL3.0 /3.2:*

***<ip>\_irq.c>***

***<ip>\_lld.c>***

Example: wdg\_irq.c, wdg\_lld.c

Naming Convention for SoC Specific files:

***<asrmod\_irq.c>***

***<asrmod\_lld.c>***

*Naming Convention for MCAL4.0:*

*<Asrmod>\_<Ip>\_lld.c*

Example: **Can\_FlexCan\_LLD.c**

Naming Convention for SoC Specific files:

*<Asrmod>\_irq.c*

*<Asrmod>\_lld.c*

Example: **Can\_irq.c, Can\_LLD.c**

* …/drivers/Autosar/asrmodl/**specific**/**resource** contains TPB resource files

Example: asrmod1\_xpc5602s\_lqfp100.txt

Note: SMCAL follows the same naming rule like MCAL 4.0

###### Doc Folder

This is the naming convention to be used for the documentation inside the **Doc** Folder:

**Note:** Excel documents can have the following extensions:

* **.xls** - Office 2003 format, can also contain macros
* **.xlsx** - Office 2007 format, can not contain macros
* **.xlsm** - Office 2007 format, can contain macros

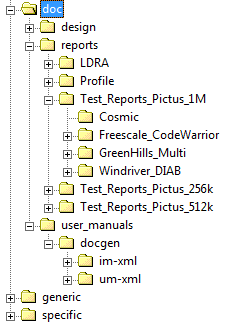
The **<reports>** folder content is as follows and as displayed in the below picture (example):

**+LDRA** folder - under **<reports>** folder

**+Profile** folder - under **<reports>** folder

**+Review checklist** - under **<reports>** folder

**+Test\_Reports\_<Platform\_name>** folder - under <**reports>** folder

****

* **<Module>**
* **doc –** under **<module>** folder
* **design -** under **<doc>** folder

-AUTOSAR\_MCAL\_<MODULE>\_SDD.eap - Design document

Example: AUTOSAR\_MCAL\_DIO\_SDD.eap

-AUTOSAR\_MCAL\_<MODULE>\_TP.doc - manual written document

Or

-AUTOSAR\_MCAL\_<MODULE>\_TS.pdf - automatically generated document

and

- AUTOSAR\_MCAL\_<MODULE>\_TS\_CFG.html or AUTOSAR\_MCAL\_<MODULE>\_TS\_CFG\_COMMON.html **for common tests**

AUTOSAR\_MCAL\_<MODULE>\_TS\_CFG\_<derivative1>.html

AUTOSAR\_MCAL\_<MODULE>\_TS\_CFG\_<derivative2>.html

AUTOSAR\_MCAL\_<MODULE>\_TS\_CFG\_<derivative3>.html

**Test configuration files are automatically generated by the tool used to prepare the Test Specification.**

**Note:** Test Configuration files depend on the input received from make file and are generated as follows:

* if the tests support only one derivatives, a single .html configuration file will be generated (e.g AUTOSAR\_MCAL\_ADC\_TS\_CFG.html)
* if the tests support multiple derivatives, more .html configuration files will be generated (one common .html file and one. html for each derivative or one for each derivative /one common).

Example: AUTOSAR\_MCAL\_DIO\_TP.doc, AUTOSAR\_MCAL\_DIO\_TS.pdf, AUTOSAR\_MCAL\_SPI\_TS\_CFG.html, AUTOSAR\_MCAL\_ADC\_TS\_CFG\_COMMON.html, AUTOSAR\_MCAL\_ADC\_TS\_CFG\_1M.html

* **reports -** under **<doc>** folder

The next folders are under **reports** Folder:

* **LDRA**
* AUTOSAR\_MCAL\_<MODULE>\_LDRA\_reports.zip - zip containing the LDRA reports files (txt or htm),   grouped in <test\_name1>\_tbwrkfls directories.
* **Profile**
* AUTOSAR\_MCAL\_<module>\_<platform>\_<derivative>\_ProfileReport.xls/xlsx/xlsm -File containing profile information for tested derivatives and different compilers.

Example: AUTOSAR\_MCAL\_DIO\_56xB\_5604B\_ProfileReport.xls

**Or**

* AUTOSAR\_MCAL\_<Module>\_ProfileReport\_56xxL\_5643L\_Compiler1/Compiler2/Compiler3/Compiler4.xls/xlsx/xlsm

Example: AUTOSAR\_MCAL\_ADC\_ProfileReport\_56xxL\_5643L\_Compiler1.xlsx

AUTOSAR\_MCAL\_ADC\_ProfileReport\_56xxL\_5643L\_Compiler2.xlsx

AUTOSAR\_MCAL\_ADC\_ProfileReport\_56xxL\_5643L\_Compiler3.xlsx

AUTOSAR\_MCAL\_ADC\_ProfileReport\_56xxL\_5643L\_Compiler4.xlsx

**!!Note:**

1. **Platform name and derivative name will be automatically extracted by the tool.**
2. **The name of the Profile report file depends on the version of the tool used to extract the report.**

**+** **Review checklist** folder containing the final versions of the review checklists **(filled-out checklists)**

Example: Checklist for Code Review ,Checklist for Design Review

**!!Note: The review checklist should be available for RTM releases.**

* **Test\_Reports\_<Platform\_name>** - Ex: Test\_Reports\_Bolero\_1.5M

The next folders are under **Test\_Reports\_<Platform\_name>** Folder:

**+Compiler\_Independent** - contains an archive with all tests reports for the compiler independent tests (HTMLs and xml).

Example: Compiler\_Independent.zip

**+Freescale\_CodeWarrior** - contains an archive with all tests reports using CW compiler (HTMLs and xml)

Example: Freescale\_CodeWarrior.zip

**+GreenHills\_Multi** - contains an archive with all tests reports using GHS compiler (HTMLs and xml).

Example: GreenHills\_Multi.zip

**+Windriver\_DIAB** - contains an archive with all tests reports using DIAB compiler (HTMLs and xml).

Example: Windriver\_DIAB.zip

**+Cosmic**  **-** contains an archive with all tests reports using Cosmic compiler (HTMLs and (if available) xml).

Example: Cosmic.zip

The next files are under **reports** Folder:

* AUTOSAR\_MCAL\_<Module>\_VSMDReport.html - VSMD report

Example: AUTOSAR\_MCAL\_DIO\_VSMDReport.html

* AUTOSAR\_MCAL\_<Module>\_SizeReport\_<platform>\_<derivative>\_compiler1/compiler2/compiler3/compiler4.xls/xlsx/xlsm - Code size and stack size for compiler 1/2/3/4. See above the other possible extensions for Excel docs.

Example: AUTOSAR\_MCAL\_DIO\_SizeReport\_56xD\_5602D\_compiler1.xls

**!!Note**: **Platform name and derivative name will be automatically extracted by the tool.**

- AUTOSAR\_MCAL\_<MODULE>\_MisraReport\_<test>.log/.txt - PCLint MISRA violation log (as generated by the static analysis scripts)

Example: AUTOSAR\_MCAL\_DIO\_MisraReport\_ts\_02.log, AUTOSAR\_MCAL\_CAN\_MisraReport\_Can\_Tp\_int01.log, AUTOSAR\_MCAL\_PORT\_MisraReport\_port\_tp\_01.log, AUTOSAR\_MCAL\_SPI\_MisraReport\_Spi\_TS\_004.log

and

- AUTOSAR\_MCAL\_<MODULE>\_MisraReport\_<test>.xls/xlsx/xlsm - Parsed PCLint MISRA violation log (as generated by the static analysis scripts).

Example: AUTOSAR\_MCAL\_DIO\_MisraReport\_ts\_02.xlsm, AUTOSAR\_MCAL\_ADC\_MisraReport\_Adc\_TS\_001.xlsx, AUTOSAR\_MCAL\_CAN\_MisraReport\_Can\_Tp\_int01.xlsx

**Or:**

- AUTOSAR\_MCAL\_<MODULE>\_MisraReport >.xls/xlsx/xlsm - Parsed PCLint MISRA violation log (as generated by the static analysis scripts). Example:AUTOSAR\_MCAL\_DIO\_MisraReport.xls

**!!!Note:** The name of the MISRA report file depends on the version of the tool used to extract the report. *Example*: “make lint” tool generates a MISRA Excel report with test id included in the file name, “make lint\_all” will generates multiple MISRA excel/log/txt files with test id in the file name and one MISRA Excel file without test id in the file name. In second case, the MISRA report without test id in the file name contains all info from the other files, and this report should be available in the vod. If possible the archive containing all other MISRA logs may be also saved in the vob (not mandatory).

- AUTOSAR\_MCAL\_<MODULE>\_CodeCoverageReport.txt - Code coverage report

Example: AUTOSAR\_MCAL\_DIO\_CodeCoverageReport.txt

* AUTOSAR\_MCAL\_<MODULE>\_CodeCoverage\_Summary.xls/xlsx/xlsm - Code coverage summary report. See above the other possible extensions for Excel docs.

Example: AUTOSAR\_MCAL\_DIO\_CodeCoverage\_Summary.xls

* AUTOSAR\_MCAL\_<MODULE>\_StaticAnalysis\_Summary.xls/xlsx/xlsm - Static analysis summary report. See above the other possible extensions for Excel docs.

Example: AUTOSAR\_MCAL\_DIO\_StaticAnalysis\_Summary.xls

* AUTOSAR\_MCAL\_<MODULE>\_SummaryTestReport.html - Summary test report

Example: AUTOSAR\_MCAL\_DIO\_SummaryTestReport.html

* AUTOSAR\_MCAL\_<MODULE>\_TraceabilityMatrix.xlsm(for 3.0) (.xls or xlsx extension for 4.0) - Requirements traceability matrix.

Example: AUTOSAR\_MCAL\_DIO\_TraceabilityMatrix.xlsm

* AUTOSAR\_MCAL\_<MODULE>\_EPD\_Test\_Results.xls/xlsx/xlsm - Test results for "One EPD" and "All EPD" tests. See above the other possible extensions for Excel docs. This file is not mandatory and will be removed as tests are updated for automatic run.

Example: AUTOSAR\_MCAL\_DIO\_ EPD\_Test\_Results.xls

* **user\_manuals -** under **<doc>** folder
* **docgen -** under **<user\_manuals>** folder
* **im-xml** – folder containing the .xml files and subfolders with images and topics for Freescale owned modules. For ST modules, folder will be empty.
* **um-xml** - folder containing the .xml files and subfolders with images and topics for Freescale owned modules. For ST modules, folder will be empty.
* im-xml.zip – This archive contains all the files needed to create the IM pdf document (all from im-xml folder for Freescale drivers).
* um-xml.zip - This archive contains all the files needed to create the UM pdf document (all from um-xml folder for Freescale drivers).

***!!!Note: im-xml.zip and um-xml.zip documents are under “docgen” folder.***

* AUTOSAR\_MCAL\_<MODULE>\_IM.pdf  -  Integration manual (pdf format)

Example: AUTOSAR\_MCAL\_DIO\_IM.pdf

* AUTOSAR\_MCAL\_<MODULE>\_UM.pdf  - User manual (pdf format)

Example: AUTOSAR\_MCAL\_DIO\_UM.pdf

***!!!Note: IM/UM documents are under “user\_manuals” folder.***

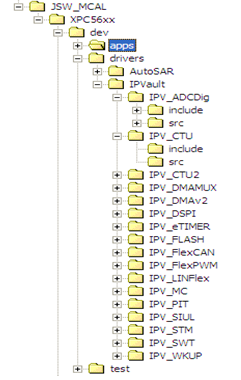
* **generic** - under **<Module>** folder
* **specific** - under **<Module>** folder
* <Module>.mak - TPB configuration file to generate the
* <Module> Tresos plugin

*Legend:*

* xyz   - xyz is a folder.
* abc   - abc is a file.

Indentation used to depict the folder tree

##### IPVault Folder

****

IPVault folder contains include and source files which are “ip” specific.

The names of the IPs are listed in the following table, in Clearcase they are prefixed by “IPV\_”

|  |  |
| --- | --- |
| Name of the IPVault Module “ip” | Description |
| IPV\_ADC | Analog to Digital Converter |
| IPV\_ADCDig | ADC Digital Interface |
| IPV\_CTU | Cross Trigger Unit |
| IPV\_CTU2 | Cross Trigger Unit 2 |
| IPV\_DMA | Direct Memory Access |
| IPV\_DMAMUX | DMA Muxing |
| IPV\_DMAV2 | Direct Memory Access |
| IPV\_DSPI | Digital Serial Peripheral Interface |
| IPV\_EMIOS | Enhanced Modular IO Subsystem |
| IPV\_ECSM | Extended Crypto Security Module |
| IPV\_EQADC | Extended QADC |
| IPV\_ETIMER | Enhanced Timer |
| IPV\_ESCI | Enhanced Serial Communication Interface |
| IPV\_INTC | Interrupt Controller |
| IPV\_FLEXCAN | Flexible Controller Area Network |
| IPV\_FLEXPWM | Flexible PWM |
| IPV\_FLEXRAY | Flexray |
| IPV\_FLASH | Flash |
| IPV\_FMPLL | Frequency Modulation Phase Locked Loop |
| IPV\_FRIPI | Flexray IP Interface |
| IPV\_LINFLEX | Local Interconnect Network Flex |
| IPV\_QSPI | Quad Serial Peripheral Interface |
| IPV\_MC | IPV Magic Carpet |
| IPV\_PIT | Periodic Interrupt Timer |
| IPV\_PITRTI | PIT Real Time Interface |
| IPV\_RTC | Real Time Clock |
| IPV\_SIU | System Integration Unit |
| IPV\_SIUL | System Integration Unit Lite |
| IPV\_SWT | Software Timer |
| IPV\_STM | System Timer Module |
| IPV\_WKPU | Wake Up Unit |

Table 6 IPVault Modules Names

* …/drivers/IPVault/**<IP>**/**include** folder contain IP specific include files

Naming convention for MCAL3.0/3.2:

<ip>\_lld.h

Reg\_eSys\_<ip>.h

<ip>\_lld\_CfgEx.h

Examples:

emios\_lld.h

Reg\_eSys\_emios.h

LINFlex\_LLD\_CfgEx.h

Naming convention for MCAL4.0:

<Asrmod>\_<Ip>\_lld.h

Reg\_eSys\_<Ip>.h

<Asrmod>\_<ip>\_lld\_CfgEx.h

Examples:

Can\_FlexCan\_LLD.h

Reg\_eSys\_FlexCan.h

Pwm\_eTimer\_LLD\_CfgEx.h

* …/drivers/IPVault/<**IP>**/**src** folder contain IP specific source files

Naming convention for MCAL3.0/3.2:

<ip>\_lld.c

Reg\_eSys \_<ip>.c

Examples:

emios\_lld.h

Reg\_eSys\_ETIMER.c

Naming convention for MCAL4.0

*<Asrmod2\_Ip2\_lld.c>*

Example: Can\_FlexCan\_LLD.c

Naming Convention for SoC Specific files:

*<Asrmod1\_irq.c>*

*<Asrmod2\_lld.c>*

Example: Can\_irq.c, Can\_LLD.c

Note: SMCAL folloes the same naming rule like MCAL 4.0.

**Notes:**

* Autosar does not stipulate how to map the MCAL drivers on SoC IPs.

The Autosar defines the interfaces with upper levels. It does not give us any suggestion about how to map the Autosar drivers on SoC IPs. (ie: GPT can be implemented either using STM or eMIOS… )

* There is not a direct mapping (one to one) between ASR Module and SoC IPs The following cases have been identified:
* one to one mapping (ie. MCAL LIN Driver <-> MPC56XXL 3.0: LinFLEX IP)
* one to many mapping (i.e MCAL GPT Driver <-> MPC56XXL 3.0 IPs: STM, eMIOS, RTC, PIT )
* many to one mapping (i.e MCAL PORT, DIO <-> MPC56XXL 3.0 IP: SIUL)

A more detailed picture of Drivers folder highlighting the ASR-IP Interconnection:

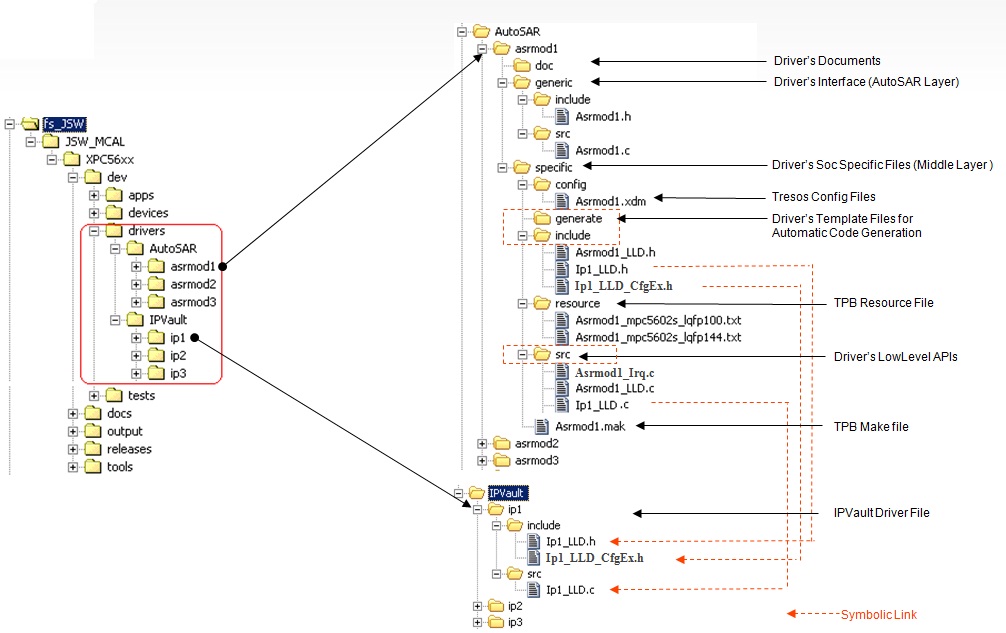


Figure 6 A detailed drivers folder

#### JSW\_MCAL/XPC56xx/dev 🡪Test Folder

Note: This section is applicable for MCAL 3.0/3.2 and 4.0 only.

For SMCAL projects, the full testing is performed by independent test group and test artifacts are located in vnv\_test VOB.

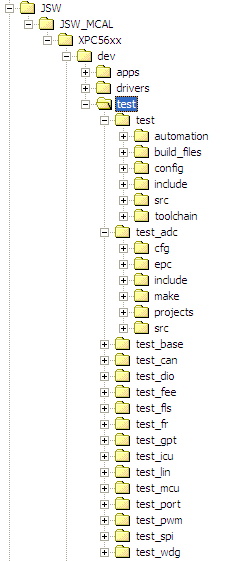
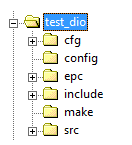
****

Figure 7 Tests folder structure

This repository shall contain all integration related test configuration and source code.

* …/test/**test\_adc** folder contain Autosar Module specific:
* Cfg - files as well as \*.epd and \*.xdm files Example: <asrmod1>\_cfg.h, or <asrmod1>\_cfg.c
* epc files
* project files (for the case when CW and GHS IDEs are used to run the tests)
* include -test case specific headers Example: test\_<Autosar1>.h
* test case specific source files Example: test\_<Autosar1>.c
* make -test case specific makefile, Example: test\_01.mak
* tool chain contains compiler .mak files



MCAL 3.2 product specific rules:

* Create new ASR3.2 specific folders where to store MCAL3.2 **epc** files;
* Files which might be impacted by the Autosar 3.2 development are .epc files and .c files (traceability tags should be updated)

MCAL 4.0 folder structure for **Test Folder**:

The …/test/**test\_asrmod** folder contain Autosar Module specific:

**+…/generic** files

+…/**generic**/**include** files

+…/**generic**/**src** files ex: Asrmod\_TC\_001.c, Asrmod\_TC\_002.c

+ …/**make** files used define the test suites ex: Asrmod\_TS\_API\_01.mak, Asrmod\_TS\_API\_02.mak, Asrmod\_TS\_PERF\_01.mak

+ …/**specific files** for registering theTest Suites to “Beart” and run them , for including configuration files and other platform dependent files (interupt vector table…/ Platform init…)

+…/**specific**/**cfg/cfg\_<XX>/<derivative>**

Where <XX> should be unique identifier: 01, 02, …

Where <derivative> or silicon variant should be: **512k, 1M, 1M5, 2M, 4M, …..**

+…/**specific**/**include** files

+…/**specific**/**src** to register Test Suites to “Beart” and run themfiles like AsrmodEnvSEtup.c, Vector\_VLEASRMOD.c

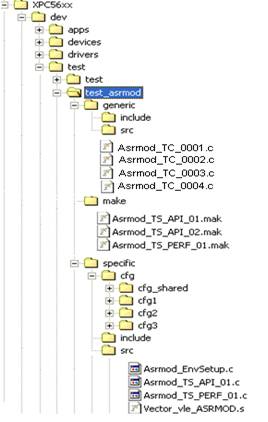
****

Figure 8 JSW\_ASR4.0 Test Folder Structure

**Rules:**

-One test per file

-All file from generic folder can be considered platform independent.

### JSW\_MCAL/XPC56xx 🡪 output Folder

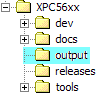
****

Figure 9 JSW(JSW\_ASR4)output folder structure

The …/XPC56xx**/output** folder contains all files and folders generated by TPB and Build Environment. These elements will not be under source control.

### JSW\_MCAL/XPC56xx 🡪 releases Folder

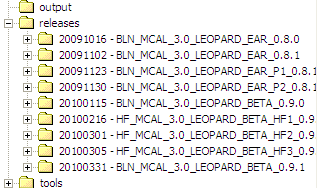
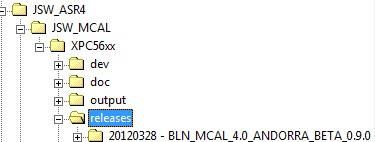
****

Figure 10 JSW(JSW\_ASR4)releases folders structure

This repository shall contain all of the software release artifacts.

### JSW\_MCAL/XPC56xx 🡪 tools Folder

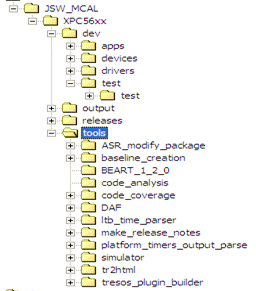


Figure 11 JSW(JSW\_ASR4)tools folder structure

This folder will be a symlink pointing to the repository that contains all necessary tools to facilitate the build and release of software.

### vnv\_test/auto/asr/mcal 🡪Folder

The ITG tests folder structure managed in vnv\_test VOB is describved in ITG Configuration and Change Management Procedure available at the below location:

<http://compass-phx.freescale.net/livelink/livelink/228862859/ITG_Configuration_and_Change_Management_Procedure.docx?func=doc.Fetch&nodeid=228862859>

## ClearCase Branching Strategy for MCAL/SMCAL Software

### Branch types

Branches are used to support development by providing a mechanism that allows a developer to isolate work for bug fixes, code reorganization, prototypes, platform specific development, etc. The project uses the following branch types:

* Main branch type. Used as a trunk branch instead on /main.
* Mainline branch – mln - is used to deliver an integrated and tested release. A mainline branch can be used for internal delivery and external delivery.
* Debug branch type – dbg - A debug branch is used for a non-formal work on a short-term basis. Developers may use a debug branch to try out unofficial changes to code. No CLEARQUEST record for these temporary changes is required.  
  A debug branch must not be used as a baseline for a branch other than another debug branch. When a debug branch type is no longer being used, it must be lock-obsolete.
* Development branch type – dev- A development branch holds an official change made by a developer: it is always associated with a CLEARQUEST change request and includes the CR identifier in its name. Only one development branch should be associated to a CR.  
  A development must neither be created off a debug branch nor have a debug branch in its extended path. It must not be created off another development branch. A new development branch should always be created off the product baseline. A development branch/associated CR can be used across VOBs.  
  A development branch shall be caught up with the latest stable integration, checked in, built, and tested before submission. No changes may be made to a branch after it has been submitted for release. When a development branch is ready for a release, it must be submitted for integration by the developer.
* Integration branch type – int - An integration branch is used to either integrate development efforts or combine multiple integration branches. An integration branch is always associated with a component mainline branch. Only a designated development team representative can create an integration branch.  
    
  No inspection is performed on an integration branch. An integration branch must be created off an official release label. No change other than for correcting a merge conflict should occur on an integration branch.

### Branches Naming Convention

The following table details the Branch Naming conventions of the MCAL 3.0, MCAL 3.2, MCAL4.0 and SMCAL projects.

**The branches’ name must be always in lower cases**

| Branch Type | Naming Convention |
| --- | --- |
| Mainline - mln | 1. **For Autosar Drivers – all software development VOBs** 2. **For High-level (common files)**   mln\_<asr\_module>\_mcal<asr\_version>   1. **For low-level (specific files)**   mln\_<asr\_module>\_mcal<asr\_version> \_<platform\_name>   1. **For IPVault Drivers – all software development VOBs**   mln\_<ip\_name>\_mcal<asr\_version>  st\_ mln\_<ip\_name>\_mcal<asr\_version>  *where <asr\_version> is 3.0 or 4.0*   1. **For Tests Folder – all software development VOBs, including vnv\_test VOB**   The same as for case 1) a) for generic tests and 1) b) for specific tests   1. **For apps folder - all software development VOBs**   mln\_<sample\_app\_name>\_mcal<asr\_version> \_<platform\_name>   1. **For tools - tools VOBs**   mln\_<tool\_name> or mln\_tools  Note: For the safety VOB, \_mcal\_ keyword will be used due to reusing the same scripts that generates the branches  Example:   * 1. mln\_gpt\_mcal3.0\_xpc56xxl, or mln\_gpt\_mcal3.0   2. mln\_gpt\_mcal3.2\_xpc560xb, or mln\_gpt\_mcal3.2   3. mln\_emios\_mcal3.0   4. mln\_emios\_mcal3.2   5. mln\_samcal\_mcal3.0\_xpc56xxl   6. mln\_samcal\_mcal3.2\_ xpc560xb   7. mln\_gpt\_mcal4.0\_xpc56xxl, or mln\_gpt\_mcal4.0   8. mln\_emios\_mcal4.0   9. mln\_samcal\_mcal4.0\_xpc56xxl   10. mln\_tools or mln\_tpb |
| Integration - int | 1. **For Autosar Drivers - all software development VOBs** 2. **For High-level (common files)**   int\_<asr\_module>\_mcal<asr\_version>   1. **For low-level (specific files)**   int\_<asr\_module>\_mcal<asr\_version> \_<platform\_name>   1. **For IPVault Drivers - all software development VOBs**   int\_<ip\_name>\_mcal<asr\_version>  *where <asr\_version> is 3.0 or 4.0*   1. **For Tests Folder - all software development VOBs, including vnv\_test VOB**   The same as for case 1)b)   1. **For apps folder - all software development VOBs**   int\_<sample\_app\_name>\_mcal<asr\_version> \_<platform\_name>   1. **For tools folder – tools VOBs**   int\_tools or int\_<tool\_name>  Example: mln\_common   1. int\_gpt\_mcal3.0\_xpc56xxl, or int\_gpt\_mcal3.0 2. int\_gpt\_mcal3.2\_xpc560xb, or int\_gpt\_mcal3.2 3. int\_emios\_mcal3.0 4. int\_emios\_mcal3.2 5. int\_samcal\_mcal3.0\_xpc56xxl 6. int\_samcal\_mcal3.2\_xpc560xb 7. int\_gpt\_mcal4.0\_xpc56xxl, or int\_gpt\_mcal4.0 8. int\_emios\_mcal4.0 9. int\_samcal\_mcal4.0\_xpc56xxl 10. int\_tools, or int\_tpb |
| Development - dev | dev\_<CR\_ID>\_<core\_ID>\_{Short\_Text}  Example: **dev\_engr01234\_ngys001\_fix\_bug** |
| Debug - dbg | dbg\_<core\_ID>\_{short text}  Example: **dbg\_vburlac1\_l1t\_overflow** |
| Field | Description |
| <CR\_ID> | CLEARQUEST Change Request number |
| <short\_text> | Short text for whatever user’s willing |
| < userid> | User’s core ID |
| <asr\_module> | Autosar module |
| <asr\_version> | Autosar version |
| <platform\_name> | Platform Name |
| <ip\_name> | IP name |

Table 7 Branch types naming convention

**Note:**

* No Branches on first level directories of the JSW or SASW Drivers’ VOBs
* SASW Branches creation and following the JSW\_ASR4 naming rules
* vnv\_test VOB development are following the same branching strategy with test folders previously located in MCAL VOBs before transfer.

To avoid the problem of duplicated files creation, no branches will be created on directories that are at the first level of the VOB structure.

The problem with branching on director is the following: if two files are created in the same directory with the same name but in different branch, when you want to merge the two branches, ClearCase considers that those files are different, and you can not merge the files into a single one.

Regarding platform names, from historical reasons, due to JDP partnership, in JSW and JSW\_ASR4 VOBs, they start either with “x”, or with “m”, eg: xpc560xb, mpc560xb

Starting with January 2013, platform names shall start with “X”. Platform names are contained in: names of the Clearcase branches, labels, and ClearQuest baselines eg: xpc560xb.

Note that User Documentation contains the platform names in the format required by the Integrator, which might differ from the format used in Configuration and Change Management.

### Branching Schemes

The following schema details the branching scheme of the driver development. All development will be done on a development branch and released on a release branch. An integration branch can be used to solve development branch conflicts. One release branch will be created per hardware module on which the driver will be ported.

MCAL 3.0 and MCAL 3.2 products are developed in the same code base. The work for MCAL 3.2 will be stored in Clear Case on the same branch as for MCAL 3.0 for all text files, the differentiation between specific content being marked with the following defines:

**#if (MCAL\_ASR\_RELEASE==MCAL\_ASR\_3\_2)**

**#if (MCAL\_ASR\_RELEASE==MCAL\_ASR\_3\_0)**

**#if (MCAL\_ASR\_RELEASE!=MCAL\_ASR\_3\_2)**

**#if (MCAL\_ASR\_RELEASE!=MCAL\_ASR\_3\_0)**

All files that cannot accommodate all 3.0 and 3.2 content in the same version will have specific branches created for ASR3.2 content (e.g. design documents, quality reports, user manuals, etc).

The following files, which are prepared during MCAL 3.2 development, should be stored on ASR 3.2 specific branch:

**Build Environment files:**

* config\_beart.mak
* config\_tpb.mak
* release\_configuration.txt
* config\_common\_vars.mak
* variables\_common.xml

**Autosar module files:**

* AUTOSAR\_MCAL\_<Module>\_SDD.eap
* AUTOSAR\_MCAL\_<Module>\_TP.doc
* ch\_configuration\_parameters.xml/ <module>\_enable\_multi\_sync\_transmit\_param.xml
* <Module>NonAutosar.JPG/<Module>General.JPG
* All quality related files like test reports, Traceability Matrix, Misra Report, Profile Report, LDRA reports, etc.

#### For Autosar Drivers

##### For Generic files

We have the following branching scheme:

All Autosar 3.0 and 4.0 files stored in …/generic folders shall have the following branching scheme:

Where x.y from mcalx.y can be mcal3.0, mcal4.0

**Note:** No specific branch should be created for MCAL 3.2 ASR modules.

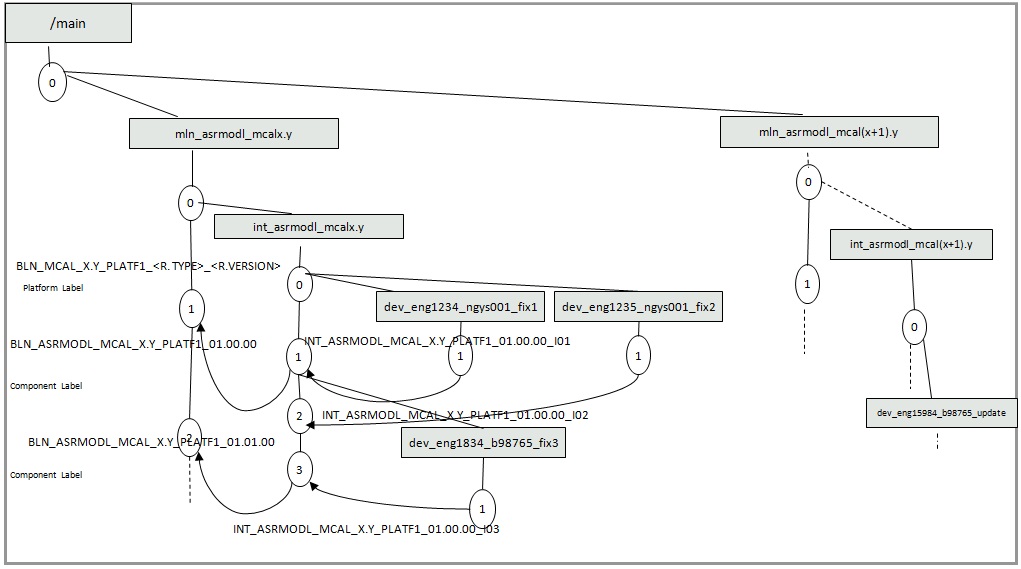
****

Figure 12 Autosar drivers – Generic Files Branching scheme

Note: This scheme is applicable also the SMCAL Development

##### For Specific files

We have the following branching scheme:

All Autosar 3.0 and 4.0 files stored in …/specific folders shall have the following branching scheme:

Where x.y from mcalx.y can be mcal3.0, mcal4.0

**Notes:**

No specific branch should be created for MCAL 3.2 ASR modules.

This scheme is applicable also SMCAL Development specific files

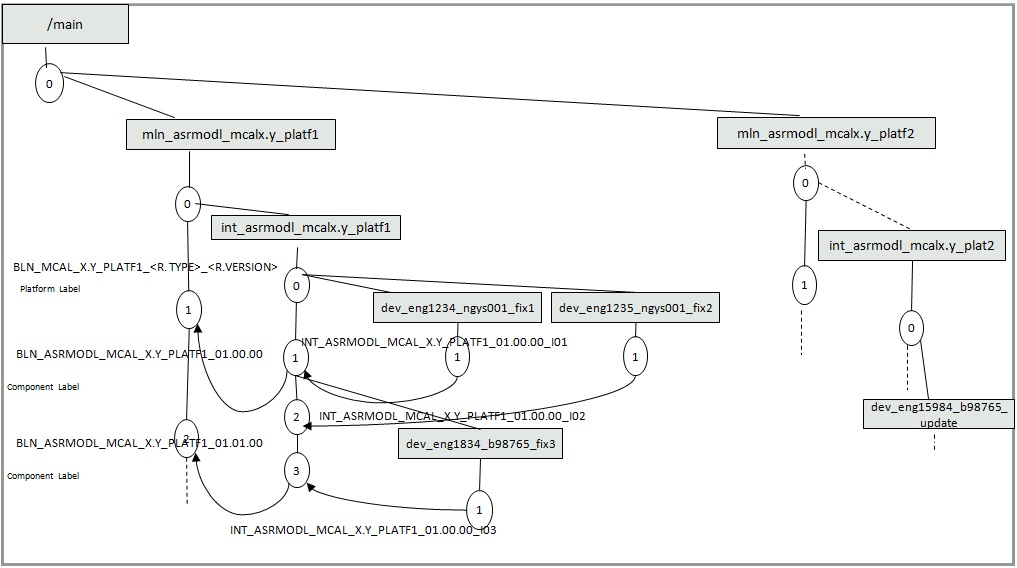


Figure 13 Autosar Drivers Specific Files Branches scheme

##### Doc Files

All Autosar 3.0, 4.0 and 3.2 files stores stored in …/doc folder shall have the following branching scheme.

Where x.y from mcalx.y can be mcal3.0, mcal4.0, mcal3.2.

This scheme is applicable also SMCAL Development Doc files

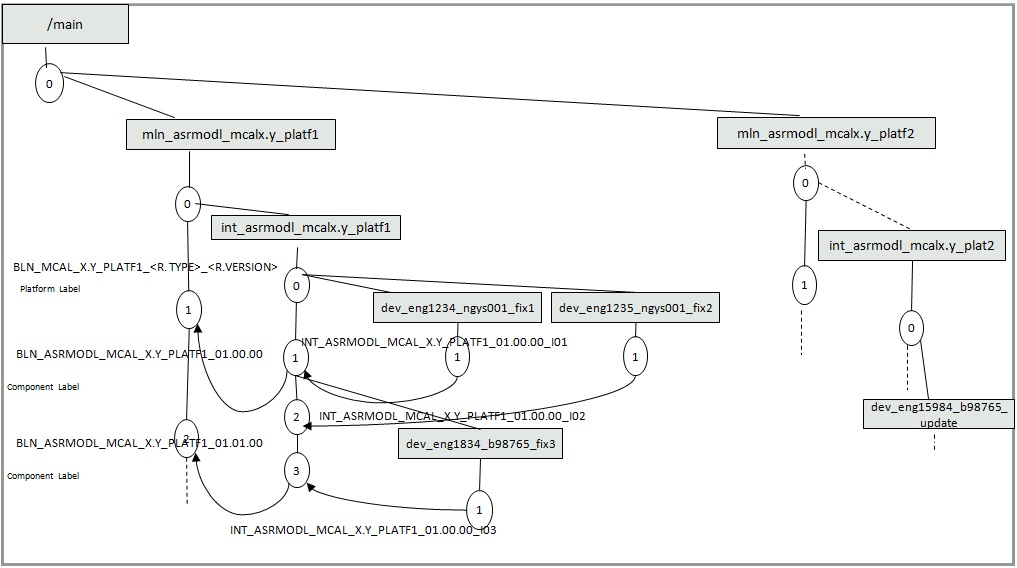


Figure 14 Branching scheme for doc files

##### For IPVault Drivers

We have the following branching scheme

All IPVault 3.0 and 4.0 files shall have the following branching scheme:

Where x.y from mcalx.y can be mcal3.0, mcal4.0

**Note:** No specific branch should be created for MCAL 3.2 ASR modules

This scheme is applicable also SMCAL Development IP Vault files

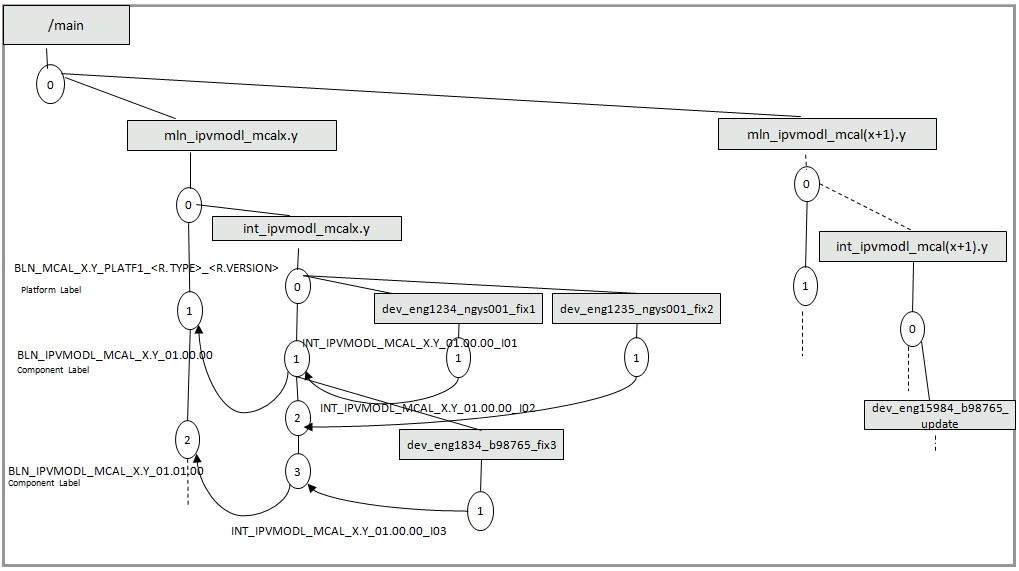


Figure 15 IPVault Drivers branching scheme

#### For Sample Applications Folder

All Autosar 3.0, 4.0 and 3.2, including SMCAL sample application files shall have the following branching scheme.

Where x.y from mcalx.y can be mcal3.0, mcal4.0, mcal3.2.

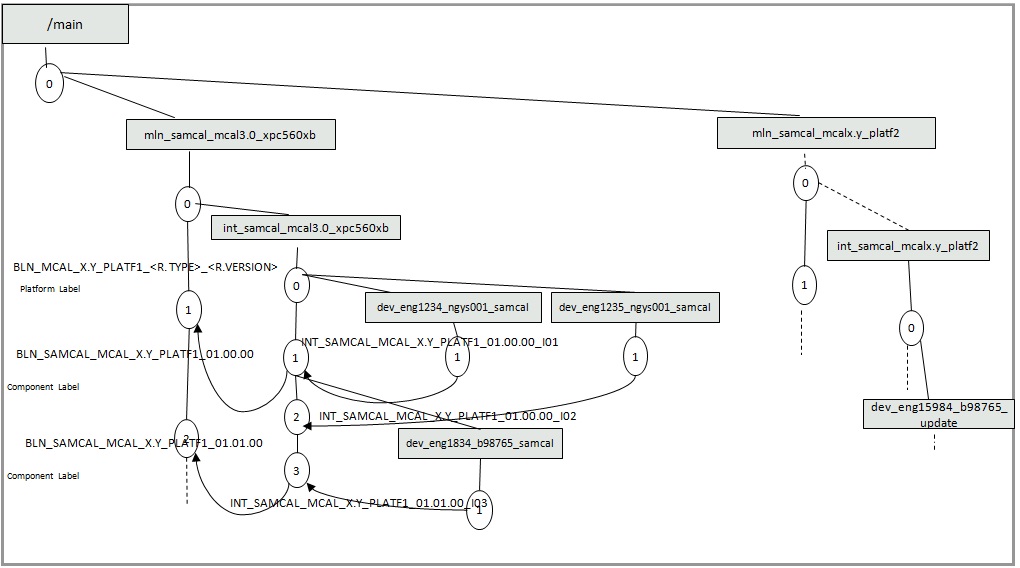


Figure 16 Sample Application Folder branching scheme

#### For Tests Folder

All Non-Safety Autosar 3.0, 4.0 test files – located either in MCAL software development VOBs, or in vnv\_test VOB - shall follow the below branching scheme and the two rules listed below

* Generic files shall have Autosar dependent and platform independent branches and shall follow the schema described in section 2.3.3.1.1
* Specific files shall have Autosar dependent and platform dependent branches and shall follow the schema described in section 2.3.3.1.2

Where x.y from mcalx.y can be mcal3.0, mcal4.0.

For MCAL3.2, build environment files listed in section 6.3 shall be 3.2 specific branches.

For the projects using ITG for test development, the test artifacts are managed in vnv\_test VOB.

ITG Configuration Management Process is described in an independent procedure located here:

<http://compass-phx.freescale.net/livelink/livelink/228862859/ITG_Configuration_and_Change_Management_Procedure.docx?func=doc.Fetch&nodeid=228862859>

#### For Tools Folder

See Tools Software Configuration Management Plan located here:

<http://www.freescale.com/go/222705358>

## Clearcase Labeling Strategy for MCAL /SMCAL Software

Project labels are created when a significant milestone is completed, or it is critical to capture significant work.

There are many types of labels:

* Component Labels
* Release Labels
* Private Labels
* Integration Labels

### Component Labels

Notes:

Clearcase Official Component label have the same name with ClearQuest Component baselines.

IP Modules will always have prefix \_IPV\_ in the name of the labels/baselines like:

BLN\_IPV\_FLEXCAN\_ MCAL\_3.0\_01.00.00

The naming convention for Autosar and IPVault drivers is different.

#### For IPVault Drivers

BLN\_<IP\_NAME>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<Baseline\_version>

Where:

* <IP\_NAME> - the name of the IP, example: IPV\_EMIOS
* <PRODUCT\_NAME> - MCAL for JSW/JSW\_ASR4 vobs and SMCAL for SASW vob
* <ASR\_VERSION> - the Autosar Version: Example: 3.0 or 4.0
* <Baseline\_version> - the version of the baseline/release of the following format:

xx.yy.zz where xx – major, yy-minor zz-patch (service release)

ex: 00.08.01

Note: Low-level drivers are NOT platform dependent, but Autosar dependent.

Examples:

BLN\_IPV\_FLEXCAN\_ MCAL\_3.0\_01.00.00

BLN\_IPV\_FLEXCAN\_ MCAL\_4.0\_01.00.00

BLN\_IPV\_FLEXCAN\_ SMCAL\_4.0\_01.00.00

#### For Autosar Drivers – JSW, JSW\_ASR4, SASW VOBs

This is the naming rule used for component labels applied over Autosar module source code, documentation and test folders in software development VOBs

BLN\_<ASR\_MODULE>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Baseline\_version>

Where:

* <ASR\_MODULE> - Autosar Module – the items in the *Table 5* and *Table 8*.
* <PRODUCT\_NAME> - MCAL for JSW/JSW\_ASR4 vobs and SMCAL for SASW vob
* <PLATFORM> - Platform Name, for current project: XPC56XXL
* <ASR\_VERSION> - the Autosar Version: Example: 3.0 or 4.0
* <Baseline\_version> - Baseline/Release Version of the type xx.yy.zz

Note: Autosar Drivers are platform dependent

Examples:

BLN\_ADC \_MCAL\_3.0\_XPC56XXL\_01.00.00

BLN\_CAN\_MCAL\_3.0\_MPC574XP\_01.00.00

BLN\_ADC \_MCAL\_3.2\_XPC56XXL\_01.00.00

BLN\_GPT\_MCAL\_3.0\_XPC564XA\_01.02.00

BLN\_ADC \_MCAL\_4.0\_ XPC564XA \_01.00.00

BLN\_ADC \_SMCAL\_4.0\_ MPC574XP \_01.00.00

#### For Autosar Drivers – vnv\_test VOB

This is the naming rule used for labels applied over test artifacts managed in vnv\_test VOB

BLN\_TEST\_<ASR\_MODULE>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Baseline\_version>

Where:

* <ASR\_MODULE> - Autosar Module – the items in the *Table 5* and *Table 8*.
* <PRODUCT\_NAME> - MCAL for JSW/JSW\_ASR4 vobs and SMCAL for SASW vob
* <PLATFORM> - Platform Name, for current project: XPC56XXL
* <ASR\_VERSION> - the Autosar Version: Example: 3.0 or 4.0
* <Baseline\_version> - Baseline/Release Version of the type xx.yy.zz

Note: Autosar Drivers are platform dependent

Examples:

BLN\_TEST\_ADC \_MCAL\_3.0\_XPC56XXL\_01.00.00

BLN\_TEST\_CAN\_MCAL\_3.0\_MPC574XP\_01.00.00

BLN\_TEST\_ADC \_MCAL\_3.2\_XPC56XXL\_01.00.00

BLN\_TEST\_GPT\_MCAL\_3.0\_XPC564XA\_01.02.00

BLN\_TEST\_ADC \_MCAL\_4.0\_ XPC564XA \_01.00.00

BLN\_TEST\_ADC \_SMCAL\_4.0\_ MPC574XP \_01.00.00

Note: Platform name is documented in Project Details.

#### For Stubs Modules

BLN\_<STUB\_MODULE>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<Baseline\_version>

Where:

* <STUB\_MODULE> - Stub Module
* <PRODUCT\_NAME> - MCAL for JSW/JSW\_ASR4 vobs and SMCAL for SASW vob
* <ASR\_VERSION> - the Autosar Version: Example: 3.0 or 4.0
* <Baseline\_version> - Baseline/Release Version of the type xx.yy.zz

Note: Stub Modules are not platform dependent

Examples:

BLN\_DET\_ MCAL\_4.0\_01.01.00

BLN\_DEM\_ MCAL\_3.0\_01.00.00

BLN\_DEM\_ SMCAL\_3.0\_01.00.00

Note: This rule is applied only for Stub Modules which are NOT platform dependent, but Autosar dependent. For platform dependent Stubs, naming convention *for Autosar Drivers* will be used.

#### For Sample Applications

Labels naming convention:

BLN\_<APPLICATION\_NAME>\_MCAL\_<ASR\_VERSION>\_<PLATFORM>\_<Baseline\_version>

Example:

BLN\_SAMCAL\_MCAL\_3.0\_ XPC56XXL\_01.00.00

BLN\_SAMCAL\_MCAL\_3.2\_ XPC56XXL\_01.00.00

BLN\_SAMCAL\_MCAL\_4.0\_ XPC56XXL\_01.01.00

Note: There is no sample application in SASW Vob

#### For Test Folder

The same rule as the in case of Autosar Drivers shall be used.

### Release Labels

#### Official Release Labels – JSW, JSW\_ASR4, SASW VOBs

The naming convention for Official Release label is the following

BLN\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Release\_Type>\_<Release\_Version>

Where:

<Release version> in this case in of the type: “x.y.z”

Example:

BLN\_MCAL\_3.0\_ BOLERO\_BETA\_0.9.1

BLN\_MCAL\_3.2\_BOLERO\_BETA\_0.9.0

BLN\_MCAL\_4.0\_ LEOPARD\_BETA\_0.9.0

BLN\_SMCAL\_4.0\_ LEOPARD\_BETA\_0.9.0

**The Release label is the label which incorporates all components labels in software development VOBs.**

Note:

For releases independently tested by ITG, the MCAL Release label will be applied also over the Release Label from the vnv\_test VOB

The Release label is the label which incorporates all components labels.

#### Official Release Labels – vnv\_test VOB

The naming convention for Official Release label is the following

BLN\_TEST\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Release\_Type>\_<Release\_Version>

Where:

<Release version> in this case in of the type: “x.y.z”

Example:

BLN\_TEST\_SMCAL\_4.0\_ PANTHER\_BETA\_0.9.0

The Release label is the label which incorporates all components labels.

#### Patch Release Labels

In the case of Patches, the following label naming convention will be used:

BLN\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Release\_Type>\_<Px><Release\_Version>

Where

Px is the Patch version: ex; P1, P2 ...

Example:

BLN\_MCAL\_3.0\_ LEOPARD\_BETA\_P1\_2.9.1

BLN\_MCAL\_3.2\_BOLERO\_BETA\_\_P1\_0.9.0

BLN\_MCAL\_4.0\_ BOLERO\_BETA\_P1\_0.9.1

BLN\_SMCAL\_4.0\_ BOLERO\_BETA\_P1\_0.9.1

#### Quality Package Labels

In case of Quality Package, the following label naming convention will be used:

QP\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Release\_Type>\_<Release\_Version>\_QP<QP\_Version>

Where:

* Release\_Type is one of the following three types:
  + EAR – Early Access Release
  + Beta
  + RTM – Ready to Market
* Pi –Patch Version
* Release Version – is of the following type: xx.yy.zz
* QP Version – is of the following type: x.y

Example:

QP\_MCAL\_3.0\_LEOPARD\_RTM \_1.0.0\_ QP0.1

QP\_MCAL\_3.2\_BOLERO\_BETA\_ 0.9.0\_QP0.1

QP\_MCAL\_4.0\_LEOPARD\_RTM \_1.0.0\_ QP0.1

QP\_SMCAL\_4.0\_LEOPARD\_RTM \_1.0.0\_ QP0.1

For cases when Quality Package reports are added after release is delivered, and all CRs and component baselines are already closed, **the following approach will be followed**:

* Open a component CR for Quality Package changes (quality reports corrections);
* The files will be placed in Clearcase on a development branch at least. Ideally this dev branch will be merged also to the integration branch (as a more “official” location of a reviewed file);
* A label will be applied (**PVT** or **INT**);
* This label will be communicated to the Integration Team that will apply the QP label on that exact version (dev or int branch).
* The component CR will be reviewed and Resolved, but NOT integrated.
* It will be linked to the next component baseline (current baseline + 1).

**Note**: In case several QPs will be needed, all such CRs will be linked to that next baseline (they will be integrated when that new baseline will be delivered...)

* The integration team will apply a track tag on all the CRs used for Quality Package changes. The tack tag will have the name of the QP label, following QP label naming convention (example: **QP\_MCAL\_3.0\_LEOPARD\_RTM \_1.0.0\_ QP0.1**).

### Private Labels

Private labels are used internally for code sharing. They will not be part of any release.

Naming convention:

PVT\_<Core\_ID>\_{Short\_text}

Examples:

PVT\_B23456\_AUTOMOTIVE SW\_FIX

### Integration Labels - JSW, JSW\_ASR4, SASW VOBs

The integration labels are used to track/get the changes on integration branches. They have to be applied only on integration branches. They are correlated to the component labels.

The naming convention for Integration labels is the following:

INT\_<ASR\_MODULE>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Baseline\_version>\_<Integration version> - for software development VOBs

INT\_TEST\_<ASR\_MODULE>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Baseline\_version>\_<Integration version> - for vnv\_test VOB

INT\_<IP\_NAME>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<Baseline\_version>\_>\_<Integration version>

* <ASR\_MODULE> - Autosar Module.
* <PRODUCT\_NAME> - MCAL for JSW/JSW\_ASR4 vobs and SMCAL for SASW vob
* <IP\_NAME> - the name of the IP, example: IPV\_EMIOS
* <PLATFORM> - Platform Name, for current project: XPC56XXL
* <ASR\_VERSION> - the Autosar Version: Example: 3.0, 4.0. or 3.2(only for autosar modules)
* <Baseline\_version> - Baseline/Release Version of the type xx.yy.zz
* <Integration\_version> - Incremental version to differentiate the merges on integration branch due to different CRs integration.: Example I01 for ENGR1234, I02 for ENGR1234 and ENGR12345

Example:

INT\_SPI\_MCAL\_3.0\_ XPC56XXL\_01.01.00\_I01 – Autosar Driver

INT\_TEST\_SPI\_MCAL\_4.0\_ MPC574XP\_01.01.00\_I01 – Autosar Driver Test Artifacts in vnv\_test VOB

INT\_IPV\_DSPI\_MCAL\_3.0\_01.01.00\_I01 – IPVault Driver

INT\_SPI\_MCAL\_3.2\_XPC560XB\_01.00.00\_I01– Autosar Driver

INT\_ADC\_MCAL\_4.0\_ XPC56XXL\_01.01.00\_I01 – Autosar Driver

INT\_ IPV\_ADC\_MCAL\_4.0\_01.01.00\_I01 – IPVault Driver

INT\_GPT\_SMCAL\_4.0\_XPC574XP\_01.00.00\_I03 – Autosar Driver

INT\_ IPV\_ADC\_SMCAL\_4.0\_01.01.00\_I01 – IPVault Driver

Note: We will not create ClearQuest baselines for *Private Labels* and *Integration Labels* cases.

## Compass Folders Structure

Process documentation to be created for each project is located on compass – Intranet for Freescale Only Projects, or on Extranet for project developed with 3rd parties.

Intranet Location:

<http://compass.freescale.net/livelink/livelink?func=ll&objId=225431206&objAction=browse&sort=name>

Extranet location:

<http://compass.freescale.net/livelink/livelink?func=ll&objId=214291877&objAction=browse&viewType=1>

One folder will be created for each project. The subfolders should be the same for every project, following the structure described bellow - following the Folder Template:

Note: Projects which need a different folder structure shall describe it in Project Configuration Management Plan.

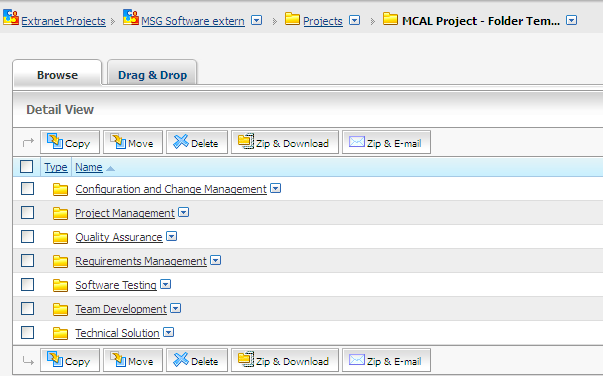


Figure 17 Compass Folder Structure

1. Configuration and Change Management Folder contains at minimum:

* Link to the MCAL procedure to be followed (this document)
* Project Configuration Management Plan
* Configuration Item List (for 3.0 and 4.0 projects)

1. Project Management Folder contains at minimum:

* Project Management Plan
* Project Schedule & WBS
* Risk Management Plan
* Project Status Meeting Minutes..
* Statement of Work

1. Quality Assurance contains at minimum:

* Quality Assurance Plan
* Metrics Plan and Report
* Internal Audits Reports – if planned for that’s specific project
* RRR and Supplier Monitoring Reviews
* Phase Gate Reviews

1. Requirements Management contains at minimum:

* Link to PRD, MRD
* Product Requirements Analysis Sheet

1. Software Testing contains at minimum:

* Test Strategy
* Customer configuration files- if available

1. Team Development contains at minimum:

* Link to MCAL Competency Matrix and Training Plan

1. Technical Solution contains at minimum:

* Link to MCAL Architecture document
* Link to Coding Rules and peer review checklists

## Errata Handling – Source Code

Same hardware errata has different IDs for each platform.In software, same implementation can be used for all platforms. This means that HW IDs can’t be used as selectors in source code.

**Errata Association**

The association between HW and SW IDs is stored in file:

*…JSW\_MCAL\XPC56xx\doc\erratas\AUTOSAR\_MCAL\_Errata* *Association.xlsx*

(in ASR4.0 case and SMCAL case)

*…JSW\_MCAL\XPC56xx\docs\erratas\AUTOSAR\_MCAL\_Errata* *Association.xlsx*

(in ASR 3.0 case)

The association contains the following fields:

**SW Errata ID**: There is a SW define built follow the rules ERR\_IPV\_<IP>\_xxxx, where the <IP> is the name of IP and xxxx is a progressive number. This is the ID to be used in source code.

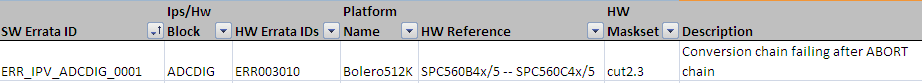
**IPs/Hw Block**: the name of HW IP affected.

**HW Errata IDs**: The PDM errata ERR00xxxx for ST errata, eyyyy for FSL errata. Usually xxxx is equal to yyyy.

**Platform Name**: Friendly name of silicon (Bolero/Komodo).

**HW Reference**: Official silicon name (MPC5602/3/4/x).

**HW Maskset**: Hardware cut/maskset version



**Usage in Source Code**

The SW errata IDs to be used for a current platform is extracted from the excel file based on filtering on platform name.

**All SW IDs will be placed in Soc\_Ip.h file as like #define with value STD\_ON.**

*Example for Komodo:*

**ERR\_IPV\_FLASH\_0005**

**ERR\_IPV\_FLEXCAN\_0002**

**ERR\_IPV\_FLEXPWM\_0002**

The Soc\_Ip.h will contain:

*#define ERR\_IPV\_FLASH\_0005 STD\_ON*

*#define ERR\_IPV\_FLEXCAN\_0002 STD\_ON*

*#define ERR\_IPV\_FLEXPWM\_0002 STD\_ON*

**Note!** A SW Errata ID define must NEVER EVER be defined as STD\_OFF. It should either be STD\_ON or not defined at all. (This is needed to simplify the source code usage - avoid the 2 “else” paths in the ifdefs)

The source code will be use the defines like below:

*/\* @errata ERR\_IPV\_FLEXCAN\_0003 \*/*

*#ifdef ERR\_IPV\_FLEXCAN\_0003*

*#if (ERR\_IPV\_FLEXCAN\_0003== STD\_ON)*

*…code…*

*#endif*

*#endif*

# Change Management Activities

MCAL - Autosar 3.0, 3.2, 4.0 – and SMCAL projects shall follow the change management rules described in this section.

These rules are applicable for both software development VOBs: JSW, JSW\_ASR4, SASW and Independent Testing VOB: vnv\_test

Project specific Change Management information, not covered by the current procedure, will be documented in Project Configuration Management Plan.

In the scope for this chapter are Change, Requests, Baselines, Track Tags, BINs Settings

There are four types of Change Requests:

a) New Works - request an enhancement on a component, a request for a new feature, to update and to track implementation progress for a planned requirement.

* New Planned/Unplanned Request (new feature) received from customer
* New Planned Request (new feature) submitted internally (software implementation, documentation, package creation)
* Internal Refactoring/Enhancement submitted internally or requested by the customer development team or acquired from 3rd parties
* Unplanned/Planned Request to make a change coming from the customer or submitted internally

b) Defects - used to report a defect on a project release or a component.

* Defects reported during and after modules integration, including integration testing, unit testing, acceptance testing – Internal Defects
* Defects reported by the Independent Test Group against an official baseline (testing performed on an official BLN type labels
* Defects reported by customer – External Defects
* Defects in software product, located both in components developed by product development team and acquired from 3rd parties
* Errors find during peer reviews and inspections that cannot be fixed before integration and their correction is postponed for further development phases
* Defects raised by Quality Engineer as result of the quality assessments

c) Integration Change Requests

Integration CRs are used to integrate two or more New-Work or Defect Change Requests, or to track changes propagation from one platform to another.

c) Inquiry Change Requests

Inquiries are used in two cases:

* When it is not known if the request is defect or new-work,
* To track questions, action items, support issues, etc.

This CR has a different workflow from the other three CR subtypes.

## Project Initiation Activities (NPI Planning Phase)

At the time of initiating a project, the Project Lead shall perform (ensure that) the following setup tasks related to Change Request Management processes for the project.

* In NPI Planning Phase initiate the Configuration Management Plan document for the project based on the template [[2](#_References)] and make all required project specific updates. This document is used only for the cases when project specific process need to be put in place. The current document being in scope for all MCAL Projects (3.0 and 4.0).
* Identify all Software Modules within the scope of this project as “Part Affected” ClearQuest BIN settings.
* Verify the platforms listed in the “Configs Affected” ClearQuest BIN settings.

+ Indentify the Change Control Board Team Members

In NPI Planning Phase, initiate the Change Control Board (CCB) for this project and schedule regular meetings for reviewing the CRs

Not: CCB from ClearQuest BIN settings can or cannot coincide with the Project CCC.

Change Request External Sources.

* Identify all Change Request (New-works and Defects) Sources and assign one or more team members to monitor these Sources on a regular basis in the table below.

|  |  |
| --- | --- |
| **Change Request Sources** | **Description** |
| Siebel | Customer Engineer submits Siebel tickets corresponding to customers problem requests, new enhancement requests or defects |
| Jira | For Elektrobit Tickets |

Table 8 External Requests Sources

At least one CQ Record will be associated to each Siebel/Jira ticket.

## MCAL BINs

1) MCAL 3.0 (AutoSar 3.0) and MCAL 3.2 (AutoSar 3.2) CRs are being submitted to **MCAL** BIN. To be able to change the BIN settings go to the ClearQuest Queries (“Navigator” pane) 🡪Public Queries 🡪MSG Bins & Projects 🡪MSG SW Automotive 🡪 MCAL 🡪 Settings 🡪 Bin Settings

2) MCAL 4.0 CRs are being submitted to **MCAL4.0** BIN.

To be able to change the BIN settings go to the ClearQuest Queries (“Navigator” pane) 🡪Public Queries 🡪MSG Bins & Projects 🡪MSG SW Automotive 🡪 MCAL4.0 🡪 Settings 🡪 Bin Settings

3) SMCAL CRs are being submitted to **SMCAL** BIN.

To be able to change the BIN settings go to the ClearQuest Queries (“Navigator” pane) 🡪Public Queries 🡪MSG Bins & Projects 🡪MSG SW Automotive 🡪 SMCAL 🡪 Settings 🡪 Bin Settings

4) Independent Test CRs are being submitted to **VV-TEST-MCAL** BIN.

To be able to change the BIN settings go to the ClearQuest Queries (“Navigator” pane) 🡪Public Queries 🡪MSG Bins & Projects 🡪MSG SW V&V 🡪 VV-TEST-MCAL 🡪 Settings 🡪 Bin Settings

The CCB members can then configure most of the other settings for the Bin, including updates to the list of CCB members and other team members.

The CQ administrator will setup:

* The Name and Classification of the bin
* Initial list of the bin’s privileged users (“CCB Members”)
* Physical location of the bin’s source and release artifacts
* Initial list of the bin’s Parts and Configurations

1. **Lists Tab**

In Lists tab – Parts List, the CCB is setting the list of parts which will be displayed as a list of for selection in the Affected Parts field of a CR.

The List of Configurations is set by the CCB in Configs List and will be displayed as a list for selection in the Configs Affected field of a CR.

The list of Parts and list of Configs is initially setup when the Bin is created, and then can be updated whenever is necessary.

Table 9 Parts Affected Table – MCAL BIN

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the “Parts Affected” | Description | Part Type | Remarks |
| ADC | Analog to Digital Converter | Autosar |  |
| BASE | Base | Autosar |  |
| BUILD\_EVD | Build environment and common files | Internal |  |
| DEM | Diagnostics Event Manager | Autosar | Stub, Platform Independent |
| CAN | Controller Area Network | Autosar |  |
| CANIF | CAN Interface | Autosar | Stub, Platform Dependent |
| CRY | Crypto Service Manager | Autosar |  |
| DET | Development Error Tracer | Autosar | Stub, Platform Independent |
| DIO | Digital I/O | Autosar |  |
| ECUM | Electronic Control Unit Manager | Autosar | Stub, Platform Independent |
| ETH | Ethernet | Autosar |  |
| ETHIF | Ethernet Infra |  |  |
| FEE | Flash EEPROM Emulation | Autosar |  |
| FLS | Flash | Autosar |  |
| FR | Flexray | Autosar |  |
| FRIF | Flexray Interface | Autosar |  |
| GPT | General Purpose Timer | Autosar |  |
| ICU | Input Capture Unit | Autosar |  |
| LIN | Local Interconnect Network | Autosar |  |
| LINIF | LIN Interface | Autosar | Stub, Platform Dependent |
| AUTOMOTIVE SW | Microcontroller Unit | Autosar |  |
| MEMIF | Memory Interface | Autosar | Stub, Platform Independent |
| PORT | PORT | Autosar |  |
| PWM | Pulse Width Modulation | Autosar |  |
| RESOURCE | Resource module | Tresos Specific |  |
| SAMPL\_APP | Resource module | Tresos Specific |  |
| SPI | Serial Peripheral Interface | Autosar |  |
| SCHM | SCHM | Autosar | Stub, Platform Independent |
| WDOG | Watchdog Timer | Autosar |  |
| WDOGIF | Watchdog Timer Interface | Autosar | Stub, Platform Dependent |
| SOC\_SPECIFIC | System on Chip | Internal |  |
| IPV\_ADC | Analog to Digital Converter | IPVault |  |
| IPV\_ADCSD | Analog to Digital Converter | IPVault |  |
| IPV\_CRP | Crypto | IPVault |  |
| IPV\_CSE | Crypto Security Module | IPVault |  |
| IPV\_CTU | Cross Triggering Unit | IPVault |  |
| IPV\_CTU2 | Cross Triggering Unit | IPVault |  |
| IPV\_DMAMUX | DMA Muxing | IPVault |  |
| IPV\_DMAV2 | Direct Memory Access | IPVault |  |
| IPV\_DFMPLL | Digital Frequency Modulation Phase Locked Loop | IPVault |  |
| IPV\_DSPI | Digital Serial Peripheral Interface | IPVault |  |
| IPV\_ECSM | Error Correction Status Module | IPVault |  |
| IPV\_EQADC | Enhanced Queued Analog-to-Digital Converter | IPVault |  |
| IPV\_EMIOS | Enhanced Modular IO Subsystem | IPVault |  |
| IPV\_ETIMER | Enhanced Timer | IPVault |  |
| IPV\_ESCI | Enhanced Serial Communication | IPVault |  |
| IPV\_INTC | Interrupt Controller | IPVault |  |
| IPV\_FLEXCAN | Flexible Controller Area Network | IPVault |  |
| IPV\_FLEXPWM | Flexible PWM | IPVault |  |
| IPV\_FLEXRAY | Flexray | IPVault |  |
| IPV\_FLASH | Flash | IPVault |  |
| IPV\_FLASH2 | Flash | IPVault |  |
| IPV\_FMPLL | Frequency Modulated Phase-Locked Loop | IPVault |  |
| IPV\_LINFLEX | Local Interconnect Network Flex | IPVault |  |
| IPV\_INTC | Interrupt Controller | IPVault |  |
| IPV\_QSPI | Quad Serial Peripheral Interface | IPVault |  |
| IPV\_MC | IPV Magic Carpet | IPVault |  |
| IPV\_MCV2 | IPV Magic Carpet | IPVault |  |
| IPV\_PIT | Periodic Interrupt Timer | IPVault |  |
| IPV\_PITRI | PIT Real Time Interface | IPVault |  |
| IPV\_RTC | Real Time Clock | IPVault |  |
| IPV\_SIU | System Integration Unit | IPVault |  |
| IPV\_SIUL | System Integration Unit Lite | IPVault |  |
| IPV\_SIULV2 | System Integration Unit Lite | IPVault |  |
| IPV\_SWT | Software Timer | IPVault | Stub, Platform Independent |
| IPV\_STM | System Timer Module | IPVault |  |
| IPV\_WKPU | Wake Up Unit | IPVault |  |
| ….. | …. | …. | …. |

Table 10 Parts Affected Table – MCAL4.0

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the “Parts Affected” | Description | Part Type | Remarks |
| ADC | Analog to Digital Converter | Autosar |  |
| BASE | Base | Autosar |  |
| BUILD\_EVD | Build environment and common files | Internal |  |
| DEM | Diagnostics Event Manager | Autosar | Stub, Platform Independent |
| CAN | Controller Area Network | Autosar |  |
| CANIF | CAN Interface | Autosar | Stub, Platform Dependent |
| CRY | Crypto Module | Autosar |  |
| DET | Development Error Tracer | Autosar | Stub, Platform Independent |
| DIO | Digital I/O | Autosar |  |
| ECUM | Electronic Control Unit Manager | Autosar | Stub, Platform Independent |
| ETH | Ethernet | Autosar |  |
| ETHIF | Ethernet Infra | Autosar |  |
| FEE | Flash EEPROM Emulation | Autosar |  |
| FLS | Flash | Autosar |  |
| FR | Flexray | Autosar |  |
| FRIF | Flexray Interface | Autosar |  |
| GPT | General Purpose Timer | Autosar |  |
| GTM | General Timer Module | Autosar |  |
| GTM\_CFG | General Configuration Timer Module | Autosar |  |
| ICU | Input Capture Unit | Autosar |  |
| IR | Integration and Release | Specific |  |
| LIN | Local Interconnect Network | Autosar |  |
| LINIF | LIN Interface | Autosar | Stub, Platform Dependent |
| AUTOMOTIVE SW | Microcontroller Unit | Autosar |  |
| MEMIF | Memory Interface | Autosar | Stub, Platform Independent |
| PORT | PORT | Autosar |  |
| PWM | Pulse Width Modulation | Autosar |  |
| RESOURCE | Resource module | Tresos Specific |  |
| RTE | Runtime Environment | Autosar |  |
| SAMPL\_APP | Resource module | Tresos Specific |  |
| SPI | Serial Peripheral Interface | Autosar |  |
| WDOG | Watchdog Timer | Autosar |  |
| WDOGIF | Watchdog Timer Interface | Autosar | Stub, Platform Dependent |
| …………… | ……………….. | ……….. | ………… |
| IPV\_ADC | Analog to Digital Converter | IPVault |  |
| IPV\_ADCDIG | ADC Digital Interface | IPVault |  |
| IPV\_CRC | Cyclic Redundancy Checker | IPVault |  |
| IPV\_CRP | Crypto | IPVault |  |
| IPV\_CSE | Crypto Security Module | IPVault |  |
| IPV\_CTU | Cross Trigger Unit | IPVault |  |
| IPV\_CTU2 | Cross Trigger Unit | IPVault |  |
| IPV\_DMA | Direct Memory Access | IPVault |  |
| IPV\_DMAMUX | DMA Muxing | IPVault |  |
| IPV\_DMAV2 | Direct Memory Access | IPVault |  |
| IPV\_DFMPLL | Digital Frequency Modulation Phase Locked Loop | IPVault |  |
| IPV\_DSPI | Digital Serial Peripheral Interface | IPVault |  |
| IPV\_EMIOS | Enhanced Modular IO Subsystem | IPVault |  |
| IPV\_ECSM | Extended Crypto Security Module | IPVault |  |
| IPV\_EQADC | Extended QADC | IPVault |  |
| IPV\_ETIMER | Enhanced Timer | IPVault |  |
| IPV\_ESCI | Enhanced Serial Communication | IPVault |  |
| IPV\_INTC | Interrupt Controller | IPVault |  |
| IPV\_FCCU | Fault Collection and Control Unit | IPVault |  |
| IPV\_FEC | Fault Error Correction | IPVault |  |
| IPV\_FEC | Fast Ethernet controller | IPVault |  |
| IPV\_FLEXCAN | Flexible Controller Area Network | IPVault |  |
| IPV\_FLEXPWM | Flexible PWM | IPVault |  |
| IPV\_FLEXRAY | Flexray | IPVault |  |
| IPV\_FLASH | Flash | IPVault |  |
| IPV\_FMPLL | Frequency Modulation Phase Locked Loop | IPVault |  |
| IPV\_LINFLEX | Local Interconnect Network Flex | IPVault |  |
| IPV\_QSPI | Quad Serial Peripheral Interface | IPVault |  |
| IPV\_MC | IPV Magic Carpet | IPVault |  |
| IPV\_PIT | Periodic Interrupt Timer | IPVault |  |
| IPV\_PITRTI | PIT Real Time Interface | IPVault |  |
| IPV\_RTC | Real Time Clock | IPVault |  |
| IPV\_SIU | System Integration Unit | IPVault |  |
| IPV\_SIUL | System Integration Unit Lite | IPVault |  |
| IPV\_SIULV2 | System Integration Unit Lite second version | IPVault |  |
| IPV\_SWT | Software Timer | IPVault | Stub, Platform Independent |
| IPV\_SSCM | System status and configuration module | IPVault |  |
| IPV\_STCU | Self-Test Control Unit | IPVault |  |
| IPV\_STM | System Timer Module | IPVault |  |
| IPV\_WKPU | Wake Up Unit | IPVault |  |
| …… | ……. | ….. | …… |

Table 10 Parts Affected Table – SMCAL

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the “Parts Affected” | Description | Part Type | Remarks |
| ADC | Analog to Digital Converter | Autosar |  |
| BASE | Base | Autosar |  |
| BUILD\_EVD | Build environment and common files | Internal |  |
| DEM | Diagnostics Event Manager | Autosar | Stub, Platform Independent |
| CAN | Controller Area Network | Autosar |  |
| CANIF | CAN Interface | Autosar | Stub, Platform Dependent |
| CRY | Crypto Module | Autosar |  |
| CRC | Cyclic Redundancy Check | HW Module |  |
| DET | Development Error Tracer | Autosar | Stub, Platform Independent |
| DIO | Digital I/O | Autosar |  |
| ECUM | Electronic Control Unit Manager | Autosar | Stub, Platform Independent |
| ETH | Ethernet | Autosar |  |
| ETHIF | Ethernet Infra | Autosar |  |
| FEE | Flash EEPROM Emulation | Autosar |  |
| FLS | Flash | Autosar |  |
| FR | Flexray | Autosar |  |
| FRIF | Flexray Interface | Autosar |  |
| GPT | General Purpose Timer | Autosar |  |
| GTM | General Timer Module | Autosar |  |
| GTM\_CFG | General Configuration Timer Module | Autosar |  |
| ICU | Input Capture Unit | Autosar |  |
| IR | Integration and Release | Specific |  |
| LIN | Local Interconnect Network | Autosar |  |
| LINIF | LIN Interface | Autosar | Stub, Platform Dependent |
| AUTOMOTIVE SW | Microcontroller Unit | Autosar |  |
| MCL | TBC | HW Module |  |
| MEMIF | Memory Interface | Autosar | Stub, Platform Independent |
| PORT | PORT | Autosar |  |
| PWM | Pulse Width Modulation | Autosar |  |
| RESOURCE | Resource module | Tresos Specific |  |
| RTE | Runtime Environment | Autosar |  |
| SAMPL\_APP | Resource module | Tresos Specific |  |
| SPI | Serial Peripheral Interface | Autosar |  |
| WDOG | Watchdog Timer | Autosar |  |
| WDOGIF | Watchdog Timer Interface | Autosar | Stub, Platform Dependent |
| …………… | ……………….. | ……….. | ………… |
| IPV\_ADC | Analog to Digital Converter | IPVault |  |
| IPV\_ADCDIG | ADC Digital Interface | IPVault |  |
| IPV\_CRC | Cyclic Redundancy Checker | IPVault |  |
| IPV\_CRP | Crypto | IPVault |  |
| IPV\_CSE | Crypto Security Module | IPVault |  |
| IPV\_CTU | Cross Trigger Unit | IPVault |  |
| IPV\_CTU2 | Cross Trigger Unit | IPVault |  |
| IPV\_DMA | Direct Memory Access | IPVault |  |
| IPV\_DMAMUX | DMA Muxing | IPVault |  |
| IPV\_DMAV2 | Direct Memory Access | IPVault |  |
| IPV\_DFMPLL | Digital Frequency Modulation Phase Locked Loop | IPVault |  |
| IPV\_DSPI | Digital Serial Peripheral Interface | IPVault |  |
| IPV\_EMIOS | Enhanced Modular IO Subsystem | IPVault |  |
| IPV\_ECSM | Extended Crypto Security Module | IPVault |  |
| IPV\_EQADC | Extended QADC | IPVault |  |
| IPV\_ETIMER | Enhanced Timer | IPVault |  |
| IPV\_ESCI | Enhanced Serial Communication | IPVault |  |
| IPV\_INTC | Interrupt Controller | IPVault |  |
| IPV\_FCCU | Fault Collection and Control Unit | IPVault |  |
| IPV\_FRIPIV10 | Flexray Interface |  |  |
| IPV\_FEC | Fault Error Correction | IPVault |  |
| IPV\_FEC | Fast Ethernet controller | IPVault |  |
| IPV\_FLEXCAN | Flexible Controller Area Network | IPVault |  |
| IPV\_FLEXPWM | Flexible PWM | IPVault |  |
| IPV\_FLEXRAY | Flexray | IPVault |  |
| IPV\_FLASH | Flash | IPVault |  |
| IPV\_FMPLL | Frequency Modulation Phase Locked Loop | IPVault |  |
| IPV\_LINFLEX | Local Interconnect Network Flex | IPVault |  |
| IPV\_QSPI | Quad Serial Peripheral Interface | IPVault |  |
| IPV\_MC | IPV Magic Carpet | IPVault |  |
| IPV\_PIT | Periodic Interrupt Timer | IPVault |  |
| IPV\_PITRTI | PIT Real Time Interface | IPVault |  |
| IPV\_RTC | Real Time Clock | IPVault |  |
| IPV\_SIU | System Integration Unit | IPVault |  |
| IPV\_SIUL | System Integration Unit Lite | IPVault |  |
| IPV\_SIULV2 | System Integration Unit Lite second version | IPVault |  |
| IPV\_STCU | TBC | IPVault |  |
| IPV\_STCUV2 | TBC | IPVault |  |
| IPV\_SSCM | TBC | v |  |
| IPV\_SWT | Software Timer | IPVault | Stub, Platform Independent |
| IPV\_SSCM | System status and configuration module | IPVault |  |
| IPV\_STCU | Self-Test Control Unit | IPVault |  |
| IPV\_STM | System Timer Module | IPVault |  |
| IPV\_WKPU | Wake Up Unit | IPVault |  |
| …… | ……. | ….. | …… |

Table 10 Parts Affected Table – VV-TEST-MCAL

Note: Autosar Drivers are listed in “Parts affected”.

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the “Parts Affected” | Description | Part Type | Remarks |
| ADC | Analog to Digital Converter | Autosar |  |
| CAN | Controller Area Network | Autosar |  |
| CRC | Cyclic Redundancy Check | HW Module |  |
| DIO | Digital I/O | Autosar |  |
| ETH | Ethernet | Autosar |  |
| FEE | Flash EEPROM Emulation | Autosar |  |
| FLS | Flash | Autosar |  |
| FR | Flexray | Autosar |  |
| GPT | General Purpose Timer | Autosar |  |
| GTM | General Timer Module | Autosar |  |
| ICU | Input Capture Unit | Autosar |  |
| LIN | Local Interconnect Network | Autosar |  |
| AUTOMOTIVE SW | Microcontroller Unit | Autosar |  |
| PORT | PORT | Autosar |  |
| PWM | Pulse Width Modulation | Autosar |  |
| SPI | Serial Peripheral Interface | Autosar |  |
| WDG | Watchdog Timer | Autosar |  |

**Note:**

1. A Change Request (new feature, or defect) affecting both the Autosar Module and IP Module, will be created for the Autosar Module, then will be cloned to be used for tracking IP Module changes.

In the Analysis phase of the CR, the Analysis champ shall create a clone for each affected IP Module.

2) In terms of Clearcase Branching creation, there will be the following approach:

There is going to be created only one development branch for both Autosar and IP Modules, having in the branch naming convention, the parent CR ID.

Table 11 Configs Affected Table – MCAL BIN

|  |  |
| --- | --- |
| Name of the “Configs Affected” | Description |
| MPC560XP\_3.0 | MCAL 3.0 Pictus Product |
| MPC563XM\_3.0 | MCAL 3.0 Monaco Product |
| MPC564XB\_3.0 | MCAL 3.0 Bolero 3M Product |
| MPC5668X\_3.0 | MCAL 3.0 Fado Product |
| MPC567XF\_3.0 | MCAL 3.0 Mamba Product |
| MPC56XXA\_3.0 | MCAL 3.0 Andorra Product |
| MPC56XXB\_3.0 | MCAL 3.0 Bolero 512k/1.5M Product |
| MPC56XXL\_3.0 | MCAL 3.0 Leopard Product |
| MPC56XXK\_3.0 | MCAL 3.0 Komodo Product |
| MPC56XXB\_3.2 | MCAL 3.2 Bolero Product |
| MPC564XB\_3.2 | MCAL 3.2 Bolero 3M Product |
| MPC5744P\_3.0 | MCAL 3.0 Panther Product |
| SPC56EL70\_3.0 | MCAL 3.0 Leopard 2M Product |

Table 12 Configs Affected Table – MCAL4.0 BIN

|  |  |
| --- | --- |
| Name of the “Configs Affected” | Description |
| MPC560XP\_4.0 | MCAL 4.0 Pictus Product |
| MPC567XR\_4.0 | MCAL 4.0 Cobra Product |
| MPC564XB\_4.0 | MCAL 4.0 Bolero 3M Product |
| MPC574XMK\_4.0 | MCAL 4.0 McKinley Product |
| MPC56XXA\_4.0 | MCAL 4.0 Andorra Product |
| MPC56XXB\_4.0 | MCAL 4.0 Bolero 512k/1.5M Product |
| MPC56XXL\_4.0 | MCAL 4.0 Leopard Product |
| MPC56XXK\_4.0 | MCAL 4.0 Komodo Product |
| MPC5744P\_4.0 | MCAL 4.0 Panther Product |
| MPC574MK\_4.0 | MCAL 4.0 McKinley Product |

Table 12 Configs Affected Table – SMCAL BIN

|  |  |
| --- | --- |
| Name of the “Configs Affected” | Description |
| MPC5643L | SMCAL Leopard Product |
| MPC574XP | SMCAL Panther Product |
| MPC577XN | SMCAL Racerunner Product |
|  |  |

Table 16 Configs Affected Table – VV-TEST-MCAL BIN

|  |  |
| --- | --- |
| Name of the “Configs Affected” | Description |
| MPC5643L | SMCAL Leopard Product |
| MPC574XP | SMCAL Panther Product |
| MPC577XN | SMCAL Racerunner Product |
|  |  |

Contains all platforms in scope for independent testing.

1. **CCB – Team Tab**

Initial list of CCB members is configured when the Bin is created in CCB – Team Tab -> *CCB Members* field

The rest of the team who will access the records of the Bin is set by any CCB member in *Other Team Members* field.

The list of CCB members and the list of team members can be updated at any time as long as the Bin is Active.

The CCB Team will not be described in this procedure due to the frequent update of the team members

1. **CR Roles Tab**

In CR Roles Tab the CCB members can configure the following CR roles – Not Applicable for MCAL development.

* Designated Integrators
* Designated Verifiers
* Designated Closers

1. **CR Assignees Tab**

CR Assignees options are:

* *CR Assignee - New state*: Users to whom CRs in New state are assigned, if no Dev Champ is specified. When blank, CCB members have joint responsibility for CRs in New state. MCAL Bins current option is Blank.
* *CR Assignee - CCB state*: Users to whom CRs in CCB state are assigned. When blank, CCB members have joint responsibility for CRs in CCB state.

MCAL Bins current option is Blank

* *CR Assignee - Resolved state (not intg'd):* Users to whom CRs in Resolved state are assigned, if the Integrated flag is not set and no Verification Champ is specified. When blank, CCB members have joint responsibility for CRs in Resolved state and not integrated.

MCAL Bins current option is Blank

* *CR Assignee - Resolved state (intg'd):* Users to whom CRs in Resolved state are assigned, if the Integrated flag is set and no Verification Champ is specified. When blank, CCB members have joint responsibility for CRs in Resolved state and integrated.

MCAL Bins current option is Blank

* *CR Assignee - Verified state:* Users to whom CRs in Verified state are assigned. When blank, CCB members have joint responsibility for CRs in Verified state.

MCAL Bins current option is Blank

1. **CR Templates Tab**

The templates which will be used for some of the CR fields can be set by any CCB member in CR Templates menu.

The following templates are enforced by the BIN settings for both MCAL bins.

“**Description” Field** will use the following template:

**For New Work CRs:**

*NewWork Classification: (internal task, improvement, feature request)*

*[...]*

*In case for feature request, does a TWG ticket exist (cloned or linked)? TBC (Yes/No)*

*NewWork Description:*

*[…]*

*Expected behavior:*

*[…]*

*Requirement source:*

*[…]*

*(e.g. cPRD, gMRD, Customer Request, Quality, ASR SWS, RM.pdf, Errata.pdf...)*

**For Defect CRs:**

*Problem detailed description (how to reproduce it):   
[...]   
Preconditions:   
[...]   
Observed behavior:   
[...]   
Reported release baseline:   
[...]   
When can it be observed? (at configuration time, at runtime, at compile time?)   
[...]   
Expected behavior:   
[...]   
Proposed solution (Optional):   
[...]*

**For** **Inquiry CRs:**

*Initial Description:*

*[…]*

*(Note: After the CR type is decided, will remain only one section from the two listed below, the other one will be deleted)*

*Problem detailed description (how to reproduce it):*

*[...]*

*Preconditions:*

*[...]*

*Observed behavior:*

*[...]*

*When can it be observed? (at configuration time, at runtime, at compile time?)*

*[...]*

*Expected behavior:*

*[...]*

*Proposed solution (Optional):*

*[...]*

*NewWork Classification: (internal task, improvement, feature request)*

*[...]*

*In case for feature request, does a TWG ticket exist (cloned or linked)? TBC (Yes/No)*

*NewWork Description:*

*[…]*

*Expected behavior:*

*[…]*

*Requirement source:*

*[…]*

*(e.g. cPRD, gMRD, Customer Request, Quality, ASR SWS, RM.pdf, Errata.pdf...)*

**“Analysis Report”** Field will use the following template:

**For MCAL bin:**

*================== JDP Projects ==================   
Analyzed By JDP on all platforms (Bolero 3.0, Bolero 3.2, Leopard, Monaco, Andorra, Bolero3M 3.0, Bolero3M 3.2, , Pictus, Komodo): TBC (Yes/No)   
Did you review and update, if necessary, the Configs Affected with all JDP affected platforms: TBC (Yes/No)   
For ALL/GENERAL CRs, did you review the impacted IPs and if necessary, did you update the Parts Affected and create additional clones - for all affected platforms TBC (Yes/No)   
Effort Estimation: TBC   
===================================================   
================== FSL non-JDP 1 Projects ==================   
Analyzed By FSL non-JDP 1  on all platforms (Mamba, Fado): TBC (Yes/No)   
Did you review and update, if necessary, the Configs Affected with all FSL non-JDP 1 affected platforms: TBC (Yes/No)   
For ALL/GENERAL CRs, did you review the impacted IPs and if necessary, did you update the Parts Affected and create additional clones - for all affected platforms TBC (Yes/No)   
Effort Estimation: TBC   
===================================================   
================== FSL non-JDP 2 Projects =================   
Analyzed By FSL non-JDP 2  on all platforms (Panther): TBC (Yes/No)   
Did you review and update, if necessary, the Configs Affected with all FSL non-JDP 2 affected platforms: TBC (Yes/No)   
For ALL/GENERAL CRs, did you review the impacted IPs and if necessary, did you update the Parts Affected and create additional clones - for all affected platforms TBC (Yes/No)   
Effort Estimation: TBC   
===================================================   
================== ST non-JDP Projects====================   
Analyzed By ST non-JDP on all platforms (Leopard 2M): TBC (Yes/No)   
Did you review and update, if necessary, the Configs Affected with all ST non-JDP affected platforms: TBC (Yes/No)   
For ALL/GENERAL CRs, did you review the impacted IPs and if necessary, did you update the Parts Affected and create additional clones - for all affected platforms TBC (Yes/No)   
  
Effort Estimation:TBC   
  
Note:     
1.CRs affecting one platform only (like Integration CRs, new-works like HF clones, or generic release preparation CRs..) would need to have the analysis performed once by one team, and skipped by the rest of the teams. The person who is performing the analysis for the affected config, will add N/A for the rest the projects.   
2. Please make sure that you add/ remove only the Configs Affected of the platforms in your portfolio   
  
Root Cause Details:   
TBC (NA for NewWork)   
  
Proposed Solution:   
TBC*

**For MCAL4.0 bin:**

*================== JDP Projects ==================  
Analyzed By JDP on all platforms (Leopard, Bolero 3M, Andorra, Pictus , McKinley): TBC (Yes/No)  
Did you review and update, if necessary, the Configs Affected with all JDP affected platforms: TBC (Yes/No)  
For ALL/GENERAL CRs, did you review the impacted IPs and if necessary, did you update the Parts Affected and create additional clones - for all affected platforms TBC (Yes/No)  
Effort Estimation:TBC  
===================================================  
  
================== FSL non-JDP 2 Projects==================  
Analyzed By FSL non-JDP 2 on all platforms (Cobra, Panther, Komodo): TBC (Yes/No)  
Did you review and update, if necessary, the Configs Affected with all FSL non-JDP 2 affected platforms: TBC (Yes/No)  
For ALL/GENERAL CRs, did you review the impacted IPs and if necessary, did you update the Parts Affected and create additional clones - for all affected platforms TBC (Yes/No)  
Effort Estimation:TBC  
  
Notes:    
1.CRs affecting one platform only (like Integration CRs, new-works like HF clones, or generic release preparation CRs..) would need to have the analysis performed once by one team, and skipped by the rest of the teams. The person who is performing the analysis for the affected config, will add N/A for the rest the projects.  
2. Please make sure that you add/ remove only the Configs Affected of the platforms in your portfolio  
===================================================  
Root Cause Details:  
TBC (NA for NewWork)  
  
Proposed Solution:  
TBC*

**For SMCAL bin:**

*Analyzed for all platforms (Leopard, Panther, Racerunner):TBC (Yes/No)   
Did you review and update, if necessary, the Configs Affected: TBC (Yes/No)   
Effort Estimation: TBC (days)   
  
Root Cause Details: TBC (NA for NewWork)   
Proposed Solution: TBC   
Safety features impacted: TBC(Yes/No)   
Impacted Deliverables:   
•    Requirements Analysis (SWRA) : TBC (Yes/No)   
•    UML Design: TBC (Yes/No)   
•    Safety Analysis Report: TBC (Yes/No)   
•    Source Code: TBC (Yes/No)   
•    Unit Test Specification/ Code: TBC (Yes/No)   
•    Validation Test Specification/ Code: TBC (Yes/No)   
•    Safety Manual: TBC (Yes/No)   
•    Integration/User Manual: TBC (Yes/No)*

**ALL BINS:**

**“CR Test Report”** will use the following template:

*List the Test Case IDs: TBC*

*eg <TP01\_TC001> …*

*OR:*

*Peer Review (only if testing is not possible): TBC (Yes/No)*

**CR Release Note Text** will use the following template:

*<Description of changes,   
e.g. Check was added to …   
The function xyz was modified because …>   
  
Files Modified:   
<e.g. Adc\_Irq.c >   
<…>*

**6) CR Permissions Choices – Permission Tab**:

|  |  |
| --- | --- |
| **Permission:** | **Required to…** |
| Reclassify the subtype, severity, and priority of CRs | * Determines who has permission to change the Subtype, Severity and Priority of CRs in a bin. Possible options are: CCB, Team, Word.   Current option is **CCB.** |
| Move Inquiry CR to Analysis state | * Specifies who is allowed to move Inquiry CRs to the Analysis state * Change the Analysis Champ and Analysis Due Date   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move non-Inquiry CRs to Analysis state | * Specifies who is allowed to move non-Inquiry CRs to the Analysis state * Change the Analysis Champ and Analysis Due Date   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move Defect CR  to Open state | * Specifies who is allowed to move Defect CRs to the Open state * Change the Dev Champ, Target Baseline, Peer Review Requirement, and Designated Reviewer   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move New work CR  to Open state | * Specifies who is allowed to move New work CRs to the Open state * Change the Dev Champ, Target Baseline, Peer Review Requirement, and Designated Reviewer   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move Integration CR  to Open state | * Specifies who is allowed to move Integration CRs to the Open state * Change the Dev Champ, Target Baseline, Peer Review Requirement, and Designated Reviewer   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move non-Inquiry CRs to NoChanges state | * Specifies who is allowed to move non-Inquiry CRs to the NoChanges state   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move Inquiry CRs from New/CCB to NoChanges state | * Specifies who is allowed to move Inquiry CRs from New/CCB to the NoChanges state   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move Inquiry CRs from Analysis to NoChanges state | * Specifies who is allowed to move Inquiry CRs from the Analysis state to NoChanges state   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move any CR from New to NoChanges state | * Move CRs to the NoChanges state * Forward CRs to other bins (from any state except New)\*   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move any CR from Analysis/Open to CCB state | * Move CRs to CCB state * Move CRs that are in New state or any other state   Possible options are: CCB, Team, Word.  Current option is **Word.** |
| Move CR directly from Resolved to Closed (no verify) | * Specifies who is allowed to close CRs from Resolved state without passing through any other state;   Possible options are: CCB only, All Closer, Nobody.  Current option is **All Closers.** |
| Move CR directly from NoChanges to Closed (no verify) | * Specifies who is allowed to close CRs from No Changes state without passing through any other state;   Possible options are: CCB only, All Closer, Nobody.  Current option is **All Closers.** |
| Edit CR headline, description, etc. of CRs in Close state | * Restricts who is allowed to edit CRs that have been moved to Closed state.   Possible options are: CCB, Team, Word, Nobody.  Current option is **Word** |
| Attach or Detach This Bin as a CR Tag | * Restrict who is allowed to attached or detach this bin as a CR tag   Possible options are: CCB, Team, Word.  Current option is **Word** |
| Allow Designated Integrators to Manage Baselines | * Allows Designated Integrators to manage Baselines; * The Designated Integrators are set in CR Roles tab;   Options are: Yes or No  By default this option is set on **Yes** |
| Allow Staging Champ to Finalize Baselines | * Allows Staging Champ to Finalize Baselines, from Stage state   Options are: Yes or No  By default this option is set on **Yes** |
| Allow Testing Champ to Finalize Baselines | * Allows Testing Champ to Finalize Baselines, from Test state   Options are: Yes or No  By default this option is set on **Yes** |
| Allow CCB Members to Reopen Finalized Baselines | * Allows CCB members to reopen baselines that are in Test or Finalized state   Options are: Yes or No  By default this option is set on **No** |

Table 13 Permissions Table

The CCB is responsible for setting the permission fields. The default value is World for most of the options, but the CCB can change the values to CCB or Team. See the table above for a description of each of the permissions. World means any ClearQuest user.

Permissions always reserved to CCB Members only:

* Modify the BIN settings (except fields reserved for admins)
* Modify the CR Subtype, Severity, Priority fields
* Create and manage Baseline records (except as delegated to Designated Integrators, Staging Champ, and/or Testing Champ)
* Request changes to the bin’s lists and subscriptions
* Request custom public queries in the bin’s public folders
* Permissions always allowed for All Users:
* View all records in the ClearQuest database
* Submit CRs to any active bin in the ClearQuest database
* Clone CRs and forward them to other bins from the New state
* Modify most CR fields (except as explicitly mentioned above or on the previous slide)

**7)** **BIN Tailoring is managed through CR Behavior and Other Behavior tabs**

|  |  |
| --- | --- |
| **Section** | **Description** |
| Require Release Note Info upon existing Analysis state | * A value will be required in the Release Notes Y/N field as CRs are exiting the Analysis state * Options are Yes or No * Current option is **Yes** |
| Require Resolution Due Date in Open State | * Enable/disable Resolution Due Date request for CRs in open state * Options are Yes or No * Current option is **No** |
| Require Target Baseline in Open state | * Enforces Target baseline filed to be filled in Open State * Options are Yes or No * Current option is **Yes** |
| Default Peer Review Requirement | * The default Peer Review Requirement filled in when CRs are moved to the Open state (can be overridden on a CR basis) |
| Enforce Peer Review | * Enable/disable peer review enforcement. Current option is **Yes**. * When enabled the peer review must be completed before the CR is resolved. |
| Enable 'RdyForInt' Status in Open state | * Enable/disable Ready for Integration status to be available in Open state. This option allows the * Options are Yes or No * Current option is **No** |
| Auto-integrate CRs on entering Resolved state | * Causes the Auto Integration of the CRs that enter Resolve state; * Options are Yes or No * Current option is **No** |
| Auto-set Resolved-in Baseline to Target Baseline | * Causes the Resolved-in baseline to automatically get set to the Target Baseline when the CR is resolved; * Options are Yes or No * Current option is **Yes** |
| Require Test Report on entering Resolved state | * Enforces Test Report field to be filed-in on entering Resolved state * Options are Yes or No * Current option is **Yes** |
| Require Release Note Info on entering Resolved state | * Enforces Release Note field to be filed-in on entering Resolved state * Options are Yes or No * Current option is **Yes** |
| Require 'Delta Lines' on entering Resolved state | * Enforces Delta Lines field to be filed-in on entering Resolved state * Options are Yes or No * Current option is **Yes** |
| Require a Related CR when Resolved Indirectly | * A related CR link will be required for CRs that are closed with No Changes and Resolution Resolved Indirectly * Options are Yes or No * Current option is **Yes** |
| Auto-Set Release Note Info when External ID present | * The Release Notes Y/N field will be set to Yes whenever a value is enter in the Eternal ID field of a CR * Options are Yes or No * Current option is **Yes** |
| Auto-close CRs in the Resolved state | * Automatically move a CR in Close state when is routed to Resolved state; * Options are Never, Immediately, After 2 weeks inactivity, Upon baseline finalized * Current option is **Immediately** |
| Auto-close CRs in the Verified state | * Automatically move a CR in Close state when is routed to Verified state; * Options are Never, Immediately, After 2 weeks inactivity, Upon baseline finalized * Current option is **Immediately** |
| Auto-close CRs in the NoChanges state | * Automatically move a CR in Close state when is routed to NoChanges state; * Options are Never, Immediately, After 2 weeks inactivity, Upon baseline finalized * Current option is **Immediately** |
| Attach Bin Tag to CRs on Forward to Another Bin | * When a CR that has a Bin Tag attached is forwarded to another Bin, the Bin Tag will be copied to the forwarded CR; * Options are Yes or No * Current option is **Yes** |
| Propagate Bin Tag when a Tagged CR is Cloned | * When a CR that has a Bin Tag attached is cloned, the Bin Tag will be copied to the clone CR; * Options are Yes or No * Current option is **Yes** |

Table 14 BIN Tailoring Table

Other Tailoring options, not applicable to MCAL bins:

* Require Resolved-in Baseline for NoChanges Dispositions;
* Disable use of Branch and Change Set fields;
* Disallow use of Open and Resolved states
* Require verify before close when Verification Champ set
* Require Origin Build on CR Submit
* Allow Entry of Build Numbers;
* Allow Incoming PSRs from DesignPDM
* Require Test Report to Finalize Baseline;
* Disable use of Baseline Config Spec fields;
* Disallow Creation of New Baselines;
* Default Destination Bin for Forwarded CRs;
* Default Affected Configs for Forwarded CRs;
* Enforce Regex Pattern for Baseline Names;
* Enforce Regex Pattern for CR Branch Name field;
* Enforce Regex Pattern for CR Change Set values;
* Input String (for testing regex patterns above);
* Allowed Choices for Meeting 'Purpose' field;
* Disable use of Branch and Change Set fields;
* Disallow use of Open and Resolved states;

**8)** **Notifications – *Notification* tab:**

In the **Notifications** tab will define who will receive an e-mail notification under the various conditions listed. The conditions are

* CRs (submit)
* CRs (all modifications)
* CRs (state ‘New’)
* CRs (state ‘Resolved’)
* Baselines (all modifications)

Roles involved here are:

* + CCB Members
  + Team Members
  + Integrators
  + Testers

The **Notification** options are disable for both MCAL and MCAL4.0 bins.

**9) Access Controls Tab:**

In Access Control Tab are configured the Security Context values applicable for a Bin. This Tab can be managed only by CQ Admins.

The Security Context determines who is allowed to see a record (CR, Baseline).

* *Allowed Security Contexts*: List of Security Contexts which may be used to control visibility of records in a bin. The list of Security Context is displayed in a list for selection in the Security Context field of a CR
* *Default Security Context for CRs*: The default security context for new CRs in MCAL bins is FSL-INTERNAL. The FSL-INTERNAL value means the CRs are visible for Freescale employees Freescale contractors only and is not visible to partners or customers.
* *Security Context for Baselines:* The security context that will be used for all baselines in a bin.

**10)** **Other tabs:**

* *Attachments:* add file attachments to the Bin
* *History:* contains the history of operations on the Bin

## Change Requests

### Change Request Lifecycle

The CR Lifecycle with its corresponding fields are applicable for all CR types.

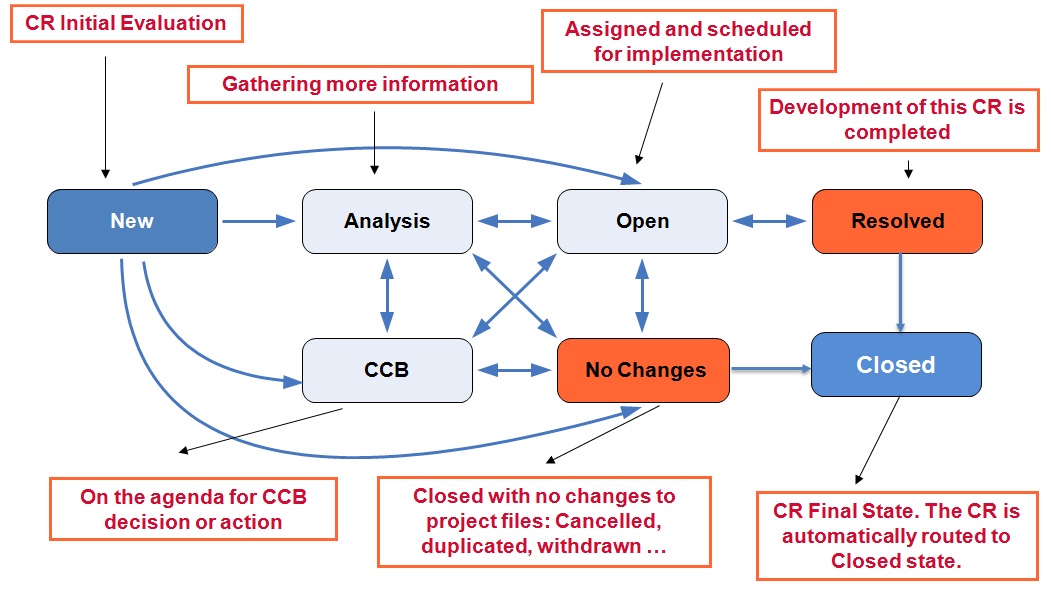


Figure 18 Change Request Lifecycle

### Submitting a Change Request (NEW State)

A Change Request can be submitted by anyone.

The table below contains fields to be filled-out when submitting a new change request—“New CR”

button. The fields are specific for the “New” State.

Note:

Mandatory fields are highlighted in RED/Italic in ClearQuest

Table 15 “New” State ClearQuest Fields

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Tab Name | Mandatory (Y/N/Read-only) |
| Business Unit | The Business Unit where the BIN, in which you want to submit the CR, is included.  Select the Business Unit. Ex: MSG BINs and Projects – for MSG Projects | Main | Y |
| Project Area | The Project Area where the BIN, in which you want to submit the CR, is included.  Select the Project Area: MSGAutomotive | Main | Y |
| Parent BIN | The BIN where the CR belongs to.  Select Parent BIN.  Ex: MCAL, MCAL 4.0, SW-GRAPHICS, CORE-SELF-TEST | Main | Y |
| Headline | Summary information to describe the issue. Rules and recommendations to enter the headline are:   * Always write the headline in understandable way. * Always use full phrases: No one-word headlines. * Be specific: The headline should uniquely reflect the defect. * Format should be for all headlines. | Main | Y |
| Description | Detailed Description of Change Request  A clear description is required for every newly submitted Defect to describe its purpose.  The reason for requested the change is included  Thesubmitter is responsible to include all relevant details that may help the champion in charge of the analysis to understand the work to do.  CR Description templates shall be used. See BINs Chapter. | Main | Y |
| State | For the Workflow “New” current state, the State is   * New   Note: This is a read only field. Will be filled-out automatically at CR submission. | Main | Read-Only |
| Status | For the Workflow “New” current state, the Status is   * New   Note: This is a read only field. Will be filled-out automatically at CR submission. | Main | Read-Only |
| Subtype | It is the field which selects the type:   * New Work - new feature, enhancement * Integration * Inquiry * Defect | Main | Y |
| Severity | **S1 – S5 (see table below)** | Main | Y only for Defects; N/A for New-Works |
| Priority | **P1-P4 (see table below)**  The default option is P3. | Main | Y |
| Security Context | This field will be automatically filled with the security context set in BIN settings | Main | Y |
| Requested/ Found By (role) | Organization who reports the defect Allowed values:   * Design/Dev (SW) * Design/Dev (HW) * Design/Dev (IC) * Design/Dev (Platform) * Test/Validation (SW) * Test/Validation (HW) * Test/Validation (IC) * Test/Validation (Platform) * SW/Platform Build & Release * Customer-External * Customer-FSL Internal * Customer/Customer Support * Field/Apps Engineering * QA/CM/Process Support * System/Architecture * Product Def Team * Program Management * Third-Party / Partner * Other | Details | Y |
| Parts Affected | Name of the module within the product The list of modules are entered into BIN settings by a CCB member.  Examples: WDOG, ADC | Details | Y |
| Configs Affected | Name of Specific Product this Problem is reported against. Use official product name or ALL/GENERAL. The list of products are entered into BIN settings by a CCB member. | Details | Y |
| External Issue ID | ID of the External Problem Source (Siebel) | Details | N |
| Origin Baseline | **During CR Analysis state, the Analysis Champ shall update the  “Origin Baseline” field of the Defect CR to reflect the baseline where the defect was introduced into the codebase for the first time.**  Example:  CR ENGR000xx is a Defect CR.   * New State: The CR was raised having as Origin Baseline BLN\_MCAL\_3.0\_BOLERO\_RTM\_3.0.2 (the baseline for which the defect was found/ reported). * Analysis State: During the analysis phase, the Analysis Champ is analyzing the code and discovers that this defect  was introduced into the code base in BLN\_MCAL\_3.0\_PICTUS\_RTM\_2.9.0. release. The Analysis Champ will update the Origin Baseline field and he will select BLN\_MCAL\_3.0\_PICTUS\_RTM\_2.9.0.   **Also, in the Analysis state, the Configs Affected and the Severity of a Defect CR shall be reviewed and updated by the CR analyzer!**  Note:  The Origin Baseline is mainly used to generate the report of known defects for **older releases** and **frozen branches**. It is only way how to distinguish which releases are affected by the defect. If the origin baseline is fulfilled wrongly, our customers might not be inform about the serious defects or we confuse the customers by the tickets which do not affect their source code. | Details | Y |
| Phase Found | Specifies when in the project lifecycle that the defect was reported (i.e., the current phase when the defect is submitted) Use **Post-release** for defects in *delivered* work product. Mandatory on submit for all **Defects.**  Allowed values:   * Planning * Requirements * Architecture * Design * Implementation/Coding * Component Testing * System Integration * System Testing * Post-Release * External | Details | Y |
| Reproducibility | Please specify several steps and conditions when a problem occurs. A problem occurrence should be repeatable. There are four options here:   * Always (every time or almost every time) * Sometimes (not every time, but often) * Rarely (very difficult to reproduce) * Unknown (reproducibility not tested) | Details | Y |

**Rules to be used for selecting properly the CR Severity:**

Severity – impact on SW functionality and customers’ project/application - Freescale Severity set after impact analysis;

Decision flow is the following:

It is functional issue?

If No –> Severity 4 or 5

If Yes 🡪 Severity 3, 2, 1

Is a central or essential function is unusable & there is no reasonable workaround & showstopper for customers -> **Severity 1**

Is a central or essential function significantly degraded & no or incomplete/inadequate workaround & possible showstopper for many customers -> **Severity 2**

Note: For New works and Integration, the severity field is N/A.

Table 16 Severity Types

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Definition** | **Examples** |
| **S1**- Catastrophic (product unfit for use) | Typical Characteristics:  - Central or essential function is unusable  -Wholly unacceptable to target customers/users  -No reasonable workaround for customers/users  Other Potential Impact:  -Probable showstopper or immediate stop ship  **Note: Hot-Fix most likely will be prepared** | *Unable to use AUTOSAR memory stack to read/write into EEPROM.* OR  *Driver cannot be compiled;*  OR  *No messages are transmitted in any configuration (by communication drivers).* |
| **S2** - Serious (essential function degraded) | Typical Characteristics:  -Central or essential function is significantly impaired OR performance/reliability is significantly degraded  -Most customers/users will be dissatisfied; many will complain  -Workarounds, if any, are incomplete or inadequate  Other Potential Impact:  - Possible showstopper, but not necessarily | *When entering critical code, if interrupts are disabled, the pre-interrupts states were not saved and re-enabled on exit.*  OR  *The CAN controller is not set to sleep when sleepmode is requested.*  Explanation: Essential function is significantly impaired; there is  no workaround; customers request a HotFix. |
| **S3**-Moderate (apparent functional problem) | Typical Characteristics:  -A problem that is apparent to the customer/user (likely to be encountered and noticed) that impedes a desired function  -Does not significantly degrade a central or essential function  -Most customers/users will be willing to work around or ignore the problem until a fix is available; some may complain  Other Potential Impact:  -A few customers/users may request return/repair of shipped units  -End user perception of quality damaged | *MSGReset Reason stored in arbitrary public location as opposed to fixed public location*  OR  *[IPV\_FLEXCAN] MB are locked when using Canif\_TxConfirmation; After a CAN message is sent the CANIF is notified by the CAN driver state before the hardware state is set.*  Explanation: No essential functionality is degraded; most of customers will not be affected or they accept a workaround. |
| **S4** - Minor (cosmetic or unapparent problem) | Typical Characteristics:  Cosmetic = Does not impede or materially degrade any intended function of the product  Unapparent = Not likely to be noticed or not likely to be encountered by most users  If noticed or encountered, the will probably ignore the problem; complaints are unlikely  Other Potential Impact:  Too many Severity 4 defects are likely to lead to a damaged perception of quality and may diminish customer/user satisfaction  Examples:  Typographical, stylistic errors in customer documentation. | *Automotive SWClockSrcFailureNotification configuration check*  *OR*  *[PWM] Autosar MCAL Leopard v0.9.0 (ASR 3.0) incomprehensible documentation*  Explanation: No functional problem, moreover simple workaround or another solution exist |
| **S5**- Trivial (transparent to customer/user) | Typical Characteristics:  Not visible to the customer/user  Cosmetic = Does not impede or materially degrade any intended function of the product  Other Potential Impact:  Too many Severity 5 defects may have a detrimental impact on the maintainability of the work product  Example: Documentation problems | *Code traceability must be checked against Design IDs*  Explanation: Error in internal documentation / process |

Table 17 Priority Types Field

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| **P1 - Critical** | **Showstopper, critically urgent** |
| **P2- High** | **Precedence over other work** |
| **P3- Medium** | **Standard planning queue** – default one |
| **P4- Low** | **Low priority, non-essential** |

The priority specifies the degree of urgency committed by the development team/PM for the implementation of the CR.

It is required for all CR subtypes. .The initial desired priority is set by the submitter. The CCB/PM can (and should) change the priority to reflect the actual priority according to the planning, if the case.

Once submitted, only the CCB can change the priority.

The CCB-assigned priority for a CR:

Table 18 CR Priorities – more details

|  |  |  |
| --- | --- | --- |
| **Priority** | **...tells external parties:** | **...tells the developers:** |
| 1 | “We will start working at the CR immediately. It will be delivered as a Hot Fix in a few days.” | “Work on this CR has the highest priority. The CR must be resolved in a few days and delivered to customer as a Hot Fix.” |
| 2 | “We are pushing this CR ahead of all other work. It will be de-livered in the next release, but has higher priority than other CRs planned for that release.” | “Work on this CR should be started immediately and should be completed before working on normal priority CRs.” |
| 3 | “We are addressing this CR in our standard release planning. It will be delivered as scheduled.” | “Work on this CR should be executed according to plan.” |
| 4 | “We will address this CR as time and resources allow. It may be dropped from the scheduled release without advance notice.” | “Work on this CR only after all higher priority CRs have been completed.” |

Note: S1 Defects are usually having P1(HF). But S2 might have as well P1 (HF), if customer request them urgently

From New state, the CR can go to CCB, Analysis or No Changes in the case or CR rejection, as follows:

a) Software Development CRs, especially external defects or new-works, submitted by Customer Engineers, Project Managers, or Tech Leads will go from New 🡪 CCB state, where the decision is made if the CR is going to be implemented, postponed, put on-hold or rejected. If it is going to be implemented, an Analysis Champion is assigned. After Analysis completes, The Analysis Report filed is filled-in, then it is sent back to CCB for Analysis Report approval.

b) CR that need further analysis, and do not need CCB Decision for Analysis, OR CCB meeting is not scheduled yet and analysis needs to be performed might be routed from New 🡪 Analysis State, when it is clear that SW developer will do the analysis. Then the CR will be routed to CCB state for Analysis Report approval.

c) From New state, a CR will go to 🡪 No Changes state in the case of CR withdrawal.

d) From New state, a CR might go to 🡪 Open ONLY in the case of INTEGRATION CRs and CLONES in cases when the Analysis was performed by the parent CR.

### Evaluating a Change Request (CCB and ANALYSIS States)

During CCB state, CR Analysis Report is evaluated, the changes impact is assessment by the Tech Lead, or other CCB member. As result of this analysis, CR will go forward to be implemented (CCB status: “Pending”), or will be set on-hold or postponed (CCB Status: “On-Hold”, “Postponed”

Table 19 Analysis and CCB States Fields

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| Analysis Champ | The name of the engineer who performs the defect analysis. For remote center, the CCB will assign the defect to the manager that is responsible to assign it to someone in his team.  The CR is assign for analysis to the Component Owner who is working on the platform in development at that time. | Y |
| Analysis Due Date | The date the engineer estimates he will finish the CR analysis.  CCB shall determine when the analysis has to be complete depending of the charge of each engineer. When an engineer is committed to a date, he has to provide the analysis in time so that the CCB is able to review it and assign it for work to local or remote resources.  If the engineer knows by advance the analysis will be late, he has to escalade to the management and negotiate for reprioritization of the defect analysis.  The CCB is responsible to put the defect as “Assigned” (see status field description) and give analysis instructions discussed during the committee (“Note log” field in ClearQuest). It may require additional information if description is not clear. | Y |
| Actual Analysis Effort | The Analysis Effort must be when exiting the Analysis State. It represents the actual effort that was needed to analyze the defect. | Y |
| Analysis Report | CR Analysis Report template shall be used. See **MCAL** **bins** chapter – CR templates section. | Y |
| Status | CRs can change Status without changing State. The Status field is not used for all states  For CCB state we may have:   * **Pending** * **On-Hold** * **Postponed**   Once the CR is assigned to the CCB State, the initial Status is ‘**Pending**’.  The CCB can change the status to ‘**OnHold’** if something is needed from a third party before work on the CR can proceed.  The CCB can change the status to ‘Postponed’ if there is a reasonable certainty that the CR will be reconsidered for implementation within the next 18 months. Otherwise, simply reject the CR (there is a special disposition in the NoChanges state for CRs that are “postponed indefinitely”).  For Analysis state the options are:   * **Assigned:** initial Status for the analysis state * **In-progress**: indicates that the analysis has started * **On-Hold**: used when waiting on an external party; the CR will return to InProgress when work resumes.   When entering the Analysis State, the Status is “**Assigned**”. When analysis starts, the CR will have the state “**Analysis-In Progress**”  After the first analyzer finishes the analysis, he will route the CR from “**Analysis – Assigned**” to “**CCB-Pending**". The rest of the analyzers will have to review the analysis performed by the first analyzer and update the Analysis Report field. | Y |
| Note Log Tab | Provide analysis instructions specifying what information is needed | Y |
| Release Notes | Options available: **Yes/ No**  The check box is set on Yes to make the CR visible to  Customers, in Release Note, and it is used in CR Queries.  **Important**! The Release Notes field should be updated when exiting the Analysis state.  *Criteria:* Any CR which changes an artifact visible to the client (**any file in release package or the origin of: source, xml, UM or IM**) shall have the Release Notes set on **Y (Yes).**  CRs should have the field set to **N (No)** if above is not true and:   * the CR is a pure internal task/ work ticket (e.g. requirements analysis, add test for profiling, update the UML Design) * the CR is a FBR or HF clone * the CR is for changes in files not included in the release package (e.g. QP documents) |  |

Important!

During Analysis State, the Configs Affected Field will be updated with all the Affected Platforms.

This info if very important for linking the CRs to the Integration CRs.

Project Managers are querying after specific config and link the CRs to the Integration CRs.

Also, for Defect CRs, the Origin Baseline field must be updated to show the baseline in which the defect was introduced for the first time.

Each CR should be analyzed by both/all JDP team and non JDP team for all projects. As a result of this analysis the Config Affected field and CR Analysis Report shall be updated.

*Note:* If the first analyzer concludes that the CR affects only the platform he did the analysis for, he has to add N/A to the rest of the projects listed in the Analysis Report field.

For each team, there is a CR analysis responsible that is supervising the CR Analysis process and is working with Component Owners from his team to make sure that the CRs are Assigned for Analysis and the CRs Analysis is complete.

After Analysis state, the Description filed might be also updated to be synchronized with the analysis result and to include more accurate information.

From Analysis state a CR can be routed to:

* CCB – Send the analysis report back to the CCB for a decision – if needed CCB Analysis Report Approval
* Open – Assign the CR to a current or future baseline – No need for CCB Analysis Report Review
* NoChanges – Reject the CR due to duplication, indirect resolution, cancellation

### Implementing a Change Request (OPEN State)

After finishing the defect analysis, the next state is “Open”.

Table 20 Open State fields

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| Dev Champ | The name of the engineer who will fix the defect, responsible for Implementation, unit testing and peer review. | Y |
| Resolution Due Date | The Due Date for Development Champ to complete implementation, testing and peer review of the CR and move it to Resolved state. | N |
| Target Baseline | Specify the **Target Baseline** in which this defect should be released | Y |
| Estimate Dev Effort | The estimated development effort (Unit Testing effort included). | N |
| Peer Review Requirement | The minimum peer review requirement recommended. Allowed values:  Mini-Walkthrough  Walkthrough  Inspection | Y |
| Designated Reviewer | The name of the person that will review the implementation of the CR | N |
| Linked Peer Reviews | The list of Peer Review records linked to the CR. The Peer Review information displayed in this field, are: Review ID, Date, Review Type, Session Type, Outcome | Y |

When the development is complete, the champion must change the defect state to “Resolved”. After this state the CR will be automatically routed to Closed state.

### Closing the Change Request (RESOLVED State)

There is one way to move a defect into Resolved state. The defect will go from Open state into Resolved state.

The following fields are going to be filled-in, when routing a defect into Resolved state.

Table 21 Resolved State ClearQuest Fields

| Field | Description | Mandatory (Y/N) | | |
| --- | --- | --- | --- | --- |
| Final Disposition  **(Resolution Tab)** | This field is automatically filled in with the final state of the CR (Resolved). This field is editable only for the CRs which are closed with No Changes. | Y | | |
| Resolution Description  **(Resolution Tab)** | Example: defect has been Implemented, Unit tested and Reviewed.  If the final state in **No Changes**, this field will contain an explanation for the rejection reason | Y | | |
| Resolved-in Baseline  **(Resolution Tab)** | The Resolved- in Baseline is now required upon resolving the CR.  Resolved-in Baseline is automatically set to the Target Baseline. This option is enabled through BIN settings. | Y | | |
| Branch Name  **(Integration Tab)** | The ClearCase development branch name | Y | | |
| Change Set **(Integration Tab)** | The ClearCase Config Spec | Y | | |
| Actual Dev Effort  **(Resolution Tab)** | The actual development effort (Unit Testing effort included) | N | | |
| Delta Lines  **(Integration Tab)** | Represents the modified LOC: added+changed+deleted | Y | | |
| Delta Pages  **(Integration Tab)** | Number of updated pages | N | | |
| Phase Originated | * Specifies when in the project lifecycle that the defect was introduced into the work product * Optional on submit of **Defect** but a value must be specified before the **Defect** can be resolved * Field options are the same as for Phase Found. | Y | | |
| Root Cause | Answers the question: “Why was the defect created?”  Not to be confused with: “Why did the defect escape?”  Optional on submit but must be filled in before resolved   * Allowed values: below table | Y | | |
| Test Result  **(Verification Tab)** | The CR resolver | | Y |
| Test Report  **(Verification Tab)** | Represents the list of tests and their result, to see if they passed or failed. This field applies in the case when the defect is being tested by the developer.  CR Test Report template shall be used. See **MCAL** **bins** chapter – CR templates section. | | Y |
| Configs Tested  **(Verification Tab)** | Represents the platforms on which the tests have been run.  This field applies in the case when the defect is being tested by the developer | | Y |
| Release Notes (Y/N)  **(Verification Tab)** | Indicates whether the CR should be visible to customers, displayed in release note or other customer reports.  Options are: Yes or No | | Y |
| Release Note Text  **(Verification Tab)** | A short summary of the changes. Release Note template shall be used. See **MCAL** **bins** chapter – CR templates section. | | Y |

Note: For New works and Integration, the root cause field is N/A

Table 22 Root Cause Types Field Elements for Defects

|  |  |  |
| --- | --- | --- |
| Root Cause | Description | |
| Category 1 – Planning/Oversight |  | |
| Missing ambiguous or incomplete requirements | A fault caused by ambiguous and incomplete requirements | |
| Late or frequently changing requirements | A fault caused by late of frequently changing requirements | |
| Poor communication within the team | A fault caused by the fact that the communication inside the team was poor | |
| Poor communication with other teams | A fault caused by the fact that the communication with other teams was poor | |
| Unclear ownership of responsibility | A fault caused by the fact that there was enough time allocated for development activities (requirements analysis, design, implementation or testing) | |
| Not enough time for engineering/design | A fault caused by the fact that responsibilities were not properly defined and communicated | |
| Category 2 – Engineering/Design |  | |
| Inattention to detail (e.g. typographical error) | A fault caused by not enough attention paid for programming | |
| Incorrect handling of timing or concurrency | | A fault caused by incorrect handling of timing or concurrency |
| Incorrect handling of boundary or edge condition | | A fault caused by incorrect handling of boundary or edge condition |
| Incorrect handling of startup shutdown or suspend | | A fault caused by incorrect handling of startup shutdown or suspend |
| Incorrect handling of bad input or error cases | | A fault caused by incorrect handling of bad input or error cases |
| Incorrect handling of memory or resources | | A fault caused by incorrect handling of memory or resources |
| Failure to optimize for size speed or power | | A fault caused by failure to optimize for size speed or power |
| Failure to consider dependencies or side effects | | A fault caused by missing to consider dependencies or side effects |
| Failure to address all configurations or variants | | A fault caused by not addressing all configurations or variants |
| Failure to adapt imported or inherited source | | A fault caused by failure in adapting the imported or inherited source |
| Failure to maintain backward compatibility | | A fault caused by failure in maintaining backward compatibilities |
| Failure to comply with conventions and standards | | A fault caused by not complying the standards or convention |
| Failure to complete required reviews or testing | | A fault caused by not finalizing peer reviews or not enough testing |
| Failure to implement in clearest or simplest way | | A fault caused by choosing complex and difficult implementation methods. |
| Failure to implement complete requirements | | A fault caused by incomplete implementation of requirements. |
| Missing ambiguous or incomplete requirements | | A fault caused by missing, ambiguous orincomplete requirements |
| Change made without informing/checking | | A fault caused by not checking the changes made or not informing other people about the changes  A fault caused by incorrect merging or integration. |
| Incorrect integration or merge | |
| Inadequate or incorrect end-user documentation | | A fault caused by incorrect user documentation |
| **Category 3: Education/Training** | |  |
| Inadequate training in technical domain | A fault caused by the fact the developer did not have undergone the adequate technical training | |
| Inadequate training in tools or equipment | A fault caused by the fact the developer did not have undergone the adequate training for tools and equipment | |
| Insufficient knowledge of the system or product | A fault caused by the fact the developer did not have enough knowledge of the system or product | |
| **Category 4: Tools/Equipment** |  | |
| Missing tools/equipment for testing/debugging | A fault caused by the fact that required tools or equipment for debugging or testing were missing | |
| Unstable or error-prone tools/equipment | A fault caused by the fact the tools or equipment had errors or they were unstable | |
| **Category 5: System/Architecture** |  | |
| Poorly documented architecture or interfaces | A fault caused by the fact that the available architecture, design or interfaces documentation was poor | |
| Excessively complex architecture or interfaces | A fault caused by the fact that the available architecture, design or interfaces documentation was too complex | |
| Inadequate or constrained architecture | A fault caused by the fact that the available architecture, design or interfaces documentation was not adequate or has certain constraints | |
| **Category 6: External Causes** |  | |
| Defective or poorly documented 3rd-party IP | A fault caused by the fact the 3-rd party IPs were defective or poorly documented | |
| Defective or poorly documented outsourced work | A fault cased by the fact outsourced work was defective or poorly documented | |
| Defective or poorly documented upstream dependency | A fault caused by the fact upstream documentation was defective or poorly documented | |
| Defect in physical implementation or manufacturing | A fault caused by bugs physical implementation or manufacturing | |
| Change or incompatibility in run-time environment | A fault caused by change or incompatibility in run-time environment. | |
| Errata due to FSL NPI silicon issue | A fault caused by an issue in the NPI silicon. | |

Table 23 Root Cause Types Field Elements for New Works

|  |  |
| --- | --- |
| **Root Cause** | **Description/Examples** |
| Planned Requirement | A change request which has been planned, may be new feature or a change |
| New/Changed Requirement | A change request which has not been planned, may be new feature or a change |
| Internal Refactoring | Optimization, enhancement |

Note:

The Root Cause filed for New Work will be filled in at submission,

For defects, this filed will be filled in at CR closure.

### Other Fields (Resolved state)

Implemented Changes verification is performed by peer review and testing:

1) Peer review – to each CR, the process requires to have at least one peer review attached to it

2) Testing – Unit/Unit integration testing results documented in the Quality tab-Test Report (fields description available in Table 15). Each CR which introduces changes into code should have a test case ID attached.

Table 24 Resolved State ClearQuest Fields

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| Reopened After Integration  **(IntegrationTab)** | The Reopened After Integration check box is automatically checked when a Closed is Reopen | N |
| Linked Integration CRs  **(Integration Tab)** | The list of the Integration CRs linked to the CR. The list is not editable.  The number of Linked ICRs must be: number of Configs Affected minus 1. | N |
| Errata Workaround  **(Documentation Tab)** | Workaround description | N |

### Accepting a Change Request

Change Request acceptance is the peer review acceptance.

**Mandatory!! Each Change Request has at least one peer review attached to it. One peer review might be attached to more CRs. The peer review final state is “Accepted” or “No Faults Finds”. The CR will not be able to be routed to “Resolved” state, with the attached peer review in a state different than “Accepted” or “No Faults Found”.**

The moderator of the peer review is the person who is approving the changes and gives the acceptance.

### Closing a Change Request (NO CHANGES State)

The defect will be closed with No Changes, or rejected, in one of the following cases:

Table 25 No Changes State ClearQuest Fields for Defects

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| Final Disposition | The defect will be closed with No Changes, or rejected, in one of the following cases. Allowed values for this field:   * Non-reproducible * Functions as designed * Resolved indirectly\* * Resolved via third-party\*\* * Known issue, Third-party * Known issue, No fix planned * Cancelled/invalid/obsolete * Duplicate of another CR \*\*\* * Withdrawn by submitter | Y |

### 

Table 27 No Changes State ClearQuest Fields for New Works

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| Final Disposition | The defect will be closed with No Changes, or rejected, in one of the following cases. Allowed values for this field:   * Declined by CCB Postponed indefinitely * Resolved indirectly\* * Resolved via third-party\*\* * Cancelled/invalid/obsolete * Duplicate of another CR \*\*\* * Withdrawn by submitter | Y |

Table 27 No Changes State ClearQuest Fields for Inquiry Change Requests

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| Final Disposition | The defect will be closed with No Changes, or rejected, in one of the following cases. Allowed values for this field:   * Completed * Declined by CCB * Postponed indefinitely * Resolved indirectly\* * Resolved via third-party\*\* * Cancelled/invalid/obsolete * Duplicate of another CR \*\*\* * Withdrawn by submitter | Y |

\* “Resolved indirectly” means that the issue or request went away with any direct changes for this CR. E.g., the symptoms stopped occurring (possibly due to other changes); the request was no longer needed due to other changes; etc. In this case the CR is closed with *No Changes*, Final Disposition “*Resolved indirectly”* and Resolution Description *“Resolved indirectly in ENGR00XYZW”.*

When closing a CR with “Resolved indirectly” a Related CR is required to be added in Links tab (add the CR which indirectly resolved the current CR). Relationship should be *“is permanently fixed by”.*

**Important!** When closing a CR with Final Disposition “Resolved Indirectly”, make sure the change was implemented on all affected platforms.

\*\* “Resolved via third-party” means that the CR was resolved through changes in an external component or subcomponents provided separately by a third-party.

\*\*\* If the Final Disposition is “Duplicate of another CR”, the ID of the duplicate CR must be entered in Duplicate Of field.

### Other Fields (not mandatory and not linked to any state)

Table 28 Other Fields/Tabs (not mandatory and not linked to any state)

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| **Documentation Tab** | This tab is used for documentation related information. | N |
| **Note Log Tab** | * Additional information to be added during CR lifecycle | N (Y for Analysis State) |
| **Linked CRs Tab** | * The **Links Tab** contains the list of Related CR and the list of Integrated CRs (used for Integration CRs only) * Specifies other CRs that would be of interest to anyone looking at or working on this CR (for information only!) * The list can be modified at any time by any user * The link is bi-directional (this defect is automatically added to the list of the other defects) | N |
| **Tags Tab** | * List of Bin Tags, Customer Tags and Track tags. MSGuses only Track Tags (see chapter 4.7) | N |
| **Attachments** | * To add file attachments | N |
| **History** | * The **History** tab contains the history of operations on the **CR**. * Click on the history number to view additional details about an action, such as the list of all fields changed. * This tab displays, also a summary of the CR history, like: *Last Modified Date, Closed Date, Closed By, Most Recent Submitter or Most Recent Submit Date.* | N |
| **Notify** | Most email notification rules for the team are set by the CCB. The CC list is only for specific user who wishes to always be informed about changes to this particular defect. | N |

## Inquiry CRs

This is the lifecycle of the Inquiry type Change Requests:

**New**

**Open**

**Analysis**

**Resolved**

**NoChanges**

**CCB**

Figure 19 Inquiry Change Requests Lifecycle

“Open” and “Resolve” states do not exist for this type.

Are used for:

* Recording requests when it is not known if it is a defect or a new-work. When this aspect is decided, the Subtype will be changed.
* Tracking questions, action items, support issues, investigation issues, etc

Do not follow the same workflow as the other CR subtypes

If not converted to one of the other CR subtypes, then the typical workflow for an Inquiry CR is *New* >>> *Analysis* >>> *NoChanges*

Most of the work for Inquiry CRs is done in the Analysis state

## Integration Change Requests

Integration CRs shall be created for:

* Autosar Generic and Specific Changes propagation (even if for Autosar generic case there in nothing to propagate from Configuration Management standpoint);
* Autosar Generic and Specific Changes for integrating development of more CRs affecting the same file/configuration Item.

Integration CRs shall not be created for:

* IP Vault CRs, because IP Vault drivers are independent components for each the latest version is integrated in each release or for stubs (DEM, DET, CANIF …).

Note: ICRs are created for IPVs or stubs only for Frozen Branch development.

**Integration CRs:**

* Follow same workflow as Defect and NewWork CRs
* Have an assigned Dev Champ, have a Target Baseline, must be integrated in time for that baseline, etc.
* Are designed to track the integration of a set of CRs, or changes propagation

Integration CRs workflow is presented in the following diagram:

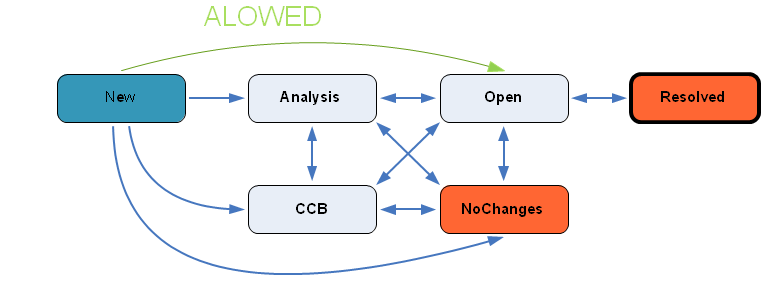


Figure 20 Integration CRs workflow

How and when are we using Integration CRs?

When a new CR affecting platforms A. B, C, D was reported on platform A.

How to proceed?

* + Resolve the CR in the next release (A, B, C or D, BLN or HF) as planned
  + Link the CR to the Integration CRs on other affected platforms (Integration CRs have been already created) when the CR is routed to Open state

**Notes:**

Integration CR will be created by the component owner who is performing the analysis, right after analysis for all affected platforms is finalized.

Who?

* Project Managers to be responsible for creation and maintenance of the Integration CRs - Proposal

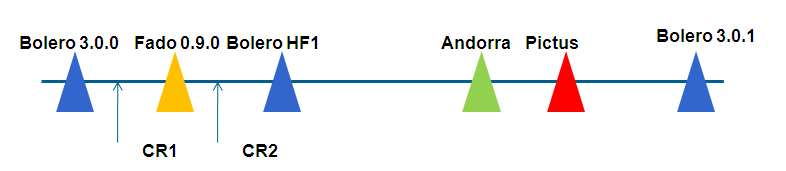
****

Figure 21 Integration CRs example

CR1: Raised on Bolero affecting ADC on all platforms

* + will be implemented on Fado
  + will be linked to the ADC Integration CRs for all other projects (Andorra, ….)

Example:

* A CR ENGR1234 is submitted for Bolero Platform
* The CR ENGR1234 affects Leopard Platform as well
* ENGR1234 is planned to be developed for Leopard Platform which is the first planned release

*Solution/Steps:*

* The CR ENGR1234 will have Configs Affected both Bolero & Leopard
* The CR ENGR1234 will be used for Leopard release
* An integration CR ENGR5678, was already created for Bolero. ENGR1234 CR is linked to ENGR5678.

There are two specific cases for Integration CRs Usage:

1. Assume that a CR raised for Platform A (ENGR1234), impacts also Platform B 🡪 Create an integration CR: ENGR5678 for platform B

ENGR5678 shall be created/updated (if existing) right after ENGR1234 is analyzed, not later !

In Clear Case, the change from dev\_engr1234\_ngys001\_gptfix will be integrated to the dev branch of ICR ENGR5678. See fig. 22.

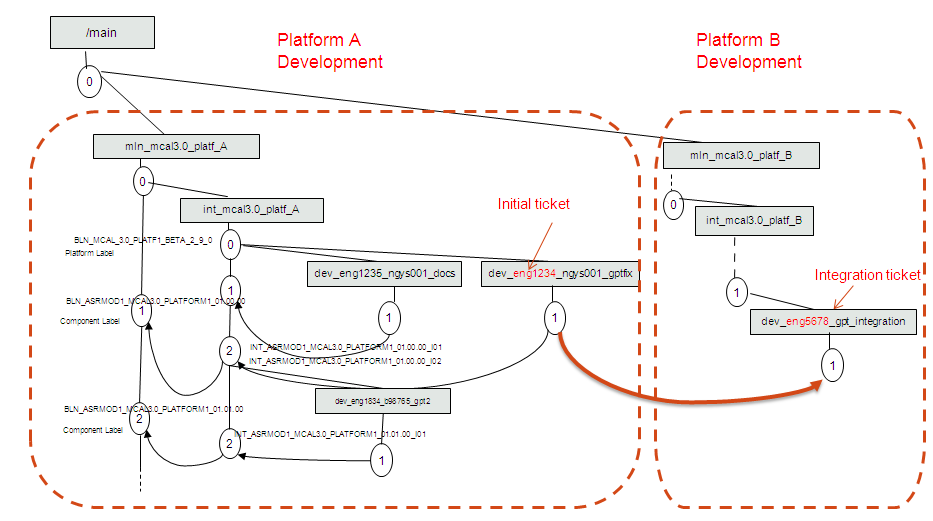


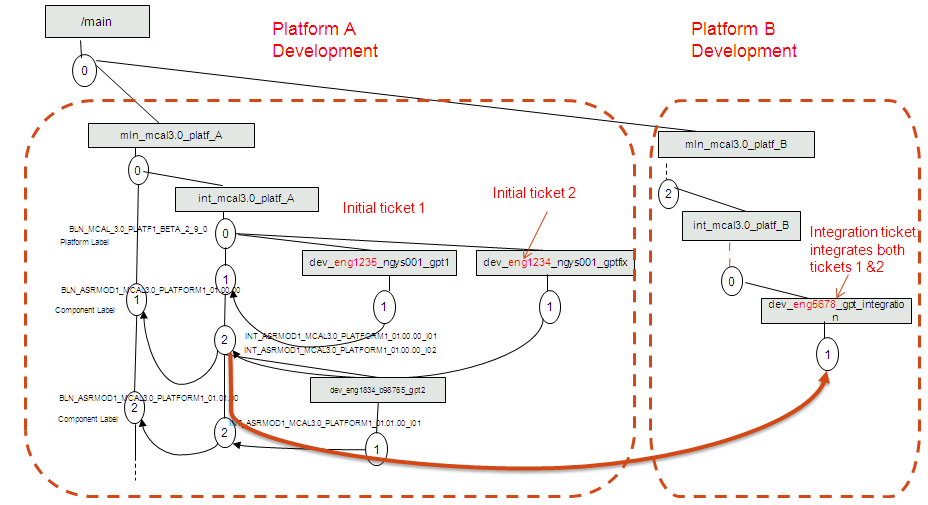
Figure 22 Integration CR Usage 1

1. Assume that 2 CRs are raised for Platform A (ENGR1235 and ENGR1234), and impact Platform B (Integration CR ENGR5678).

In Clear Case, the integration will be done from the integration branch of platform A to dev branch of ICR5678 (ICR dev branch created for Platform B).

See fig. 23: version 2 from int branch, which contains both CRs, is merged to dev branch of ICR ENGR5678.

**Note:** For “bulk” integration, is it recommended to integrate the changes from latest version of int or mln branch.

Figure 23 Integration CR Usage 2

Important:

* During CR Analysis, the Configs Affected Field will be updated with all the Affected Platforms. This info if very important for linking the CRs to the Integration CRs.
* The CR Integration Analysis query is checking and reporting periodically the Clear Quest ICR violations. The query is checking each CR to see if the number of Configs Affected is the number of linked ICR minus 1.This report lists all Defect/NewWork CRs which do not have correct link to immediate Baseline (Target Baseline) or Integration CR. Only Closed CRs, which affect Autosar modules, are evaluated.

## Forwarding and Cloning a Change Request

1. Why forward a CR?

* Submitted to the wrong bin
* The CR is a clone for another bin (“clone-and-forward”)

Who is allowed to forward?

* Everyone if the CR is still in the New state
* In any other state, forward may be restricted by CCB
* CRs in the Resolved state must be reopened first
* Forwarded CRs enter the destination bin in the New state
* To the destination bin, the CR is just like a newly submitted CR
* Fields unique to each bin are cleared and must be filled-in again (e.g., Affected Parts, Affected Configs, Origin Baseline)

1. Why/When Cloning a CR?

Clones are treated like new CRs. They are created in the New state. They are created in the same bin as the original CR.

There are some cases when we clone CRs:

1. CRs affecting Multiple Modules (Autosar or IP Modules)
2. External New Feature when TWG/marketing review/approval is needed (Freescale Only);
3. CRs used for Hot Fix preparation and FBR Preparation
4. CRs affecting more than one Code Base - use “clone and forward” option – clone is done in the the BIN where the CR was originally submitted and then FW in the BIN where need also to be implemented.

Case 1 – CRs affecting Multiple Modules (also called “umbrella” CRs)

This case is applicable when a CR (Defect/New Work/ Inquiry) must be analyzed and implemented for all Autosar modules. In this case the submitter creates an “umbrella” CR having ALL/GENERAL as Affected Parts and the tag [ALL GENEREL] in the headline.

The CCB will clone the umbrella CR for all Autosar modules and the analysis of the CR will be done, by the analyzer, in the clones.

The analysis champ (ASR module owner) will analyze the clone CR, will fill out the analysis report and will clone the CR for all IPVs impacted by this change.

Because the change and configuration process is different between ASR and IP\_Vault modules, a change request which is affecting an ASR module file and an IPV module file, shall have a CR for ASR module and a clone for IPV module.

Note! If an “umbrella” CR has to be analysis before cloning for all modules, the submitter should state this request in the Description filed of the umbrella CR (e.g.: *“To be analyzed by a technical leader before cloning to all ASR modules!”*)

After all clones are resolved, the umbrella CR will be closed with NoChanges.

Case 2 –New Features

New feature CRs received from customers are first entered in the TWG bin and analyzed by the TWG group.

After the new feature is approved by the TWG group, the requirement manager will clone the CR for all product bins (MCAL, MCAL4.0..) and will add the new feature to cPRD.

Important! All new feature requests, received from internal or external sources, shall be first submitted to TWG bin for approval.

Case 3 – CRs used for HF preparation

A not resolved CR included in a Hot Fix request will be cloned and the clone CR will be used for Hot Fix preparation.

Hot Fix clone CRs should have the following characteristics:

- [HF] tag should be included in CR headline; *Eg: [HF] [AUTOMOTIVE SW] Inconsistent usage of parameter Automotive SWMode (HF)*

- HF clones will Priority 1

- Will have only one config affected: the platform for which the Hof Fix is requested;

- Release Notes Y/N filed will be No

After the implementation of the hot fix is finalize, the clone closed with NoCanges, Final Disposition “Obsolete” and Resolution Description: “HF CRs become obsolete after the HF delivery”.

Case 4 – CRs affecting both MCAL and MCAL4.0 products

ClearQuest has no equivalent to “clone-and-forward” To clone a CR to another bin: Clone first. Then forward. It must be done as two separate operations (CQ limitation).

This procedure (“clone-and-forward”) will be applied in case a CR is submitted to a BIN (example: MCAL) and it affects also a component baseline from another BIN (example: MCAL 4.0).

First step is to clone the CR. Clones are treated like a new CR, they are created in the New state

in the same bin as the original CR.

Fields copied from the original CR: Headline, Description, Severity, Priority, all fields entered on the ‘Details’ and ‘Analysis’ tabs, Release Notes, Related CRs, Integrated CRs, all tags on the ‘Tags’ tab, and the E-mail CC List

Fields NOT copied to the clone: Development planning fields: Dev Champ, Target Baseline, Peer Review Requirement, Designated Reviewer, Estimated Effort, Attachments

*After cloning the CR, the next step is to forward the new CR (clone) to the target BIN.*

Forwarded CRs enter the destination bin in the New state and mandatory fields must be filled-in again.

Mandatory fields, not copied from the initial CR: Parent BIN, Parts Affected, Configs Affected,Origin/ Fouand-in Baseline.

After this state the CR will follow its standard workflow.

## Specific Track Tags Usage

A tag is a mechanism to cross-reference a set of CRs that are scattered across many bins. Queries can return all CRs that have a particular tag attached to them.

Track tags are ClearQuest records with no workflow, used as labels to be applied with the purpose of grouping different CRs.

We are using track tags to group CRs, for three purposes:

1. Mark and group the CRs included in a Hot Fix;
2. Mark and group CRs used for quality package preparation. This case is applicable only for those situations when the quality package is delivered after the release package was delivered.
3. Private Track Tags used to mark CRs needed for different personal purposes (group CRs required by a customer for a Frozen Branch, mark CRs that need to be analyze the non JDP team, etc.).

All CRs which are planned to be included in a hot-fix release or used for Quality reports corrections, will have a track-tag applied, having a suggestive name

Steps in Track Tag creation:

* New Track Tag
* Name and Description of the tag:

The tag name is the name of the HF Label or QP label.

* Add the name of the person who controls the tag;
* Attach the tag already created to the CR (“Tags” menu).

1. Naming rules for Quality Package Track Tag:

QP\_MCAL\_<ASR\_VERSION>\_<PLATFORM>\_<Release\_Type>\_<Release\_Version>\_QP<QP\_Version>

Where:

Release\_Type is one of the following three types:

EAR – Early Access Release

Beta

RTM – Ready to Market

Release Version – is of the following type: xx.yy.zz

QP Version – is of the following type: x.y

*Example:* QP\_MCAL\_3.0\_ANDORRA\_RTM\_2.0.0\_QP0.1

1. Hot Fix Track Tag – refer to MCAL HF Procedure [11] for more details.

HF\_MCAL\_<ASR\_VERSION>\_<PLATFORM>\_<Release\_Type>\_<HFx><Release\_Version>

Where:

Release\_Type is one of the following three types:

EAR – Early Access Release

Beta

RTM – Ready to Market

Release Version – is of the following type: xx.yy.zz

HF Version – is of the following type: x

*Example:* HF\_MCAL\_3.0\_PICTUS\_RTM\_HF1\_3.0.0

1. Private Track Tags

PVT\_<Core ID>\_Short Text

Where:

<Core ID> - is the core ID of the Track Tag owner

<Short Text> - is a short description of the Track Tag

The purpose of the private track tag should be described in the Track Tag record (Main Tab).

*Example*: PVT\_B36114\_FBR01\_CRs

## Change management rules for MCAL 3.2 projects

MCAL 3.2 records are managed in MCAL BIN.

MCAL 3.2 platforms are treated in the same way as 3.0 platforms and will have their own Config Affected in the MCAL BIN (e.g: MPC564XB\_3.2, MPC56XXB\_3.2).

For each Change Request which affects multiple configurations including an Autosar 3.2 configuration, an integration CR should be created to integrate the changes on all affected configuration (except for IPVs and Stubs). For more details regarding Integration CRs, see chapter 3.5.

Baselines used for MCAL 3.2 projects are following the standard workflow and are managed as follows:

* Platform baselines will be Autosar specific and the baseline name will contain the 3.2 Autosar version. E.g.: BLN\_MCAL\_3.2\_BOLERO\_RTM\_1.0.0
* Autosar component baselines will be Autosar specific and the baseline name will contain the 3.2 Autosar version. E.g.: BLN\_ADC\_MCAL\_3.2\_XPC560XB\_01.00.00,
* IPV and stub baselines will be common between MCAL 3.0 and MCAL 3.2 releases. MCAL 3.2 specific baselines will not be used for IPV and stubs! E.g.: BLN\_IPV\_FLEXCAN\_ MCAL\_3.0\_01.01.0

Naming rules used for MCAL 3.2 are described in chapter 3.9.

## Errata Handling – Clear Quest

When a new HW errata is available for an MCAL platform, a New Work CR will be created for review and analysis of the new errata.

Depending on the result of this review, the next step is:

1. If a software workaround need to be implemented, the New Work CR is closed with Resolved and a new Defect CR is raised;
2. If the errata does not impact the software, the New Work ticket will be closed with No Changes.

**Note:** For both cases the result of the errata review shall be recorded in the analysis report - proposed solution section.

The change request used for implementation of HW errata will be recorded in Clear Quest as Defect CR, containing the following information:

* **Headline:** shall contain the Errata ID (e.g: [ADC] New errata e4186 and e4146 implementation for ADC)
* **Reported By (role):** Design/Dev (HW)
* **Root Cause:** “Defect in physical implementation or manufacturing”
* **Linked CRs tab**: Add the New Work CR used for review in the Related CRs filed

## Baselines

### Baseline Lifecycle

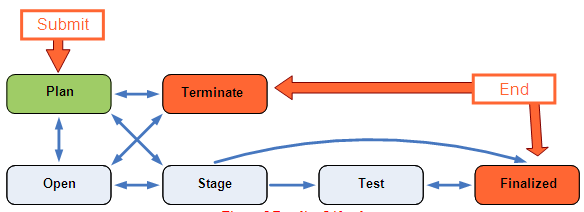


Figure 24 Baseline Lifecycle

### Baselines Introduction

Baselines are the way of collecting more Defects or New Works according to a previous established pattern. Examples: Group all the defects/new works targeting the same component, into a baseline. All the defect IDs will be linked to the baseline.

Releases containing changes will generate a new baseline. A baseline is identified by a unique ID in ClearQuest. The CCB Member or the Integration Engineer are responsible to manage baseline for project releases.

Every CR is assigned to exactly one Target Baseline.

That Target Baseline must be in the *same bin* as the CR

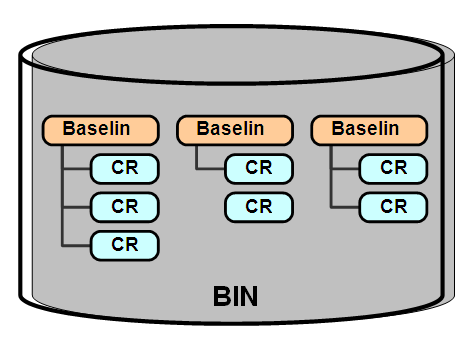


Figure 25 ClearQuest Baselines

**Platform baselines** then “*consume*” **component baselines.**

**Example:**

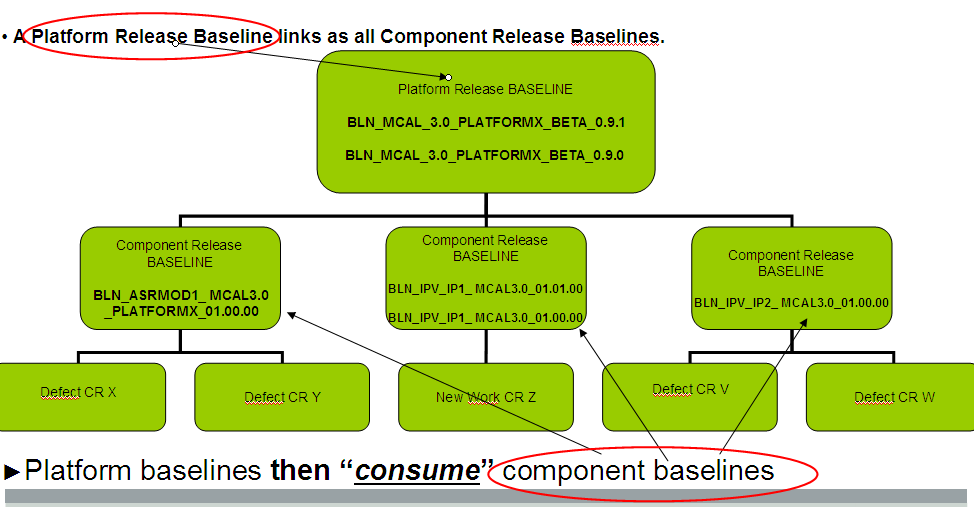


Figure 26 ClearQuest Platform Baselines

Each bin represents a component or project

* + The baselines should form *one* “series”
  + Forks in the series are permitted (for patches, point releases), but parallel/mixed series of baselines in one bin should be avoided.

Some component teams manage multiple bins

* + The team might have several independent components (i.e., the components are built and released separately)
  + The team might have a single component with several independent mainlines (each with its own series of baselines)
  + Shared changes must be managed using propagation (i.e., using cloned CRs or Integration CRs)

### Planning a Baseline (“PLAN”)

When a Baseline is created, it is in the “Plan” state. Plan state means it is a *future* baseline

A baseline can include other components baselines.

In Plan state, the components owners will attach their development CRs to a Baseline. At this stage CRs cannot be integrated. Check box is disabled in ClearQuest.

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| **State** | The State of the baseline according to Baseline workflow.  This field is automatically filled out. | Read-only |
| **Parent BIN** | The BIN where the baseline belongs to. | Y |
| **Baseline Name** | The name of the baseline (usually coincides with the ClearCase label name)   * Each team decides its own naming convention * Baseline names must be **globally unique** in the CQ database * Good practice is to put the bin name in the baseline names * Try to follow the naming convention used in the vobs * E.g.: BLN\_MCAL\_4.0\_PICTUS\_BETA\_0.9.0 or BLN\_ADC\_MCAL\_4.0\_XPC560XP\_01.00.00   Baseline names may contain:   * Letters (uppercase only) and numbers * Underscore (\_), hyphen (-), or dot (.) * *No embedded spaces or lowercase letters* * Baseline names can be changed by the CCB at any time * **For Baseline naming convention see chapter 4.14 of the current procedure** * The Baseline’s CQ ID always stays the same * The Baseline name is a unique value | Y |
| **Baseline Name (short)** | A short name of the Baseline. This field of a Baseline record is not used for MCAL projects and can be left blank. | N |
| **Planned Stage Date** | The date when the baseline is expected to be frozen. “**Stage**” means to lock the baseline and put the release under revision control. | N |
| **Derivation and Previous Baseline** | The **Derivation** and **Previous Baseline** fields are used to record the “*lineage*” of the baseline.  **Derivation** field choices:   * “Foundation Baseline” * “Next in Existing Series” * “First in Derived Series”   Note: Set to ‘**Foundation Baseline**’ and ‘**N/A**’ for the first baseline in the bin.  More about baseline lineage:   * Every bin should have one at most two **Foundation Baselines** * Plan ahead as many baselines as you want | Y |
| **Linked CRs Tab** | Contains the list of CRs linked to the baseline.  The Linked CRs field is not editable. | Read Only |
| **Linked Baselines Tab** | Contains the list of Baselines linked to the current baselines.   * **Consumed Baselines**: list of the baselines which are consumed by the current baseline. Usually **component baselines** are displayed here. * **Linked Consumer Baselines**: List of the Baselines which will consume the current baseline. Usually **release baselines** are displayed here. This field is not editable. | N |

Table 29 Plan State Fields (not mandatory and not linked to any state)

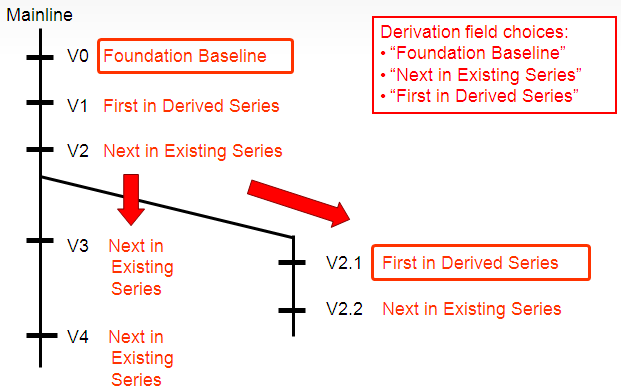
**Important !!!**

**Baselines Derivation Rules are following Clearcase derivation of the mainline branches.**

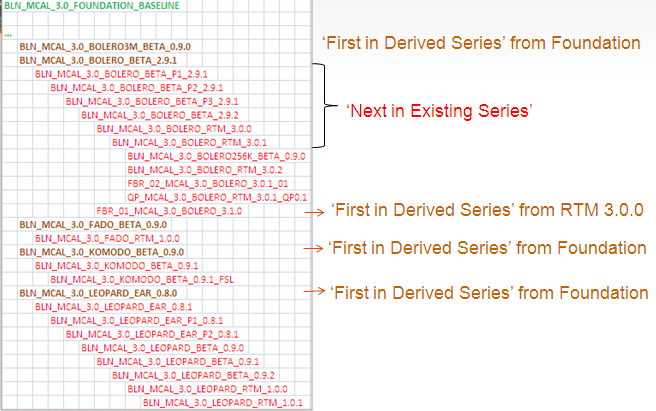
The first Component Baselines for Autosar and IP modules, firstly created in the VOB shall be derived as “First in Derived Series” from CodeBase Foundation Baseline (ex: BLN\_MCAL\_4.0\_CODEBASE\_00.00.00)

For the next platforms, the Component Baselines for Autosar modules (platform dependent), shall be derived as “First in Derived Series” from the previous baselines created during previous releases.

For the next platforms, the Component Baselines for IP Modules (platform independent) shall be derived as “Next in Existing Series” from latest created baselines.



Examples of Release Baselines Derivation:



Examples of Component Baselines Derivation:

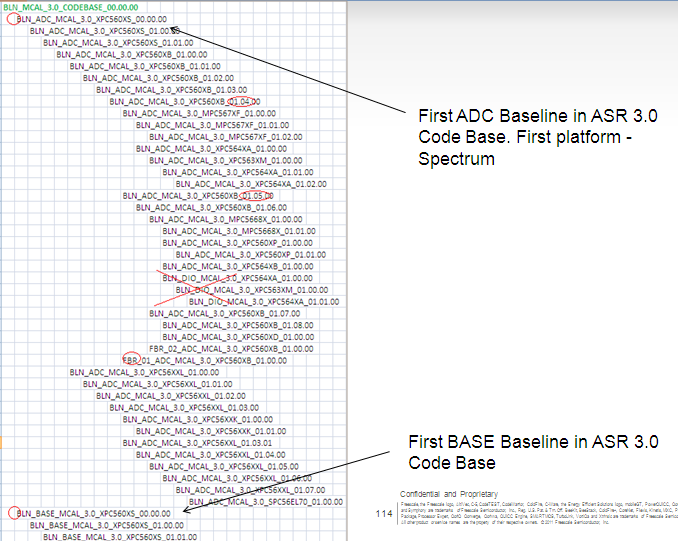


Figure 27 Baselines Derivation

Other examples:

Rule 1: First New Platform: Component Baseline Derivation

Current Baseline: BLN\_FEE\_MCAL\_3.0\_XPC563XM\_01.00.00,

Previous Baseline: BLN\_FEE\_MCAL\_3.0\_XPC564XA\_01.00.00

First Monaco release

Current Baseline: BLN\_FEE\_MCAL\_3.0\_XPC564XA\_01.00.00,

Previous Baseline : BLN\_FEE\_MCAL\_3.0\_MPC5668X\_01.00.00

First Andorra release

Current Baseline: BLN\_FEE\_MCAL\_3.0\_MPC5668X\_01.00.00,

Previous Baseline: BLN\_FEE\_MCAL\_3.0\_XPC560XB\_01.02.00

First Fado release.

Rule 2: Same Platform: Comp Baseline Derivation

Current Baseline: BLN\_FEE\_MCAL\_3.0\_XPC560XB\_01.02.00,

Previous Baseline:BLN\_FEE\_MCAL\_3.0\_XPC560XB\_01.01.00

Third Bolero release.

### Opening a Baseline (“OPEN”)

-Open state means it is the *current* baseline

-Typically only one baseline is in the Open state at a time

Work performed:

-Continued planning (add/remove CRs)

-Development, peer review, unit test of all CRs

-Integration and integration test of Resolved CRs

At this stage defects can be marked as Integrated to enable early integration and testing with daily build. In this case daily build will be generated from a component integration label.

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| **Target Branch** | The **branch name** where the changes will be integrated | Y |
| **Development Config Spec** | The development config spec that will be used for the baseline (ex CS based on development branch) | Y |
| **Integration Config Spec** | The integration config specs that will be used for the baseline (ex: integration label)  Any changes to the config specs after the baseline is opened causes email to be sent to full team | Y |

Table 30Plan State Fields

### Staging a Baseline (“STAGE”)

A Baseline is moved to the “Stage” state in preparation for a release. Responsible: Staging Champion

Work performed in Stage state:

* Integrate any CRs that are not yet integrated
* Perform the official build and final sanity check
* Stage the release artifacts and lock the baseline

To enter Stage state, all child CRs must already be resolved

* Coincides with Release Readiness or Test Readiness Review
* At this time, the Baseline is frozen (e.g: labels are locked under Clearcase).
* All defects shall be in “Resolved“ state. A defect can not be added into the baseline or removed from the baseline.
* Defects merge will be finalized and integrated into the Baseline.
* Daily build is performed from components released and critical sanity tests are performed against the build.

While in the Stage state:

* CRs can no longer be added or removed from the baseline
* The resolved CRs in the baseline can no longer be reopened

Note: The Staging Champ can use the ‘IntegrateAll’ utility to update the Integrated flag of all CRs at once

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| **Staging Champ** | The person responsible to stage the release (usually a build engineer)  Note: If you try to change a Baseline to the Stage state and there are CRs in the Baseline that are not in the Resolved state, you will get the error shown above. You will need to return to the Baseline record by clicking the Return button, move the CRs to the Resolved state, then modify the Baseline and move to the Stage state. | Y |
| **Development Config Spec** | The development config spec that will be used for the baseline (ex CS based on development branch) | N |
| **Integration Config Spec** | The integration config specs that will be used for the baseline (ex: integration label)  Any changes to the config specs after the baseline is opened causes email to be sent to full team | N |

Table 31 Stage State Fields

WARNING:

The baseline’s contents *cannot* be changed after it has been moved to the Test or Finalized state. Escaped defects and any new changes must be resolved in a future baseline. Be sure that everything is in the Baseline before it is moved to the Stage, Test, or Finalized state.

### Testing a Baseline (“TEST”)

Note! This state will not be used for MCAL Development. It is used only in the case of independent testing team, or in the case of system testing.

### Closing a Baseline (“FINALIZE”)

From the Test state, the baseline is moved to the Finalized state. This is a released state. The CCB defines the criteria for moving the baseline to the Finalized state.

If problems are discovered in the Finalized state, the CCB can move the baseline back to the Test state for further testing.

If issues found are not critical they will be resolved in next baseline

Finalized typically means the baseline has been deployed or released, unless the QA Designation is ‘Rejected/Failed’ or ‘Internal Use Only’

The QA Designation is updated to reflect the final status after testing:

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| **QA Designation** | Update the **QA Designation** to reflect the final determination after testing.  The level of quality achieved by the baseline (so far). | Y |

Table 32 Finalized State Fields

**QA Designation options are:**

| Field | Description |
| --- | --- |
| **Rejected/Failed** | * The Baseline is known bad * Not suitable for use by anybody |
| **Internal Use Only** | * For use by the development team only * The quality has not been declared |
| **Alpha Engineering** | * May not be complete to the requirements * Little/no development testing performed * Documentation may not be current |
| **Beta Engineering** | * Functionally complete to requirements * Development testing largely completed |
| **Production Candidate** | * Submitted for validation / acceptance * Code coverage 100%, fully testes * Documentation fully updated |
| **General Deployment** | * Passes validation / acceptance * Approved for final production |

Table 33 QA Designation Field Option

### Rejecting a Baseline (“TERMINATE”)

If a Baseline is created and then abandoned, the Baseline is moved to the “Terminated” state. All defects associated with the Baseline must be moved to a different Baseline or closed as NoChanges.

The baseline can be restored later if needed.

Terminated state means the baseline has been *abandoned*

Staged baselines, even if found to be bad go to Finalized

All CRs in the Terminated baseline be moved to a different Target Baseline or closed as NoChanges

Terminated baselines can be reopened, if necessary.

### Other Baseline related Fields/Tabs

| Field | Description | Mandatory (Y/N) |
| --- | --- | --- |
| **Metrics/ Reviews Tab** | * Contains metrics information like: *Lines Of Code Total, Lines Of Code Added, Lines Of Code Changed, Lines Of Code Deleted*. For MCAL and MCAL 4.0 baselines this kind of information are not required. * *Linked Peer Reviews*: Shows the list of Peer Review IDs, reviews performed for release artifacts. This section is Not Editable. | N |
| **Track Tags Tab** | * This Tab contains the list of Track Tags applied on this baseline. | N |
| **Attachments Tab** | * To add file attachments | N |
| **Note Log Tab** | * Additional information to be added during Baselines lifecycle | N |
| **History Tab** | * The **History** tab contains the history of operations on the **Baseline**. * Click on the history number to view additional details about an action, such as the list of all fields changed. | N |
| **Notify Tab** | * *Current Assignee*: Current assignee will always receive notifications. If this field is blank, this means it is assigned to Parent Bin's CCB. * *Email CC List*: The CC list specific user who wishes to always be informed about changes to the Baseline. | N |

Table 34 Other Fields (not mandatory and not linked to any state)

### MCAL Baselines Naming Convention

There are two types of ClearQuest Baselines:

a) Component baselines associated to Component Clearcase labels. For each component will be created one baseline. One baseline may incorporate one or more change requests

b) Release baselines associated to Release Clearcase labels. For each release will be created one release baseline. The release baseline is a parent baseline incorporating all component baselines which will be released.

a) Component Baselines

According to this document- CM Chapters- for each component release label there will be one component baseline.

The component ClearQuest Baseline will be Autosar and Platform dependent.

Note !: The name of the ClearQuest Target Component Baseline is the same with the name of the Clearcase Component Label.

Naming convention:

For IPVault Drivers

BLN\_<IP\_NAME>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<Baseline\_version>

Where:

<IP\_NAME> - the name of the IP: example: EMIOS

<PRODUCT\_NAME> - MCAL or SMCAL

<ASR\_VERSION> - the Autosar Version: Example: 3.0 or 4.0

<Baseline\_version> - the version of the baseline/release of the following format:

xx.yy.zz where xx – major, yy-minor zz-patch (service release)

ex: 00.08.01

Note: Low-level drivers are NOT platform dependent, but Autosar dependent

**Note:**

1. For new platforms, the numbering of the IPV baselines version will continue from last IPV baseline. The Derivation of the baseline will be “Next in Existing Series”.
2. For MCAL 3.2 projects, the IPV baselines will follow the above rule (note 1) and the baseline name will follow the MCAL 3.0 naming convention.

Ex:

BLN\_IPV\_FLEXCAN\_ MCAL\_3.0\_01.00.00

BLN\_IPV\_FLEXCAN\_ MCAL\_4.0\_01.00.00

BLN\_IPV\_FLEXCAN\_ SMCAL\_4.0\_01.00.00

For Autosar Drivers

BLN\_<ASR\_MODULE>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Baseline\_version>

Note: Autosar Driver are platform dependent

Where:

<ASR\_MODULE> - Autosar Module

<PRODUCT\_NAME> - MCAL or SMCAL

<PLATFORM> - Platform Name, for current project: MPC56XXL

<ASR\_VERSION> - the Autosar Version: Example: 3.0, 3.2 or 4.0

<Baseline\_version> - Baseline/Release Version of the type xx.yy.zz

**Note:** For new platforms, the numbering of the Autosar driver baseline version will continue will start from 01.00.00. The Derivation of the baseline will be “First in Derived Series”.

**Ex:**

BLN\_ADC \_MCAL\_3.0\_XPC56XXL\_01.00.00

BLN\_ADC\_MCAL\_3.2\_XPC560XB\_01.00.00

BLN\_ADC \_MCAL\_4.0\_XPC56XXL\_01.00.00

BLN\_ADC \_SMCAL\_4.0\_XPC56XXL\_01.00.00

For Stub Modules

BLN\_<STUB\_NAME>\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<Baseline\_version>

Where:

<STUB\_NAME> - the name of the Stub: example: DEM

<PRODUCT\_NAME> - MCAL or SMCAL

<ASR\_VERSION> - the Autosar Version: Example: 3.0 or 4.0

<Baseline\_version> - the version of the baseline/release of the following format:

xx.yy.zz where xx – major, yy-minor zz-patch (service release)

ex: 00.08.01

Note:

1. For a new platform, the numbering of the Stub baseline version will continue from last IPV baseline. The Derivation of the baseline will be “Next in Existing Series”.
2. For MCAL 3.2 projects, the IPV baselines will follow the above rule (note 1) and the baseline name will follow the MCAL 3.0 naming convention.

Stub Modules are NOT platform dependent, but Autosar dependent

**This rule is applied only for Stub Modules which are NOT platform dependent, but Autosar dependent. For platform dependent Stubs, case b) naming convention will be used.**

Ex:

BLN\_DEM\_ MCAL\_4.0\_01.00.00

BLN\_DEM\_ MCAL\_4.0\_01.00.00

b) Release Baselines:

Naming convention for Release Baselines (Parent Baselines):

BLN\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Release\_Type>\_<Release\_Version>

BLN\_<PRODUCT\_NAME>\_<ASR\_VERSION>\_<PLATFORM>\_<Release\_Type>\_<Px><Release\_Version> - in case of Patch Releases

Where:

<ASR\_VERSION> - the Autosar Version: Example: 3.0, 3.2 or 4.0

<PRODUCT\_NAME> - MCAL or SMCAL

Release\_Type is one of the following three types:

* EAR – Early Access Release
* Beta
* RTM – Ready to Market

Pi –Patch Version

Release Version – is of the following type: x.y.z (0.8.0/0.8.1 for the first/second EAR, 0.9.0/0.9.1 for the first/second Beta, 1.0.0 for the first RTM )

Note: For Hot Fixes, no dedicated baseline will be created

Note: There is a dummy baseline created in MCAL BIN. When is not known which is the baseline the Change Request belongs to, it will be linked temporarily to this dummy baseline. The name of the dummy baseline is : BLN\_MCAL30\_XYZ.

Examples:

* 1. For normal releases:

BLN\_MCAL\_3.0\_LEOPARD\_BETA\_0.9.1

BLN\_MCAL\_3.2\_BOLERO\_BETA\_0.9.0

BLN\_MCAL\_4.0\_LEOPARD\_RTM\_1.0.0

BLN\_SMCAL\_4.0\_LEOPARD\_RTM\_1.0.0

* 1. For patch releases:

BLN\_MCAL\_3.0\_LEOPARD\_BETA\_P1\_0.9.1

BLN\_MCAL\_3.2\_BOLERO\_BETA\_P1\_0.9.0

BLN\_MCAL\_4.0\_LEOPARD\_RTM\_P1\_1.0.0

BLN\_SMCAL\_4.0\_LEOPARD\_RTM\_P1\_1.0.0

**For Project Process Documentation**

* 1. For JSW/JSW\_ASR4 Vobs:

BLN\_<PRODUCT\_NAME> \_<ASR\_VERSION>\_< PLATFORM>\_PROJECTDOC\_ <Baseline\_version>

Where:

<ASR\_MODULE> - Autosar Module

<PRODUCT\_NAME> - MCAL or SMCAL

<PLATFORM> - Platform Name, for current project: MPC56XXL

<Baseline\_version> - Baseline/Release Version of the type xx.yy.zz

Note: Process baseline will not be linked to the release baseline.

Ex:

BLN\_MCAL4.0\_KOMODO\_PROJECTDOC\_01.00

* 1. For SASW Vob:

BLN\_DOCS \_SMCAL\_<ASR\_VERSION>\_< PLATFORM>\_ <Baseline\_version>

Where:

<PLATFORM> - Platform Name, for current project: MPC56XXL

<Baseline\_version> - Baseline/Release Version of the type xx.yy.zz

Note: Process baseline will not be linked to the release baseline.

Ex:

BLN\_DOCS\_SMCAL\_4.0\_XPC5744P\_01.00.00

# MCAL/SMCAL Code Bases Synchronization

## MCAL QM Code Bases Synchronization

There are three main QM Code Bases:

* MCAL2.1 (following the old CM process, not described in the current procedure)
* MCAL3.0
* MCAL4.0

In scope for synch-up/out of synch-up scope:

1. **External New Feature Requests** are going through TWG 🡪managed in TWB BIN and propagated/cloned in each applicable QM Code Base
2. **External Improvements** are going through Customer Engineers CCB 🡪 propagated in each applicable QM Code Base
3. **External Defects** are going through Customer Engineers CCB 🡪 propagated in each applicable QM CB. (Note: external defects raised before December 2012, will be analyzed in small steps during Customer Engineers CCB meetings)

Customer Engineers CCB Meeting Minutes and CRs list are maintained at the below location:

<http://compass-phx.freescale.net/livelink/livelink?func=ll&objId=228154120&objAction=browse&sort=name>

1. **Internal New-Works** currently not synchronized
2. **Internal Defects** currently not synchronized

The QM Code Bases Synchronization Matrix is available in the below figure:

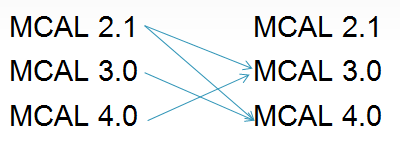


Figure 28 QM-QM Code Bases Synchronization Matrix

## MCAL QM - SMCAL Code Bases Synchronization

There are three main QM Code Bases and one Safety Code-Base:

* MCAL2.1 (following the old CM process, not described in the current procedure)
* MCAL3.0
* MCAL4.0
* SMCAL ( SMCAL development started from MCAL 4.0 CB in Q4 2012)

In scope for synch-up/out of synch-up scope:

1. **External New Feature** Requests Propagation by TWG – to be planned for the future
2. **External Improvements** are going through Customer Engineers CCB 🡪 propagated in each applicable QM CB. External improvements implemented in QM MCAL 4.0, are analyzed/need Safety Manager approval before cloning to SMCAL BIN
3. **External Defects** Propagation by CE CCB – started January 2013 for external defects raised starting with January 2013

Customer Engineers CCB Meeting Minutes and CRs list are maintained at the below location:

<http://compass-phx.freescale.net/livelink/livelink?func=ll&objId=228154120&objAction=browse&sort=name>

1. **Internal New-Works** currently not synchronized
2. **Internal Defects** propagation: Started February 2013 - Periodic Lists of Defects resolved on MCAL4.0 analyzed by SMCAL teams and propagated in SMCAL BIN if impacted

The mandatory synch-up from defects perspective is from MCAL4.0 to SMCAL – green line

The QM-SMCAL propagation matrix is available in the below figure:

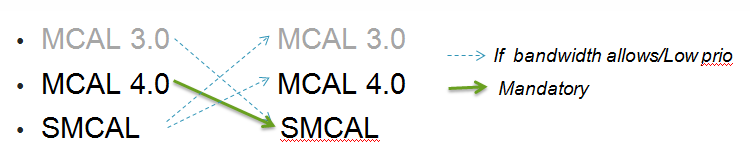


Figure 29 QM – SMCAL Synchronization Matrix

Synch-Up status between MCAL 4.0 and SMCAL is maintained on Project Management wiki page:

<http://wiki.freescale.net/display/MSGAutoSW/QA+-+MCAL+Codebases+Synchronization>