GENIVI Alliance

GENIVI Document TR00029

GENIVI Lifecycle Domain Overview

Technical Report

Version 1.0

2014-08-14

Sponsored by:
GENIVI Alliance

Abstract:
This document is an export of the GENIVI Wiki Lifecycle Domain. It provides an overview of the architecture and requirements for the complete Lifecycle Domain

Keywords:
GENIVI, Lifecycle

License:
This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
Copyright © 2014, Company ABC, Company XYZ.

All rights reserved.

The information within this document is the property of the copyright holders and its use and disclosure are restricted. Elements of GENIVI Alliance specifications may be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (and such third parties may or may not be members of GENIVI Alliance). GENIVI Alliance and the copyright holders are not responsible and shall not be held responsible in any manner for identifying, failing to identify, or for securing proper access to or use of, any or all such third party intellectual property rights.

GENIVI and the GENIVI Logo are trademarks of GENIVI Alliance in the U.S. and/or other countries. Other company, brand and product names referred to in this document may be trademarks that are claimed as the property of their respective owners.

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

The full license text is available at http://creativecommons.org/licenses/by-sa/4.0

The above notice and this paragraph must be included on all copies of this document that are made.

GENIVI Alliance
2400 Camino Ramon, Suite 375
San Ramon, CA 94583, USA
## Revision History

Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-08-14</td>
<td>1.0</td>
<td>David Yates</td>
<td>Initial revision exported from the GENIVI Wiki on the 14th Sept 2014</td>
</tr>
</tbody>
</table>
Subdomain Lifecycle

- Scope of work item
- Navigation within this subdomain
- Management of work item
  - Execution Projects
  - Risks, issues and open questions
- Subdomain documentation
- List of all persons contributed to EG SI Lifecycle Wiki
- Overview of Wiki pages
- Presentations for Lifecycle
  - Dublin
  - San Jose
  - Paris
  - Shanghai
  - Barcelona
  - San Diego
  - Misc. Presentations

Scope of work item

The scope of this work package is a full development activity of Lifecycle. That means starting with studying the past / history of this package; specify the system- and software architecture; doing the design and implementation of the common parts including the integration and test.

Finally the Lifecycle system architecture specification parts will be located in the GENIVI compliance architecture model. The software and the design specification you can find in the implementation model section of the GENIVI model.

However, additionally the interfaces will be delivered as header file independent if there will be a common implementation or not.

The functional scope of Lifecycle:

The current activity will cover:

- Interfaces to/of Thermal- and Supply management. Thermal- and Supply management itself will be handled as a black box only. Additionally the architecture is taking in account different deployments (one- and multi-node architectures).
  Please find more details at the concept description and following pages.

Navigation within this subdomain

At the bottom of each page linked arrows are located which helps to navigate within this subdomain. It provides the red-line for reading the pages.

Management of work item

If you have some remarks or concerns, you see some problems or you have just a question please fill in your item in the first table. If you
want to address some action, assigned or not, please use the second table.

These tables reflect our main status / activities.

## Execution Projects

<table>
<thead>
<tr>
<th>Project</th>
<th>WIKI page of the Node Startup Controller Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Startup Controller</td>
<td>WIKI page of the Node State Management Project</td>
</tr>
</tbody>
</table>

## Risks, issues and open questions

The column options are:

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>Submitter (Name, Company)</th>
<th>Topic</th>
<th>Answer</th>
<th>Replied by (Name, Company)</th>
</tr>
</thead>
</table>
| 1  | Question | Torsten Hildebrand, Continental | To release the compliant lifecycle requirements it is necessary to know who are the mandatory reviewers. | • Alexander Wenzel (BMW)  
• Manfred Batheil (BMW)  
• Christian Muck (BMW)  
• Fabien Hernandez (PSA) | Christian Muck, BMW |
| 2  | Question | David Yates, Continental | In the requirements the term "Middleware" is used - what does this refer to in the context of GENIVI? | The term "middleware" can be replaced with "GENIVI". It is also assumed that middleware doesn't provide a GUI. | Holger Behrens, Wind River |
| 3  | Question | David Yates, Continental | What is the GENIVI definition of the terms "Consumer" and "Device" as used in the Requirements? | Consumers - Interpretation depends on the HW partitioning in question. In case there is an automotive/vehicle controller, or some parts of the PSM are running on the vehicle controller (OSEK), while platform state handling is done on both CPUs (OSEK and Linux) by a bunch of SW components - aka. consumed Device - here my interpretation is that for example the PSM part running on the vehicle controller could potentially power-up/down the CPU running Linux (aka. system), or the CD/DVD drive (aka. device), based on platform state. | Holger Behrens, Wind River |
| 4  | Information | AMM, Paris | During the presentation on the Node State Manager (AMM, Paris) it was suggested by a number of listeners that we could reduce IPC overhead by having a static shutdown consumer registration process. | Following the meeting we have looked into whether this would be possible and come to the conclusion that currently this is not possible as we use the ordering of the registration during start-up to define the ordering for shutdown. Perhaps in the future something can be improved here with a static ordering for core components but currently we do not believe that the extra complexity of doing this is warranted to save one D-Bus message from each consumer during the start-up process. | David Yates, Continental |
During conversations with Lennart Poettering during the AMM, he suggested we look at systemd generators to see whether they are of use to us.

http://www.freedesktop.org/wiki/Software/systemd/Generators/

We have looked through the documentation for the Generators and can see that they are very useful for certain topics. However, we currently do not see a use case where they can assist in the current Lifecycle concept.

Theoretically we could have used generators to handle the LUC during the start-up process but dynamically altering the target files based upon the LUC persistency information before the systemd dependency graph is created. However, this would have required a similar amount of effort as is needed to do this in the Node Startup Controller as currently defined and would have delayed the starting of the Mandatory components.

Therefore we felt that we will currently leave the concept as is for now but will consider the use of generators in the future.

David Yates, Continental

### Jira item lists

In general all tasks and activities are maintained within Jira.

### Tasks list

This list of items gives you an overview about the main activities (tasks) within subdomain Lifecycle (including the status and the result). Therefore it reflects quite easily the subdomain status (what is done, what is ongoing, what next).

#### Sub-tasks of User Management (18 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Plannedstart</th>
<th>Due</th>
<th>Assignee</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GT-5</td>
<td>Lifecycle - System Architecture</td>
<td>Aug 05, 2011</td>
<td></td>
<td>David Yates</td>
<td>Reopened</td>
</tr>
<tr>
<td></td>
<td>GT-1297</td>
<td>Lifecycle - Boot Manager - Freeze SW Platform Req. and UC Realizations</td>
<td>Dec 21, 2011</td>
<td></td>
<td>David Yates</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-6</td>
<td>Lifecycle - Node Startup Controller - Component Realization and Compliance</td>
<td>Jan 25, 2012</td>
<td></td>
<td>David Yates</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-1299</td>
<td>Lifecycle - Node State Mgmt - Freeze Concept, SW Platform Req. and UC Realizations</td>
<td>May 04, 2012</td>
<td></td>
<td>David Yates</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2065</td>
<td>Milestone: Deliver implementation of phase 1 functionality to GENIVI for approval</td>
<td>26/Jun/12</td>
<td>Jun 26, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2057</td>
<td>Technical specifications and phase 1 of implementation</td>
<td>05/Jun/12</td>
<td>Jun 26, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2066</td>
<td>Incremental development of status reporting, startup cancellation, startup prioritization and launching of applications at runtime</td>
<td>27/Jun/12</td>
<td>Jul 17, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2068</td>
<td>Milestone: Deliver a feature-complete implementation of the Boot Manager including phase 2 functionality to GENIVI for approval</td>
<td>24/Jul/12</td>
<td>Jul 24, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2067</td>
<td>Improvements of the code and product quality by means of testing, reviewing and cleaning</td>
<td>17/Jul/12</td>
<td>Jul 24, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2058</td>
<td>Phase 2 of implementation and bug fixing/quality improvements</td>
<td>27/Jun/12</td>
<td>Jul 24, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2069</td>
<td>Write manual pages and API documentation</td>
<td>25/Jul/12</td>
<td>Jul 27, 2012</td>
<td>Unassigned</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-1300</td>
<td>Lifecycle - Node State Mgmt - Freeze final Architecture and SOW</td>
<td>Jul 27, 2012</td>
<td></td>
<td>David Yates</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2070</td>
<td>Make testing the unit startup implementation of the Boot Manager</td>
<td>30/Jul/12</td>
<td>Aug 03, 2012</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>
If you can not see the Jira issue lists, you are not member of the Jira Project Tracking group.

**Action list**

This list give you all actions which are tracked with Jira for the subdomain Lifecycle. Those actions supports the actual Lifecycle tasks.

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Planned</th>
<th>Due</th>
<th>Assignee</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT-552</td>
<td>Care about getting the cancellation of an ongoing startup procedure into the system architecture guidelines</td>
<td>Feb 02, 2012</td>
<td>David Yates</td>
<td>In Progress</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT-556</td>
<td>Check about the policy of the out of memory killer behavior in the kernel</td>
<td>Apr 25, 2013</td>
<td>David Yates</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT-555</td>
<td>There should be investigations on how the cgroups feature of the kernel works today across multiple CPUs.</td>
<td>Apr 30, 2013</td>
<td>David Yates</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT-554</td>
<td>There should be a good explanation about how to use exceptional resource management and what are pitfalls in the documentation and guidelines</td>
<td>May 24, 2013</td>
<td>David Yates</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT-557</td>
<td>See if BMW can provide some feasibility test cases on how to use cgroups</td>
<td></td>
<td>Christian Muck</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If you can not see the Jira issue lists, you are not member of the Jira Project Tracking group.

**All closed tasks and actions**

All closed items

**Communication**

**Minutes**

In general there is no Lifecycle specific teleconferences. Therefore no separate minutes are done. The list of all EG-SI minutes can be found here.

Here are some interesting minutes for this topic:

- Kickoff meeting for requirements and documentation collection ([SysInfraEGMinutes20110210](#))
- Kickoff meeting of this work item ([SysInfraEGMinutes20110113](#))
- Presentation of systemd - filesystem mounting ([SysInfraEGMinutes20110317](#))
- Kickoff meeting as F2F for the execution project Bootmanager [Nowadays renamed to NodeStartupController] ([SysInfraEGMinutes20120529](#))

**Telco**

Currently it is not planned to have a special regular cyclic teleconference for this work item Lifecycle. In case there is a need, please send an email to me (Torsten Hildebrand) with the item to be discussed.

**Released assets**
This table will contain all important released documentation, specification and information.

The column options are:
Status: Reviewed, Released, Archived

<table>
<thead>
<tr>
<th>ID</th>
<th>Name of asset</th>
<th>Purpose of asset</th>
<th>Author (Name,Company)</th>
<th>Release date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20110322_Mounting_with_systemd.ppt</td>
<td>Explanation of systemd filesystem parallelization and mounting works</td>
<td>Christian Muck (BMW)</td>
<td>2011-03-17</td>
<td>Reviewed</td>
</tr>
</tbody>
</table>

**Subdomain documentation**

The Subdomain documentation (SysInfraEGLifecycleSubsysDoc) will contain the following:

- A list of former/legacy documentations and specifications
- Lifecycle in system context
- Requirements analysis (the compliance requirements, use cases and scenarios)
  - Vehicle Requirements
  - SW Platform Requirements
- Functional analysis (the black box realization based on use cases and scenarios)
  - Use Cases (from vehicle point of view)
  - Static views
  - Dynamic views
- Design
  - Structure definition
    - Blocks / Components
    - Activity diagrams
  - Public interfaces
  - Design analysis (the white box realization based on use cases and scenarios)
  - Process views
- Integration
  - Dependencies
  - Deployment
- Test / Compliance
  - Test criteria
  - Test cases

**List of all persons contributed to EG SI Lifecycle Wiki**

David Yates 290 , Torsten Hildebrand 233 , Jannis Pohlmann 148 , Michael Kerrisk 122 , Christian Muck 100 , Ben Brewer 29 , Gunnar Andersson 21 , Fabien Hernandez 20 , Markus Boje 11 , Axel Endler 9 , Sriram Gopalan 8 , Jonathan Maw 5 , Jean-Pierre Bogler 3 , Manfred Bathelt 3 , Erik Boto 2 , Holger Behrens 2 , Alex Brankov 1

**Overview of Wiki pages**

The tree representation shows all child pages of this parent side:

- SysInfraEGLifecycleClosedItems
- SysInfraEGLifecycleExecPrjctBootManager
  - Node Startup Controller Description
  - SysInfraEGLifecycleExecPrjctBootManagerArchitectureOverview
  - SysInfraEGLifecycleExecPrjctBootManagerDBusInterfaces
    - ReviewBootManagerDBusInterfaces
  - SysInfraEGLifecycleExecPrjctBootManagerDependencies
  - SysInfraEGLifecycleExecPrjctBootManagerExcaliburSchedule
  - SysInfraEGLifecycleExecPrjctBootManagerFunctionalScope
  - SysInfraEGLifecycleExecPrjctBootManagerInternalArchitecture
  - SysInfraEGLifecycleExecPrjctBootManagerMinutes20120611
  - SysInfraEGLifecycleExecPrjctBootManagerMinutes20120618
  - SysInfraEGLifecycleExecPrjctBootManagerMinutes20120625
  - SysInfraEGLifecycleExecPrjctBootManagerMinutes20120702
  - SysInfraEGLifecycleExecPrjctBootManagerMinutes20120709
  - SysInfraEGLifecycleExecPrjctBootManagerMinutes20120716
  - SysInfraEGLifecycleExecPrjctBootManagerMinutes20120813
  - SysInfraEGLifecycleExecPrjctBootManagerRequirements
  - SysInfraEGLifecycleExecPrjctBootManagerTestScenarios
  - SysInfraEGLifecycleExecPrjctBootManagerTrackingInJIRA
  - SysInfraEGLifecycleExecProjectBootManagerLegacyApps
  - SysInfraEGLifecycleExecProjectBootManagerSystemdInteraction
  - SysInfraEGLifecycleExecProjectBootManagerSystemdInteraction
- SysInfraEGLifecycleExecPrjctNodeResMgmt
- SysInfraEGLifecycleExecPrjctNSM
- Node State Manager Description
Presentations for Lifecycle

This section contains links to all the presentations that have been made for the Lifecycle domain during EG meetings and AMM's

Dublin

systemd_lennart.pptx

San Jose

GENIVI_Lifecycle_NSMData.ppt
GENIVI_AMM_SanJose_LifecyclePOC.ppt
Lifecycle Model Walk Through and System Shutdown Concept.pptx

Paris

AMM_Paris_LifecycleNSM.ppt
AMM_Paris_LifecycleResourceMgmt.ppt

Shanghai

GENIVI AMM Lifecycle.ppt
GENIVI AMM LifecycleStatus.pptx

Barcelona

Recording from the presentation of the GENIVI Resource Management concept

(If the video embedding above does not work in your browser, then download the attachment.)

GENIVI AMM Resource Management.ppt
GENIVI AMM Lifecycle Working Session.ppt
HealthMonitoring.doc
UCR_DynamicResources.doc

San Diego

GENIVI San Diego Infrastructure Usage.pptx
Persistency_SWL.doc
ProfileManagerUsageQuery_SWL.doc
NodeHealthMonitor_SWL.docx

Misc. Presentations

This section contains some misc. presentations used for presenting proposed changes in components.

GENIVI Lifecycle LUM Handling.ppt
## SysInfraEGLifecycleClosedItems

### Subdomain Lifecycle - All closed Jira items / issues

<table>
<thead>
<tr>
<th>JIRA Issues (64 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Issue ID</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>GT-1424</td>
</tr>
<tr>
<td>GT-1347</td>
</tr>
<tr>
<td>GT-1346</td>
</tr>
<tr>
<td>GT-941</td>
</tr>
<tr>
<td>GT-914</td>
</tr>
<tr>
<td>GT-568</td>
</tr>
<tr>
<td>Agreement</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>GT-553</td>
</tr>
<tr>
<td>GT-551</td>
</tr>
<tr>
<td>GT-272</td>
</tr>
<tr>
<td>GT-270</td>
</tr>
<tr>
<td>GT-267</td>
</tr>
<tr>
<td>GT-266</td>
</tr>
<tr>
<td>GT-260</td>
</tr>
</tbody>
</table>
SysInfraEGLifecycleExecPrjctBootManager

Execution Project Node Startup Controller

1. Team

GENIVI representatives:
- Gunnar Andersson (EG-SI Lead)
- David Yates (GENIVI Project Manager / Mentor, Topic Lead Lifecycle)

Codethink representatives:
- Alex Brankov (Codethink Project Manager)
- Jannis Pohlmann (Technical Lead)
- Ben Brewer (Backup Technical Lead)
- Francisco Marchena (Engineer)
- Jonathan Maw (Engineer)

2. Statement of Work

This execution project aims at transforming the Node Startup Controller (former called Boot Manager) from an Abstract Component into a Specific Component by delivering an implementation of the Node Startup Controller to be integrated into the GENIVI Excalibur release.

This wiki page provides an overview of all information generated as part of the project. This includes the project schedule, requirements, an overview of the architecture, pointers to the implementation, documents and technical specifications as well as progress reports in the form of telco minutes.

3. Project Schedule

- Schedule and project tracking for Excalibur

4. Weekly Status Reports

This section will include summaries of all weekly progress reports.
- cw2412
- cw2512
- cw2612
- cw2712
- cw2812
- cw2912
5. Requirements

- Consolidated Requirements Specification
- Excalibur Dependencies

6. Architecture Overview

- Functional Scope
- Architecture Overview
- Internal Software Architecture
- Interaction with systemd
- Legacy application start/shutdown

7. Documents and Specifications

- Node Startup Controller D-Bus interfaces
- Test Scenarios

8. Implementation

The source code repository for the boot manager is available here:
https://git.genivi.org/git/gitweb.cgi?p=node-startup-controller;a=summary

Working with GENIVI Git repositories is described this wiki page.

9. Baseline Integration Status

This section will summarise the integration of the Node Startup Controller implementation into the GENIVI baseline.

10. Telcos

Weekly telcos have been set up on Mondays, 15:00 - 16:00 CET. This section lists the minutes of these calls.

- Telco 2012-06-11
- Telco 2012-06-18
- Telco 2012-06-25
- Telco 2012-07-02
- Telco 2012-08-13

11. Overview of Wiki Pages

- Node Startup Controller Description
- SysInfraEGLifecycleExecPrjctBootManagerArchitectureOverview
- SysInfraEGLifecycleExecPrjctBootManagerDBusInterfaces
  - ReviewBootManagerDBusInterfaces
- SysInfraEGLifecycleExecPrjctBootManagerDependencies
- SysInfraEGLifecycleExecPrjctBootManagerExcaliburSchedule
- SysInfraEGLifecycleExecPrjctBootManagerFunctionalScope
- SysInfraEGLifecycleExecPrjctBootManagerInternalArchitecture
- SysInfraEGLifecycleExecPrjctBootManagerMinutes20120611
- SysInfraEGLifecycleExecPrjctBootManagerMinutes20120618
- SysInfraEGLifecycleExecPrjctBootManagerMinutes20120625
- SysInfraEGLifecycleExecPrjctBootManagerMinutes20120702
- SysInfraEGLifecycleExecPrjctBootManagerMinutes20120709
- SysInfraEGLifecycleExecPrjctBootManagerMinutes20120716
- SysInfraEGLifecycleExecPrjctBootManagerMinutes20120813
- SysInfraEGLifecycleExecPrjctBootManagerRequirements
- SysInfraEGLifecycleExecPrjctBootManagerTestScenarios
- SysInfraEGLifecycleExecPrjctBootManagerTrackingInJIRA
- SysInfraEGLifecycleExecProjectBootManagerLegacyApps
- SysInfraEGLifecycleExecProjectBootManagerSystemdInteraction
- SysInfraEGMinutes20120529

Node Startup Controller Description

This is the draft proposal of the public web page text for NSC.
Introduction

The Node Startup Controller (NSC) was introduced into the lifecycle package for GENIVI in order to handle some startup and shutdown functionality. It essentially "extends" systemd to implement some IVI requirements that are not done by systemd itself because they are not generally applicable for all Linux systems (as determined through discussion with the systemd community). However, similar functionality might be desired in non-automotive systems so we hope this can be useful and/or develop into something shared across domains.

The main areas of responsibility for the NSC are:

- Last User Context (LUC) Management
- Legacy Application Shutdown
- Target Startup Monitoring

Last User Context (LUC) Management

The Last User Context (LUC) Management holds information about what applications the user was using in the last lifecycle in order to allow the same applications to be restored automatically or prioritized over other applications the next time the system boots up. Applications are stored in the LUC in groups whose start order is defined at build time. The choice is fully flexible but for a typical Automotive IVI system, these groups could be:

- Foreground application
- Audible application
- Background applications

The LUC management simply stores the systemd unit name (e.g. navigation.service) for each LUC-managed application and communicates with systemd to start up these applications shortly after boot. Systemd D-Bus interfaces are used instead of simply a stored systemd target file on disk. This is to allow the behavior to be flexible according to system policy and because the behavior might also be dynamic as described in the next section. In the current implementation some configuration is available to control the startup order.

Legacy Application Shutdown

In this context "Legacy applications" means applications that do provide a systemd unit file but are otherwise unaware of, or do not make use of other GENIVI/Lifecycle components. This means that the applications will not actively register themselves as a shutdown consumer in the Node State Manager (NSM). (That is the normal way for the NSM to know what applications are active and can be controlled in order to sleep/shutdown. To solve the "legacy applications" the NSC provides a mechanism to separately register such applications as shutdown consumers.

Whenever the Node State Manager decides to perform a shutdown it will ask the shutdown consumers to shut down in reverse order of their creation. To the NSM it does not matter whether or not the consumers are registered by applications themselves or by the Node Startup Controller (the "legacy applications" case).

There is some other behavior also during shut-down which should be mentioned here for completeness, although it is controlled by the Node State Manager (NSM) as opposed to NSC. Specifically it is the requirement to implement "cancelled shutdown" which is not typically present in non-IVI Linux systems.

Target Startup Monitoring

The Node Startup Controller is responsible to inform the Node State Manager when a particular boot state has been reached (e.g. focused.target, lazy.target are completed). To achieve this the Node Startup Controller (NSC) monitors signals from systemd indicating the completion of target/unit completion and then contacts NSM through its D-Bus interface.

SysInfraEGLifecycleExecPrjctBootManagerArchitectureOverview

Boot Manager Architecture Overview

The high-level architecture of the boot manager implementation looks is defined as follows:
The functionality described on the Functional Scope page is provided by three different D-Bus services. These services and their interaction with the rest of the system are described in this document.

All D-Bus interfaces are versioned to allow incompatible future interfaces to be developed using the same naming prefix.

The interfaces org.genivi.BootManager.BootManager1 and org.genivi.BootManager.LegacyAppHandler1 are implemented in a executable for start-up performance reasons.

**org.genivi.BootManager.BootManager1**

This service implements and provides the following functionality:

- **LUC Management**:
  - A method to register one or multiple applications with the LUC; each of these applications is associated with one of the three types (audible, background, foreground)
  - A method to deregister one or multiple applications with the LUC; each of the applications is optionally associated with a type for which it should be removed from the LUC

The org.genivi.BootManager.BootManager1 implementation interacts with the rest of the system as follows:

- It is started by systemd.
- It registers itself with systemd's watchdog mechanism.
- It subscribes to the org.freedesktop.systemd1.Manager service to receive JobRemoved signals.
- It registers itself with the com.conti.NodeStateManager.Consumer service as a shutdown consumer at start-up.
- It queries the com.conti.NodeStateManager.LifecycleControl interface for whether or not it should restore the LUC.
- If required, it restores the LUC applications by starting them in three groups.
- It registers itself with the com.conti.NodeStateManager.Consumer service as a shutdown consumer at start-up.
- It allows applications and other components to register/deregister applications with the LUC.

**org.genivi.BootManager.LegacyAppHandler1**

This service implements and provides the following functionality:

- **Legacy Application Shutdown**
  - A method to register a new shutdown consumer for an arbitrary unit

The org.genivi.BootManager.LegacyAppHandler1 implementation interacts with the rest of the system as follows:

- It is started by systemd either because it is implemented in the same executable as org.genivi.BootManager.BootManager1 or because a legacy application has been started and the ExecStartPost command is being executed.
- It registers itself with systemd's watchdog mechanism.
- For each legacy application that is registered with the org.genivi.BootManager.LegacyAppHandler1, it creates a new NSMShutdownConsumer object and registers it with the com.conti.NodeStateManager.Consumer service as a shutdown consumer.
**Boot Manager D-Bus Interfaces**

NOTE: The interfaces on this page have not been finalized yet.

- `org.genivi.BootManager1`
  - Interface Specification
  - Data Exchange Format
- `LegacyAppHandler`
  - Interface Specification

### org.genivi.BootManager1

**Interface Specification**

```java
package BootManager;

/** @description: Type used for systemd unit names, e.g., "appl.service". ***/
typedef UnitName is String

/** @description: Type for arrays of systemd unit names. ***/
array UnitNameArray of UnitName

/** @description: Interface for managing the GENIVI LUC (Last User Context) **/

The GENIVI BootManager remembers applications that were used in the last session of a user (LUC). It uses this LUC data to restore these applications on the next start-up.

The BootManager is a passive component in the sense that it does not remember applications on its own; instead, applications need to register and deregister themselves proactively.

Applications can be registered for different LUC types. The allowed values for LUC types are: "foreground", "background" and "audible". Other values will be disregarded.

```
The BootManager must be set into the LUC registration state before RegisterWithLUC is called.

@details: org.freedesktop.DBus.GLib.Async == true

**>
method BeginLUCRegistration{
}

<** @description: Registers one or more applications for certain LUC types.

Applications may be listed multiple times. For LUC types where only a single application may be registered at a time, the last application in the corresponding list wins.

The Register() method may be called multiple times. Every invocation will add to the current LUC rather than overwriting it.

Unknown LUC types will be ignored.

@details: org.freedesktop.DBus.GLib.Async == true

**>
method RegisterWithLUC {
  in {
    <** @description: A mapping of LUC types to unit name arrays. **>
    LUCMap apps
  }
}

<** @description: Finishes the LUC registration.

Writes the LUC back to disk atomically and deletes the previous LUC.

The BootManager must be set into the LUC registration state before this is called.

@details: org.freedesktop.DBus.GLib.Async == true

**>
method FinishLUCRegistration{
Data Exchange Format

- The `apps` parameter of the `Register()` and `Deregister()` methods is a dictionary with the following structure:

```python
{
    "audible":    [ "app3.unit" ],
    "background": [ "app2.unit" ],
    "foreground": [ "app1.unit", "app2.unit" ]
}
```

where the different LUC types are the keys of the dictionary and the values of the dictionary are lists of application unit names.

- The LUC types are encoding as strings to allow them to be easily introspected and extended in the future.
- The `LastUserContext` is a D-Bus property and can be accessed and monitored. It uses the same format as the `apps` parameter.

LegacyAppHandler

Interface Specification

The LegacyAppHandler uses a command-line interface (CLI) to register apps on start-up using the systemd ExecStartPost field.

It is an interface for registering legacy apps with the NSM.

Legacy applications are applications that provide a systemd unit file but are unaware or do not make use of any GENIVI components.

The GENIVI Legacy App Handler registers these apps with the NSM (Node State Manager) as shutdown consumers, so that when the NSM performs a shutdown it can shut down the application in reverse order of their creation.

The basic interface is:

```bash
legacy-app-handler <unit-name> <mode> <timeout>
```

- `unit-name`: An application unit filename.
- `mode`: Shutdown mode for which to register.
- `timeout`: Time to wait (ms) for the application to shut down (integer).

ReviewBootManagerDBusInterfaces

Review meeting for Boot Manager Interfaces

2011 June 26, 1300–1440

Participants

- Gianpaolo
- Torsten
- David
- Jannis
- Klaus (until 1400)
- Gunnar
- Manfred

Discussion on process.

Should we use Jira items? One for the topic of review and subitems for things that need to be discussed and to document input. (Towards the end of the meeting decided this is not typical for GENIVI - Mail threads and then summarizing the mail thread outcome is what will be used).

For now we’ll start with these minutes.
Topics:

Is BM duplicating systemd interface?

Is Boot Manager only wrapping systemd, why is BM needed and what does it add?

- Abstraction (non-dependence) of systemd was desired.
  - Legacy applications that know systemd could be tempted to use systemd directly. BM interfaces a bit more controlled.
  - Some simplification. systemd interface is a little more complicated.

  Manfred: BM interface seems to be synchronous. How about concurrent activity? how will that be handled?

  Sriram: All are good reasons, but is BM the right candidate that should fulfill this responsibility?
  Sriram: The term legacy application is not clear.

  BM is an extension of systemd
  - Start and Stop will have conditions - maybe not always allowed. We cannot add such conditions to systemd itself.

  Manfred: we never discussed to try this, did we? Lennart Poettering already visited some Genivi meetings in person, and was very open for changes.

- For example during a shutdown procedure, it might not be allowed to perform Start on a unit.

- All components normally allowed to use systemd interface. BM interface would be limited to only some client, i.e., systemd direct access is not recommended for the equivalent systemd interfaces (Start, Stop, ...). Some systemd functions are still planned to be used directly: Example watchdogs. An application must use the systemd notify interface when completed its init.

  Sriram: It is far more confusing to use 2 components that overlap in services. Either there should be a full abstraction of systemd OR systemd should be the interface.

Manfred describes a different way to limit/configure systemd behavior that would achieve similar result. Configure systemd D-Bus interface so that only one specific component can access systemd. This component would implement policies.

Group majority feels this is a design change which we cannot use now.

  Sriram: I think wrapping systemd is also a design/architecture change, albeit an easy one. However that may not be the right one in the longer term how other components will use the wrapper

However, it would be good to pick up and fully understand Manfred's idea. Manfred would like to understand if there is a plugin to change the behavior.

Manfred, please write down in email. What is the requirement?

String types to be defined as type defs (with limited value range).

- Cannot control this in D-Bus API. We do not expect all applications to have a common header file?
  - A client library could be created later that helps applications use the correct values.
    + Compile time checking is good but build dependencies arise from including other components' header files.
    Will we have common header files?

  Sriram: What is the constraint in having common header files across GENIVI? I think it will lead to elegant implementation.

  Manfred: can't we use also FrancaIDL definitions instead of common header files?

+ Typedefs can still be used in interface description (Franca) just to define the value range - this should not necessarily imply the usage of header files with strict types.

Define typedef in Franca.

Gunnar create JIRA Item for the bigger discussion.

LUCHandler Deregistration function needed?

1. Some believe the list in LUC will be cleared at each new lifecycle anyway, and that Register will be used typically at shutdown to store the most recent state, so no need to deregister.
2. OTOH if a system designer want to keep LUC updated at all times (to handle for example unexpected shutdown due to power loss), then continuous Register/Deregister is necessary to keep LUC updated.
3. We have other requirements that assume that LUCHandler SHALL clear the LUC at the start of Shutdown. If we keep these, then situation 2. is not possible. This solution will control the persistence of LUC in an easier way. Versioning of the LUC in Persistence could also be possible (use "old" if power-loss creates a problem)

  Manfred: more interesting is the Registration: it needs to be done in the critical path during startup, and there is only ONE and the same component registered in each lifecycle again and again. Registering it via D-Bus means the LUCHandler needs to reside in an own process. Is there an estimate how this could affect performance?

There needs to be a method to notify that LUC has been completely updated. (Typically the implementation might use this to initiate persistent storage, but that is a detail not known to the applications).

The discussion creates new ideas. It seems to need some redesign.

Gunnar Create Jira item for this also

Separate mail threads are needed. Most are used to work that way, also in open source projects.
- Map to Array of strings - too complex type for API?
  Gunnar, genivi-dev, unless already adequately explained by Jannis’ answer.

- String values identical with systemctl - may change?
  Gunnar, genivi-dev, unless already adequately explained by Jannis’ answer.

- Is reporting unit names enough for the List method, or should we give more information?
  Jannis, genivi-dev

- systemctl Start modes needed to be exposed in interface?
  Jannis, genivi-dev

- LegacyAppHandler separate component talking to NSM. Why do clients not talk directly to NSM?
  Gunnar, genivi-dev unless already adequately explained by Jannis’ answer.

- Enum vs String.
- Strings are easier to understand for tests, debugging
  General question. genivi-dev / architect

| Sriram: I personally always vote for stronger type. enum also has an advantage of efficiency and error-free characteristics in implementation |
| Manfred: fully support Sriram’s comment |

SysInfraEGLifecycleExecPrtctBootManagerDependencies

Dependencies of the Boot Manager

For Excalibur

<table>
<thead>
<tr>
<th>Name</th>
<th>Minimum Required Version</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>systemd</td>
<td>&gt;= 183</td>
<td>For control over units; 183 required for monitoring target startup and isolation functionality for the node application mode</td>
</tr>
<tr>
<td>Linux</td>
<td>&gt;= 2.6.39</td>
<td>Required by systemd 183</td>
</tr>
<tr>
<td>GLib</td>
<td>&gt;= 2.30</td>
<td>For object-oriented and event-driven implementation, main loop functionality and GDBus</td>
</tr>
<tr>
<td>Automotive DLT</td>
<td>&gt;= 2.2.0</td>
<td>Mandatory for all GENIVI components</td>
</tr>
<tr>
<td>Node State Manager</td>
<td>&gt;= ? (Excalibur version)</td>
<td>For LUC start-up query, setting system state changes and registration of shutdown consumers</td>
</tr>
</tbody>
</table>

SysInfraEGLifecycleExecPrtctBootManagerExcaliburSchedule

Boot Manager Schedule for Excalibur

1. Project Timeframe

<table>
<thead>
<tr>
<th>Start Date</th>
<th>Summary description of activities</th>
<th>Status</th>
<th>Target date</th>
<th>Expected Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-05-23</td>
<td>Implementation of the Boot Manager</td>
<td>in progress</td>
<td>2012-08-27</td>
<td>2 engineers full-time</td>
</tr>
</tbody>
</table>

2. Work packages

<table>
<thead>
<tr>
<th>Start Date</th>
<th>Summary description of activities</th>
<th>Status</th>
<th>Target date</th>
<th>Expected Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-05-23</td>
<td>Requirements analysis and high-level design</td>
<td>finished</td>
<td>2012-06-04</td>
<td>2 engineers full-time</td>
</tr>
<tr>
<td>2012-06-05</td>
<td>Technical specifications and phase 1 of implementation</td>
<td>finished</td>
<td>2012-06-26</td>
<td>2 engineers full-time</td>
</tr>
</tbody>
</table>
3. Project Tracking in JIRA

The main entry point for information about the status of the project in JIRA is the following task: https://collab.genivi.org/issues/browse/GT-2055.

4. Milestones in JIRA

### Milestones of the Boot Manager Execution Project (4 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Plannedstart</th>
<th>Due</th>
<th>Assignee</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GT-2060</td>
<td>Milestone: Deliver final design and project plan to GENIVI for assessment and approval</td>
<td>04/Jun/12</td>
<td>Jun 04, 2012</td>
<td>Jannis Pohlmann</td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>GT-2065</td>
<td>Milestone: Deliver implementation of phase 1 functionality to GENIVI for approval</td>
<td>26/Jun/12</td>
<td>Jun 26, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2068</td>
<td>Milestone: Deliver a feature-complete implementation of the Boot Manager including phase 2 functionality to GENIVI for approval</td>
<td>24/Jul/12</td>
<td>Jul 24, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2073</td>
<td>Milestone: Package integrated into GENIVI release and handover of final documentation</td>
<td>21/Aug/12</td>
<td>Aug 21, 2012</td>
<td>Jannis Pohlmann</td>
<td>Closed</td>
</tr>
</tbody>
</table>

5. Tasks in JIRA

### Tasks of the Boot Manager Execution Project (17 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Plannedstart</th>
<th>Due</th>
<th>Assignee</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GT-1298</td>
<td>Lifecycle - Boot Manager - Freeze final Architecture and SOW</td>
<td>Dec 07, 2011</td>
<td></td>
<td>David Yates</td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>GT-1297</td>
<td>Lifecycle - Boot Manager - Freeze SW Platform Req. and UC Realizations</td>
<td>Dec 21, 2011</td>
<td></td>
<td>David Yates</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-6</td>
<td>Lifecycle - Node Startup Controller - Component Realization and Compliance</td>
<td>Jan 25, 2012</td>
<td></td>
<td>David Yates</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2061</td>
<td>Liaise with BI and NSM team, compile a full description of all functional and non-functional requirements of the Boot Manager</td>
<td>23/Mai/12</td>
<td>May 29, 2012</td>
<td>Jannis Pohlmann</td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>GT-2062</td>
<td>Define Boot Manager components and their roles/responsibilities; produce final design document</td>
<td>01/Jun/12</td>
<td>Jun 04, 2012</td>
<td>Jannis Pohlmann</td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>GT-2056</td>
<td>Requirements analysis and high-level design</td>
<td>23/Mai/12</td>
<td>Jun 04, 2012</td>
<td>Gunnar Andersson</td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>GT-2063</td>
<td>Specification of service interfaces, model interaction with systemd; design internal software architecture</td>
<td>05/Jun/12</td>
<td>Jun 12, 2012</td>
<td>Jannis Pohlmann</td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>GT-2064</td>
<td>Incremental development of LUC management and LUC-based unit startup</td>
<td>13/Jun/12</td>
<td>Jun 26, 2012</td>
<td>Jannis Pohlmann</td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>GT-2057</td>
<td>Technical specifications and phase 1 of implementation</td>
<td>05/Jun/12</td>
<td>Jun 26, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2066</td>
<td>Incremental development of status reporting, startup cancellation, startup prioritization and launching of applications at runtime</td>
<td>27/Jun/12</td>
<td>Jul 17, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2067</td>
<td>Improvements of the code and product quality by means of testing, reviewing and cleaning u</td>
<td>17/Jul/12</td>
<td>Jul 24, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>GT-2058</td>
<td>Phase 2 of implementation and bug fixing/quality improvements</td>
<td>27/Jun/12</td>
<td>Jul 24, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
</tbody>
</table>
GT-2069  Write manual pages and API documentation  25/Jul/12  Jul 27, 2012  Unassigned  Resolved

GT-2070  Make testing the unit startup implementation of the Boot Manager reproducible  30/Jul/12  Aug 03, 2012  Unassigned  Resolved

GT-2071  Prepare compliance documents required for delivery into the GENIVI BIT team  06/Aug/12  Aug 10, 2012  Unassigned  Resolved

GT-2072  Packaging the Boot Manager and aiding BIT team with the integration into the GENIVI release  13/Aug/12  Aug 20, 2012  Jannis Pohlmann  Resolved

GT-2059  Documentation and integration  25/Jul/12  Aug 27, 2012  Unassigned  Resolved

SysInfraEGLifecycleExecPrjctBootManagerFunctionalScope

Functional Scope of the Boot Manager

- Overview
- LUC Management
- Dynamic Start-up Management
- Legacy Application Shutdown
- Target Startup Monitoring
- Node Application Mode Activation

Overview

Given the requirements specification, the following functional scope for the boot manager implementation has been identified:

![Boot Manager Diagram]

LUC Management

The responsibility of LUC (Last User Context) Management is to manage information about applications the user is running in order to allow the same applications to be restored after a reboot of the system. The LUC supports three different types of applications:

- **Audible applications**
  - Applications that are current audible sources within the Head Unit (e.g. radio or CD player)
  - The LUC only allows one audible application to be registered at a time
  - Whatever app registers itself with the LUC as a audible app last overwrites any previous audible app

- **Foreground applications**
  - Applications that are currently in the focus of the HMI (e.g., navigation or web browser)
  - The LUC only allows one foreground application to be registered at a time
  - Whatever app registers itself with the LUC as a foreground app last overwrites any previous foreground app

- **Background applications**
  - Applications that run in the background but are neither in the foreground nor audible sources
  - The LUC allows multiple applications to be registered as background applications at the same time

A registration and deregistration with the LUC consists of two parameters:

1. A systemd unit filename (e.g. navigation.service),
2. The LUC type, which is one of the following strings: audible, foreground or background.
Audible, foreground and background applications are treated as start-up groups, meaning that e.g. all background apps are started in parallel. The order in which these three groups are started by the LUC Management can be configured at build-time.

In order to reduce the amount of communication with the LUC management, multiple applications can be registered and deregistered with the LUC at once.

**Dynamic Start-up Management**

One of the main responsibilities of the boot manager is to restore the LUC. In addition to this, the boot manager provides an interface for other components to request that other applications be started as soon as possible. This is important to react to situations where the user presses a button in the HMI in order to bring up a certain application (e.g., navigation) during boot.

User events are monitored by the application event handler. If a user event results in the need to start an application, this can be forwarded to the boot manager using its Runtime Application Management features. No special interface is required for this.

**Legacy Application Shutdown**

Legacy applications are applications that provide a systemd unit file but are unaware or do not make use of any GENIVI components. This also means that they do not register with the NSM (Node State Manager) as a shutdown consumer, which is a requirement for any application in GENIVI.

To solve this problem the boot manager provides a mechanism to register shutdown consumers for individual legacy applications. This works as follows:

1. the boot manager provides a D-Bus method for registering a shutdown consumer for a given unit filename,
2. the boot manager provides a helper script or binary that takes a unit filename and calls the above D-Bus method to register a shutdown consumer for this unit file,
3. an ExecStartPost command is added the unit files of legacy applications that calls the helper script or binary.

Whenever the NSM decides to perform a shutdown it will ask the shutdown consumers to shut down in reverse order of their creation. To the NSM it does not matter whether or not the consumers are registered by applications themselves or by the boot manager.

**Target Startup Monitoring**

The boot manager is responsible to set certain NSM states when certain targets (e.g. focused.target, lazy.target) have been started within or outside the boot manager. For this, the boot manager needs to monitor systemd for unit start-up events.

As of systemd 183, this is possible through systemd's JobRemoved signal. The boot manager

- sets the NSM state to BASE_RUNNING during initialization,
- subscribes to systemd in order to receive signals from systemd,
- evaluates the received JobRemoved signals by
  1. filtering out the signals that do not belong to target start-up events
  2. setting the NSM state to
     - LUC_RUNNING when focused.target has been started,
     - FULLY_RUNNING when unfocussed.target has been started,
     - FULLY_OPERATIONAL when lazy.target has been started.

**Node Application Mode Activation**

Node application modes (such as regular, transport, loading etc.) can be activated using two different ways:

1. the system is started and already knows about the desired node application mode at boot-time,
2. the system is already running and needs to switch from one mode to another.

The boot manager does not need to be aware of the list of supported modes. The two above scenarios are handled as follows:

1. When the system is started, the boot loader will set the final target (e.g. transport.target). All the boot manager has to know is whether or not to restore the LUC when starting up. This information is provided by the NSM via a GetRestoreLUC() query method.
2. The NSM or some other component initiates the switch from one mode to another at run-time. The boot manager provides an Isolate(target) method that can be used to switch to a different mode; the target would e.g. be transport.target or loading.target.

**SysInfraEGLifecycleExecPrjctBootManagerInternalArchitecture**

**Internal Software Architecture of the Boot Manager**

This page lists the expected components/classes that will end up being part of the internal software architecture of the boot manager implementation.

- Common Components
- Boot Manager Components
- Legacy App Handler Components

**Common Components**
The common components are defined as GObject classes in the `common/` directory of the boot manager source tree.

- **WatchdogClient**
  - instantiated once for each service daemon
  - responsible for setting up a timeout to call in regular intervals `sd_notify("WATCHDOG=1")`
- **ShutdownConsumer**
  - generated from the shutdown consumer D-Bus interface specification
  - used as a skeleton to implement the `Shutdown()` method called by the NSM during the shutdown phase
- **NSMConsumerProxy**
  - generated from the NSM Consumer D-Bus interface specification
  - used to register `ShutdownConsumer` objects with the NSM
  - instantiated once by each service daemon to register itself as a shutdown consumer
  - instantiated by the Legacy App Handler implementation once per legacy application

**Boot Manager Components**

- **NSMLifecycleControlProxy**
  - generated from the NSM D-Bus interface specification
  - used by `LUCStarter` to ask for whether the LUC should be restored or not
- **SystemdManagerProxy**
  - generated from the `org.freedesktop.systemd1.Manager` D-Bus interface specification
  - used by `BootManagerService` and `TargetStartupMonitor` to talk to the `org.freedesktop.systemd1.Manager` service
- **BootManager**
  - generated from the `org.genivi.BootManager.BootManager1` D-Bus interface specification
  - used as a skeleton by `BootManagerService` to implement the `org.genivi.BootManager.BootManager1` communication
- **LUCStarter**
  - takes an `NSMLifecycleControlProxy` and a `BootManagerService`
  - asks the `NSMLifecycleControlProxy` about whether or not to restore the LUC
  - if the LUC is to be restored, uses the internal `BootManagerService` API to start the LUC units
  - has a build-time-configurable order in which the different LUC type groups are started (e.g. first foreground, then background etc.)
- **LUCHandlerService**
  - takes a `GDBusConnection` and creates a `LUCHandler`
  - also creates a `GSettings` object to store the last user context
  - creates a mutual binding between the "last-user-context" property of the `LUCHandler` and the "/last-user-context" `GSettings` key
  - connects to the "handle-register" and "handle-deregister" signals of the `LUCHandler` to implement the server-side of the D-Bus methods
  - updates the "last-user-context" property of the `LUCHandler` whenever a change is requested in the "handle-register" and "handle-deregister" callbacks

**Common components used:**
- one `ShutdownConsumer` to handle shutdown requests from the NSM
- one `NSMConsumerProxy` to register the `ShutdownConsumer` with the NSM
- one `WatchdogClient` to register with systemd's watchdog mechanism

**Legacy App Handler Components**

- **LAHandler**
  - generated from the `org.genivi.BootManager.LegacyAppHandler1` D-Bus interface specification
  - used as a skeleton in `LAHandlerService` to implement the `org.genivi.BootManager.LegacyAppHandler1` D-Bus communication
- **LAHandlerService**
  - takes a `GDBusConnection` and creates a `LAHandler`
  - connects to the "handle-register" and "handle-deregister" signals of the `LAHandler` to implement the registration and deregistration of `ShutdownConsumer` objects for legacy apps
  - implements a "handle-shutdown" signal handler for every `ShutdownConsumer` create for the legacy apps

**Common components used:**
- multiple `ShutdownConsumer` to handle shutdown requests from the NSM, one for the service itself, multiple ones for the legacy apps
- one `NSMConsumerProxy` to register the `ShutdownConsumer` of the service and the `ShutdownConsumer` objects of legacy apps with the NSM
- one `WatchdogClient` to register with systemd's watchdog mechanism

---

SysInfraEGLifecycleExecPrjctBootManagerMinutes20120611

Boot Manager Telco 2012-06-11 (15:00-15:30 CET)

- **Participants**
- **Minutes**
Participants

- Torsten Hildebrand (GENIVI, Continental)
- Alex Brankov, Marc Dunford, Jannis Pohlmann (Codethink)

Minutes

Status Report

-Finished assessment of the requirements
-Finished design documentation including functional scope and architecture overview
-Talked to the BIT team to understand the integration process
-We've signed the GENIVI CLA, requested a repository
-We've also created JIRA items for the systemd and Linux versions we need
-We've clarified that the GLib dependency is ok for now
-Now working on technical specifications (control flow, sequence diagrams, D-Bus interfaces)
-Have also started working on the implementation of the LUC handler (internal repository for now)

Visibility of the Progress

Torsten: We should start using JIRA items to organize the work and make it more visible to others in GENIVI. This is important for others to be able to add their comments.

Torsten: We should go through a public review process before we start with the actual implementation. So after the technical specifications and everything are finished, we should post the plans on the genivi-dev mailing list so that others can comment.

ACTION: Torsten to look into creating a boot manager component in the GENIVI ProjectTracker project in JIRA.

Weekly Telco

- Decision: Weekly telco to be held on Mondays, 15:00-16:00 CET using Continental's Webex.
- Codethink to send their agenda to Torsten before meetings.

SysInfraEGLifecycleExecPrjctBootManagerMinutes20120618

Boot Manager Telco 2012-06-18 (15:00-16:15 CET)

Participants

- Gunnar Andersson (EG-SI Lead, Volvo)
- Torsten Hildebrand (Lifecycle Lead, Continental)
- David Yates (EG-Auto Arch, Continental)
- Jannis Pohlmann (Codethink)

Minutes

Status Report

Design document is finished (static and dynamic views), ready for review.

| David & Torsten: Review Codethink Bootmanager design documentation |

LUC implementation done.
Bootmanager dbus interface implementation is ongoing (glib-binding, XML format)

Status of the git.genivi.org repository

Process ongoing, but started.

Shipping of NSM interfaces

Conti want to provide the NSM implementation with a evaluation license to Codethink this week (still not agreed inside Continental).

| Codethink: Should also check if they can do a NDA with Continental, that the code is provided under this NDA |

Final license of NSM still not decided in Continental.
Risk: This can have impact on the milestones of the Bootmanager.
Especially to start the testing outside Codethink is quite late.

**Review of interfaces by genivi-dev/eg-si/SAT**

- **Gunnar**: Review this week within EG-SI (Interfaces)
- **Gunnar**: Review next week within EG-SI (Design)

**Status of systemd 183 / linux 2.6.39 in Compliance 3.0**

- **Codethink**: Create JIRA item for systemd 183 for Compliance 3.0

JIRA item is created, no further activities needed anymore, handled by Gianpaolo Macario.
Closed for Codethink.

**Other open actions/items:**

- **Gunnar**: Creating a boot manager component in the GENIVI ProjectTracker project in JIRA...

Done.

- **Codethink & David**: Interface in UML model and implementation need to be synchronized

---

**Telco Information**

Weekly telco to be held on Mondays, 15:00-16:00 CET using Continental's Webex. Codethink to send their agenda to Torsten before meetings.
If someone else what to join this telco, just send a mail to Torsten

---

**SysInfraEGLifecycleExecPrjctBootManagerMinutes20120625**

**Boot Manager Telco 2012-06-27 (11:00-11:45 CET)**

**Participants**

- Gunnar Andersson (EG-SI Lead, Volvo)
- Torsten Hildebrand (Lifecycle Lead, Continental)
- Jannis Pohlmann (Codethink)
- Alex Brankov (Codethink)
- Marc Dunford (Codethink)

**Minutes**

**Status report**

- Created JIRA items for milestones and task breakdown
- Implemented LUC start-up in the boot manager service
- Implemented all methods of the org.genivi.BootManager1 interface
- Pushed the resulting code to the new git.genivi.org repository
- Started the interface review during the EG SI telco last Thursday
- Working on target start-up monitoring and node application mode activation
- Working on the implementation of the legacy application handler

**Phase 1 implementation milestone today**

- We think we have more implemented than we agreed to deliver (which was LUC handler + LUC start-up)
- Missing: registration of shutdown consumers with the NSM
- Missing: check with NSM whether LUC is to be restored or not

**NSM strategy**

- We (Codethink) want to deliver all our work packages
- Therefore we need to find a way to prove to GENIVI that our implementation works
- Proposal: develop and ship an NSM dummy (implementing the bits we need) that can be dropped as soon as the NSM is available to all of GENIVI
- Needs GENIVI to agree that this would be an acceptable way to deliver in case the real NSM doesn't make it into Excalibur

**Additional Information**
Codethink: Additional founding needed, because of more GENIVI overheader as expected. Review takes more time. Additional effort because missing NSM.

Other open actions/items:

| Codethink & David: Interface in UML model and implementation need to be synchronized |
| Torsten: NSM will be sent under NDA to Codethink. |
| Codethink: Provide a presentation of the Bootmanager design for the next EG-SI meeting. |

Telco Information

Weekly telco to be held on Mondays, 15:00-16:00 CET using Continental's Webex. Codethink to send their agenda to Torsten before meetings. If someone else what to join this telco, just send a mail to Torsten

SysInfraEGLifecycleExecPrjctBootManagerMinutes20120702

0. Introduce Ben to Torsten and Gunnar
1. Review action items from last meeting:
   • Jannis to mark the JIRA items for this milestone as completed.
   • Send email to discuss ways to have fewer and shorter meetings. Who is doing this in Codethink?
   • Jannis to make a presentation of the design and present it to the meeting tomorrow.
   • Jannis to ensure that the Action items from the previous meeting are reported in the next meeting.
2. Review JIRA tracking items
3. Report on progress/status
4. Discuss issues
   • Any progress on the NSM NDA?

SysInfraEGLifecycleExecPrjctBootManagerMinutes20120709

Boot Manager Telco 2012-07-09 (14:00-14:45 CEST)

Participants

- Gunnar Andersson (EG-SI Lead, Volvo)
- Torsten Hildebrand (Lifecycle Lead, Continental)
- Ben Brewer (Codethink)
- Alex Brankov (Codethink)

Minutes

Status report

- Updated interface changes with David Yates to be discussed tomorrow at the SAT.
- Began implementing the changes in our code base internally.
- Mostly done with our dummy NSM model, still need to know about access to code and the NDA.

NSM strategy

- We should still get access to Continental's code under NDA if necessary.
- Still need information on ErrorCodes/Shutdown interface.

Other open actions/items:

Telco Information

Weekly telco to be held on Mondays, 15:00-16:00 CET using Continental's Webex. Codethink to send their agenda to Torsten before meetings. If someone else what to join this telco, just send a mail to Torsten

SysInfraEGLifecycleExecPrjctBootManagerMinutes20120716
Boot Manager Telco 2012-07-16 (15:00-16:45 CET)

Participants

- Gunnar Andersson (EG-SI Lead, Volvo)
- Torsten Hildebrand (Lifecycle Lead, Continental)
- Ben Brewer (Codethink)
- Alex Brankov (Codethink)
- Marc Dunford (Codethink)

Minutes

Status report

- NSM dummy implementation mostly complete, paused due to Gothenburg changes.

Reviewed with Torsten:

- Discuss the changes to the project requirements from Gothenburg.
  - CT128-A002: Not required.
  - CT128-A004: Not required.
  - CT128-F004: Not required.
  - CT128-F006: Not required.
  - CT128-F014: Added.

To discuss with Gunnar:

- More work than expected caused by interface changes at Gothenburg F2F.
- Some work wasted on runtime application handling wrapper which is now removed.

Issues

- Discuss milestone slip.
  - Torsten agrees it's likely the F2F added work, we should summarize any extra work and send to him to be reviewed before bringing it up with Gunnar or the BiT group.
- No action items for a few weeks.
- Updated JIRA items related to systemd and kernel version.
- Renaming the boot manager has potential to add work, more in terms of documentation than in the code though.
  - David Yates (Continental) will announce any changes on Wednesday (18th).
- Agree updates to the project requirements.
  - Agreed.
- Still need NSM code from conti, when?
  - Torsten will chase this up.

Other open actions/items:

| Torsten: NSM will be sent under NDA to Codethink. |
| GENIVI/Codethink: Naming discussion still open. |
| Torsten: Discuss the scope/budget re-arrangements for the Gothenburg F2F. |

Telco Information

Weekly telco to be held on Mondays, 15:00-16:00 CET using Continental's Webex.
Codethink to send their agenda to Torsten before meetings.
If someone else what to join this telco, just send a mail to Torsten

SysInfraEGLifecycleExecPrjctBootManagerMinutes20120813

Node Startup Controller Telco 2012-08-13 (15:00-: CET)

Participants

- Gunnar Andersson (EG-SI Lead, Volvo)
- Torsten Hildebrand (Lifecycle Lead, Continental)
- Alex Brankov (Codethink)
- Jannis Pohlmann (Codethink)
Minutes

Review action items from previous meeting

- Torsten: NSM will be sent under NDA to Codethink.

Resolved, Codethink and Conti signed an NDA and the NSM code was sent to Codethink.

- GENIVI/Codethink: Naming discussion still open.

Resolved by renaming the component to Node Startup Controller.

- Torsten: Discuss the scope/budget re-arrangements for the Gothenburg F2F.

Mark raised the issue of extra work that was necessary and requested that we take a look at the budget used so far and compare it against the actual work that was done.

Gunnar mentioned that he would have preferred to reduce scope to reduce the amount of work towards the end of the project. He would like to take a closer look at what happened between the milestones and what the extra work was.

We discussed a few examples of where functionality that had already been implemented was dropped.

Action Item: Codethink to summarise extra work and budget used so far.

Status report

- August 1st: Delivered milestone 3
- Testing
- Various quality improvements
- Work on the reference manual
- Documentation for manual testing
- August 10th: Submitted Node Startup Controller 1.0.0 for inclusion in E-0.2
- Currently talking to the BIT to make the integration happen

Next steps

- Rename the wiki pages? Update their documentation?

SysInfraEGLifecycleExecPrjctBootManagerRequirements

Consolidated Requirements Specification for the Node Startup Controller

1. General Requirements

The status of all requirements below is tracked using the status column. Possible status values are:

- satisfied: a requirement has been implemented (either in code or in process) and approved.
- implemented: a requirement has been implemented (either in code or in process) but this has not been approved yet.
- -: a requirement has been identified but not implemented yet.

<table>
<thead>
<tr>
<th>Status</th>
<th>Codethink ID</th>
<th>GENIVI ID</th>
<th>Requirement</th>
<th>Added</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfied</td>
<td>CT128-G001</td>
<td>GENIVI-128</td>
<td>License must be MPL 2.0</td>
<td>by Gunnar Andersson on 2012-05-31</td>
<td>GPL2 might be acceptable as well but we’ll assume MPL</td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-G002</td>
<td>GENIVI-128</td>
<td>The source code repository must be hosted on git.genivi.org</td>
<td>by Philippe Robin on 2012-05-31</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-G003</td>
<td>GENIVI-128</td>
<td>GENIVI's JIRA instance must be used for issue tracking and organization of action items</td>
<td>by Philippe Robin on 2012-05-31</td>
<td></td>
</tr>
</tbody>
</table>
satisfied CT128-G004 For public communication, the mailing lists genivi-dev@genivi.org and eg-si@genivi.org must be used by Philippe Robin on 2012-05-31

satisfied CT128-G005 Weekly conference calls should be set up to report on the progress by Philippe Robin, Gunnar Andersson on 2012-05-31 Torsten Hildebrand and perhaps Gunnar will be participating from the GENIVI side

satisfied CT128-G006 Minutes of the weekly conference calls should be uploaded to the GENIVI wiki by Gunnar Andersson on 2012-05-31 Gunnar wants this for keeping track of the progress; Philippe Robin requested it as well

satisfied CT128-G007 The design documentation and technical specifications should be uploaded to the execution project page in the GENIVI wiki by Torsten Hildebrand on 2012-06-01

satisfied CT128-G008 Codethink must sign the GENIVI Contributor License Agreement and officially announce which developers are authorized to contribute code under this agreement by Jannis Pohlmann on 2012-06-01 This is required for contributions in general but also a prerequisite to having a git repo set up on git.genivi.org

2. Interface/API Requirements

<table>
<thead>
<tr>
<th>Status</th>
<th>Codethink ID</th>
<th>GENIVI ID</th>
<th>Requirement</th>
<th>Added</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfied</td>
<td>CT128-A001</td>
<td>SW-LICY-003</td>
<td>The definition and structure of Lifecycle APIs must be hardware independent</td>
<td>by Mark Hatle (Wind River)</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-A003</td>
<td>SW-LICY-066</td>
<td>An API must be provided for applications to register themselves in the LUC</td>
<td>by David Yates (Continental)</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-A005</td>
<td></td>
<td>The interfaces must be specified using the Franca IDL</td>
<td>by Gianpaolo Macario on 2012-06-21</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-A006</td>
<td>SW-LICY-022</td>
<td>The Boot Manager must keep track about start up of services/application</td>
<td>by David Yates, Gunnar Andersson on 2012-06-14</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-A007</td>
<td>SW-LICY-042</td>
<td>The Boot Manager must provide a mechanism to alter the start order of applications based on Last User Mode persistent data</td>
<td>by Christian Muck (BMW)</td>
<td></td>
</tr>
</tbody>
</table>

3. Functional Requirements

<table>
<thead>
<tr>
<th>Status</th>
<th>Codethink ID</th>
<th>GENIVI ID</th>
<th>Requirement</th>
<th>Added</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfied</td>
<td>CT128-F001</td>
<td>SW-LICY-009</td>
<td>All Lifecycle components must provide monitoring capabilities with DLT for the Test Framework</td>
<td>by Mark Hatle (Wind River)</td>
<td>This is about setting the state in the NSM. For this we rely on systemd to provide a signal for when a certain target (e.g. lazy.target) has finished. Lennart has stated that the JobRemoved() signal can be used to keep track of target startups. It is available from systemd &gt;= 183 on.</td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-F002</td>
<td>SW-LICY-023</td>
<td>The Boot Manager must keep track about start up of services/application</td>
<td>by Mark Hatle (Wind River)</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-F003</td>
<td>SW-LICY-023</td>
<td>The Boot Manager must provide a configurable dynamic shutdown procedure for legacy and 3rd party applications in the system</td>
<td>by Christian Muck (BMW)</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-F005</td>
<td>SW-LICY-042</td>
<td>The Boot Manager must provide a configurable dynamic shutdown procedure for legacy and 3rd party applications in the system</td>
<td>by Christian Muck (BMW)</td>
<td>For start-up there needs to be a way to ask the boot manager to register itself as a shutdown consumer multiple times, through unit files that execute e.g. a binary that calls a D-Bus method on the already running instance and passes a shutdown-N.target to it. During shutdown the boot manager would be called to shut down multiple times. It would first shut shutdown-N.target, then shutdown-N-1.target and so on. Better idea: Add an ExecStartPost field to unit files of legacy applications that tells the boot manager to register a new shutdown consumer for this particular legacy app.</td>
</tr>
</tbody>
</table>
The Lifecycle Management must support different levels of functionality dependent on active node sessions.

NOTE: The active node sessions agreed in the requirement actually refers to the “Node Application Mode”, i.e. Transport, SWL etc.

The boot manager must provide a “isolate this target” method for different application modes. It also must ask the NSM for whether or not it is supposed to restore the LUC. It does not need to know about the modes itself. Note: With regards to the isolate method see the note above.

The LUC handler must support audible, background and foreground applications.

Multiple applications per LUC category (audible, background, foreground) must be supported.

The order in which audible, foreground and background groups are started should be build-time configurable.

The method for registering applications with the LUC should allow for a multiple applications to be registered at once.

The boot manager must register with systemd’s watchdog mechanism.

The boot manager is responsible for setting the system state in the NSM when targets such as lazy.target have been started.

All of the BootManager functionality must reside in the same process.

<table>
<thead>
<tr>
<th>Status</th>
<th>Codethink ID</th>
<th>GENIVI ID</th>
<th>Requirement</th>
<th>Added</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfied</td>
<td>CT128-T001</td>
<td></td>
<td>Developers are expected to run static quality checks of their code to detect leaks etc.</td>
<td>by Torsten Hildebrand on 2012-05-29</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-T002</td>
<td></td>
<td>Test cases must follow the use cases in the UML model</td>
<td>by Torsten Hildebrand on 2012-05-29</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-T003</td>
<td></td>
<td>Test cases should be delivered as example setups (each with unit files and LUC data) along with documentation on how to install and test the setups manually</td>
<td>by Jannis Pohlmann on 2012-06-01</td>
<td>No automated functional tests are provided because testing functionality like this across reboots is very hard and out of scope for this small project, mentioned on 12/07/2012 telco as part of the completion criteria.</td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-T004</td>
<td></td>
<td>Two different example setups for LUC startup, each with at least one application must be provided</td>
<td>by Jannis Pohlmann on 2012-06-02</td>
<td>This is to demonstrate that the boot manager restores the LUC correctly</td>
</tr>
</tbody>
</table>
### 5. Integration Requirements

<table>
<thead>
<tr>
<th>Status</th>
<th>Codethink ID</th>
<th>GENIVI ID</th>
<th>Requirement</th>
<th>Added</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfied</td>
<td>CT128-I001</td>
<td></td>
<td>The API and ABI of the implementation must be frozen on 2012-07-30 (Compliance Freeze)</td>
<td>by Gunnar Andersson on 2012-05-11</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I002</td>
<td></td>
<td>The implementation must be feature-complete on 2012-07-30 (Compliance Freeze)</td>
<td>by Gunnar Andersson on 2012-05-11</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I003</td>
<td></td>
<td>The boot manager must be successfully integrated into the Excalibur release on 2012-08-27 (Baseline Freeze)</td>
<td>by Gunnar Andersson on 2012-05-11</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I004</td>
<td></td>
<td>The boot manager must build against the GENIVI baseline</td>
<td>by Jeremiah Foster on 2012-06-01</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I005</td>
<td></td>
<td>The boot manager must to be provided to the BIT team as an RPM or DEB package</td>
<td>by Jeremiah Foster on 2012-06-01</td>
<td>This is to make developers responsible for detecting and fixing any build and missing dependency issues</td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I006</td>
<td></td>
<td>The boot manager must be working when installed in a GENIVI baseline system (e.g. the Yocto based one for QEMU)</td>
<td>by Jeremiah Foster on 2012-06-01</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I007</td>
<td></td>
<td>An early release of the boot manager in RPM/DEB form for the E-0.1 release by 2012-07-13 is a nice to have</td>
<td>by Jeremiah Foster on 2012-06-01</td>
<td></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I008</td>
<td></td>
<td>The boot manager must adhere to the semantic versioning specification</td>
<td>by Gunnar Andersson on 2012-05-11</td>
<td>For the specification, see <a href="http://semver.org/">http://semver.org/</a></td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I009</td>
<td></td>
<td>The boot manager must pass the code scanning / license compliance check performed by the License Review Team (LRT)</td>
<td>by Jannis Pohlmann on 2012-06-02</td>
<td>For more information, see Code Scanning</td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I010</td>
<td></td>
<td>The boot manager must pass the technical evaluation coordinated by the SAT/EG-SI</td>
<td>by Jannis Pohlmann on 2012-06-02</td>
<td>For more information, see Technical Evaluation</td>
</tr>
<tr>
<td>satisfied</td>
<td>CT128-I011</td>
<td></td>
<td>The boot manager must pass the quality assessment coordinated by the SAT/EG-SI</td>
<td>by Jannis Pohlmann on 2012-06-02</td>
<td>For more information, see Quality Assessment</td>
</tr>
</tbody>
</table>

### 6. Not Required

<table>
<thead>
<tr>
<th>Codethink ID</th>
<th>GENIVI ID</th>
<th>Requirement</th>
<th>Removed</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT128-R001</td>
<td>SW-LICY-05</td>
<td>The Boot Manager must ensure that critical applications are prioritized during system start-up to ensure that they are available as early as possible within the system start-up sequence</td>
<td>by David Yates, Torsten Hildebrand (Continental) on 2012-05-29</td>
<td>This is mandatory target and during the F2F meeting in Wetzlar we decided that it belongs to Boot Management but not to the Boot Manager.</td>
</tr>
</tbody>
</table>
SysInfraEGLifecycleExecPrjctBootManagerTestScenarios

Boot Manager Test Scenarios

- 1. LUC Handler
  - Registration of individual apps
  - Deregistration of individual apps
  - Registration of multiple apps
  - Deregistration of multiple apps

1. LUC Handler

For the LUC Handler implementation, we provide a set of tests that can be performed automatically once the `luc-handler` daemon has been installed to the system.

After the typical

```
./configure && make && make install
```

sequence to build and install the boot manager (and therewith the LUC Handler), the following command can be used to run the LUC Handler test suite:

```
make -C tests/luc-handler check
```

This will clear the persistent LUC data and will perform the following tests:

Registration of individual apps

1. Register a single background app
   - Verify that the background list was created and that the app was registered
2. Register another background app
   - Verify that both apps are now registered
3. Register the first background app again
   - Verify that the two apps are still registered and that the first app is only registered once
4. Register a foreground app
   Verify that the two background apps are still registered, that the foreground list was created and that the foreground app is now registered as well

Deregistration of individual apps

1. Deregister the second background app
   Verify that only the first background app is now registered for background and that the foreground app is still registered
2. Deregister the first background app
   Verify that the background list has disappeared and that the foreground app is still registered

Registration of multiple apps

1. Register two background apps at once
   Verify that the background list was created, that both apps have been registered for background and that the foreground app is still registered as well
2. Register two more background apps
   Verify that all four background apps are now registered and that they are stored in the order they were registered
3. Register two already registered background apps in a different order
   Verify that these apps are moved to the end of the background list in the order they were registered in now
4. Register a background and an audible app, so a mix of registrations for an existing and a new LUC type
   Verify that the audible list has been created, that the new app has been added to it and that the background app has also been registered

Deregistration of multiple apps

1. Deregister all registered background apps
   Verify that only the audible and foreground apps remain in the LUC
2. Deregister all apps from the foreground and audible types, so multiple types at once
   Verify that the LUC is now empty again

SysInfraEGLifecycleExecPrjctBootManagerTrackingInJIRA

Boot Manager Execution Project Tracking in JIRA

We are using the BootManager label to link JIRA tasks/items to the execution project. The following tasks/items are being tracked at the moment:

### Milestones and Sub-tasks of the Boot Manager Execution Project (4 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Plannedstart</th>
<th>Due</th>
<th>Assignee</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>GT-2060</td>
<td>Milestone: Deliver final design and project plan to GENIVI for assessment and approval</td>
<td>04-Jun/12</td>
<td>Jun 04, 2012</td>
<td>Jannis Pohlmann</td>
<td>Closed</td>
</tr>
<tr>
<td>Error</td>
<td>GT-2065</td>
<td>Milestone: Deliver implementation of phase 1 functionality to GENIVI for approval</td>
<td>26-Jun/12</td>
<td>Jun 26, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td>Error</td>
<td>GT-2068</td>
<td>Milestone: Deliver a feature-complete implementation of the Boot Manager including phase 2 functionality to GENIVI for approval</td>
<td>24-Jul/12</td>
<td>Jul 24, 2012</td>
<td>Jannis Pohlmann</td>
<td>Resolved</td>
</tr>
<tr>
<td>Error</td>
<td>GT-2073</td>
<td>Milestone: Package integrated into GENIVI release and handover of final documentation</td>
<td>21-Aug/12</td>
<td>Aug 21, 2012</td>
<td>Jannis Pohlmann</td>
<td>Closed</td>
</tr>
</tbody>
</table>

SysInfraEGLifecycleExecProjectBootManagerLegacyApps

Legacy Application Start/Shutdown

Legacy applications are applications that provide a systemd unit file but are unaware of any GENIVI components, particularly the NSM's shutdown consumer concept. The boot manager therefore provides a helper binary and D-Bus method to register shutdown consumers for legacy applications. One shutdown consumer is created and registered with the NSM for each legacy application. During shutdown, the NSM will call them to shut down in reverse order. Each shutdown consumer will then call the boot manager to stop the corresponding legacy app unit.
Interaction with systemd

The boot manager interacts with systemd mostly through the `org.freedesktop.systemd1.Manager` and `org.freedesktop.systemd1.Job` interfaces. An `org.genivi.BootManager1.Start("app1.unit")` method call, for instance, would result in the boot manager calling `org.freedesktop.systemd1.Manager.StartUnit("app1.unit", "fail")`. From this call, systemd would return a job object, e.g., `/org/freedesktop/systemd1/job/123`, that implements the `/org/freedesktop/systemd1/job/123` interface and can be used to link what is going on in systemd with the `Start()` call. The `org.freedesktop.systemd1.Manager.JobRemoved` signal can be used to wait for the result of the job.

All interaction with systemd and boot manager clients is performed asynchronously, meaning that

- the boot manager is expected to use asynchronous D-Bus calls for the communication with systemd,
- the boot manager is not allowed to block incoming method calls while it is waiting for a systemd job to finish,
- the boot manager will delay responses to incoming method calls until the corresponding systemd job has finished; clients are expected to be aware of this and make asynchronous calls to the boot manager.

LUC and Dynamic Start-up Management

The LUC start-up is a good example of how the interaction with systemd works. Starting the units results in systemd jobs being created and `JobRemoved` signals being emitted by systemd. Due to the asynchronous nature of the interaction, there is no guarantee for jobs to be finished in the order they were started:

Runtime Application Management

All of the unit start/stop/kill/etc. methods the `org.genivi.BootManager1` interface provides result in similar interaction with systemd as is explained above for the LUC start-up. All methods result in systemd jobs being executed and their results being reported back to the boot manager. The only difference is the `org.genivi.BootManager1.List` method, which does not spawn a systemd job.
Target Start-up Monitoring

For monitoring targets starting up, the boot manager subscribes to systemd using the
org.freedesktop.systemd1.Manager.Subscribe method. Afterwards, it monitors the
org.freedesktop.systemd1.Manager.JobRemoved signal. Whenever the job for an interesting target (e.g. lazy.target) is removed, it
checks the type of the job and potentially calls the NSM to apply a new system state.

SysInfraEGMinutes20120529

Minutes of kickoff meeting Execution project "Boot Manager"

F2F in Wetzlar 29.05.2012

Participants:

- Marc Dunford (Codethink)
- Jannis Pohlmann (Codethink)
- Alex Brankov (Codethink)
- David Yates (Continental)
- Torsten Hildebrand (Continental)

Agenda:

9:00 Introduction

9:30 Idea & Concept presentation incl. answering upcoming questions

  AI: David -> NSM dbus interface to be sent to Codethink
  AI: David -> Some responsibility rewording inside the overview slide of boot management and resource management

11:00 Requirement walk-through

  AI: David -> SW-LICY-025 removed
  AI: Jannis -> SW-LICY-042 design need to be done and reviewed
  Draft design has been discussed.

Latest feedback from Lennart:

  Jannis: We also discussed that the boot manager is responsible to set the
  system state in the NSM e.g. when lazy.target has been started
  (LAZY_RUNNING). We were not sure whether systemd already provided a
  generic signal for when a unit has been started.

  I discussed this with Lennart yesterday and, starting from systemd
  version 183, there is a signal called "JobRemoved" that the boot
  manager can connect to. "JobRemoved" is emitted whenever e.g. a unit
  start job has finished successfully or unsuccessfully. The signal
  reports the unit name and the result type (e.g. success). We can use
  this to filter out the targets we are interested in and so on.

  When discussing SW-LICY-057 we discussed systemd's isolate feature.
  We were not sure whether this was part of systemd's D-Bus API.

  I talked to Lennart and he told me that it is available as a special
  mode to be passed to StartUnit(), starting from version 183.

11:30 Design goal in workpackage description

  Decision: NSM take responsibility about node application mode evaluation
  and provide an interface with the information about execute LUC (focus target) or not.
  AI: David -> Provide new extension of NSM interface

12:00 Walk-Through vehicle level requirement assigned to Boot Management

  SW-LICY-066 SW-LICY-072 were not part of the SoW, but took into account inside the offer.

12:30 Lunch

13:30 Questions and answers

  1. user-id
     How to start application with an user-id? systemd unit file supports some features

        AI: David -> check if it is satisfies our needs
2. **glib**
   glib dependency acceptable?
   What are the effort to have no dependency.
   
   | AI: Jannis -> Do a design proposal to review |

3. **license**
   Source code license?
   
   | Continental: Recommend GPL2.0, but if Mozilla public license is required it is also fine.
   | Codethink: Recommend GPL2.0, but if Mozilla public license is required it is also fine. |

4. **systemd version**
   Version 27 plus patch for the target reached signal
   or lastest version which supports this feature
   
   | The points I added to the requirements analysis section above provide strong reasons to require systemd 183 as part of GENIVI Compliance 3.0. Lennart told me that it does work with Linux >= 2.6.38, so we should try to push for that in Compliance 3.0 as well. |
   | Torsten, do you see any potential problems of hardware being incompatible with this version of the Linux kernel? |
   | Torsten: No, we do not see an issue with this kernel version |

5. **Unit/Component testing**
   Static code check should be done by the developer anyway
   Component testing need to follow the defined use-cases in EA model.
   (Note one test case can cover multiple use cases)
   Full feature state need to be demonstrated with test cases.
   
   | Jannis: Only very basic testing ("Test application (inc. systemd unit files) that can be started via the Boot Manager") was part of the SOW requirements when we submitted our proposal. We’re fine with providing example collections of systemd unit files for the different use cases along with documentation on how to run manual tests. |

6. **Organisation**
   Where to host the source code project?
   If nothing is decided by GENIVI, source force will be used.
   Where to host the documentation, project status, release notes, bugreports,...?
   If nothing is decided by GENIVI, GENIVI wiki and JIRA will be used.
   Is there any design document template? David will ask the SAT team.
   If noting is provided the project will use the GENIVI wiki.
   
   | AI: Torsten-> Create an entry page on the wiki for the execution project so that we can create our requirements specification and design documents there. |
   | Jannis: Torsten, this is important to meet the first milestone. Can we get this set up ASAP? Torsten: Will be done this. |

7. **Integration**
   What are the preconditions for a component integration with the BIT team?
   Further discussion is needed on this point.
   
   | AI: Jannis-> Ask the BIT what the pre-conditions and process for integration into Excalibur are. Keep Torsten and Gunnar in CC. |

1. **Status meeting / call**
   Regularly call every week one hour.
   All AI will be listed on the project WIKI page.
   The will be a mailing list for the project
   Project status report is part of this regularly meeting.
   
   | 15:30 Walk-through of the architecture model |
   | Explaination where everything is |
   | AI: David -> Help Jannis to get the right version of the architecture model |
   
16:30 Wrap up

   | AI: Let Torsten, David and Gunnar know about Codethink's internal genivi-boot-manager@codethink.co.uk mail alias. |
   | You now know, so we’ll consider it done. 😊 |
   | AI: Codethink-> to send Torsten, Gunnar an invitation to kanban. Should be done, please let us know if there are any problems? |
SysInfraEGLifecycleExecPrjctNodeResMgmt

Execution Project Node Resource Management

1. Team

GENIVI representatives:

- Gunnar Andersson (EG-SI Lead)
- David Yates (Topic Lead Lifecycle)

2. Statement of Work

This execution project aims at transforming the logical Node Resource Management into the specific components Node Health Monitor and Node Resource Manager.

This wiki page provides an overview of all information generated as part of the project. This includes the project schedule, requirements, an overview of the architecture, pointers to the implementation, documents and technical specifications as well as progress reports in the form of telco minutes.

3. Project Schedule

- Node Health Monitor released as Abstract P2 in Horizon
- Node Health Monitor changed to Specific P1 in Intrepid
- Node Resource Manager released as Abstract P2 in Intrepid

4. Weekly Status Reports

Weekly status reports can be found within the EG-SI Weekly Meetings

5. Requirements

- Requirements Specification
- Excalibur Dependencies

6. Architecture Overview

- Functional Scope
- Architecture Overview
- Internal Software Architecture
- Resource Management presentations

7. Documents and Specifications

- Lifecycle presentations

8. Implementation
The source code repository for the Node Health Monitor is available here:

- Node Health Monitor Code

Working with GENIVI Git repositories is described this wiki page.

9. Baseline Integration Status

10. Telcos

11. Overview of Wiki Pages

The Lifecycle domain is described in a number of Wiki pages and you can navigate through the domain pages using the arrow links on the bottom of the page.

12. Component Specification

Here is the latest version of the Component Specification for the Node Health Monitor.

This is based on the latest version of the EA Model and the content of the Wiki pages.

13. Test Plan

The NHM is delivered with a unit test that can be started via the "make check" command. A coverage analysis with the tool “BullsEye” resulted in a function-coverage of 100 % and a branch-coverage of 84 % for NHM in version 1.3.3.

The unit test should be considered in every coming source code adaption. A static code analysis with the tool “Klocwork” and a memory leak check with “valgrind” did not show any findings for the NHM source.

SysInfraEGLifecycleExecPrjctNSM

Execution Project Node State Management

- 1. Team
- 2. Statement of Work
- 3. Project Schedule
- 4. Weekly Status Reports
- 5. Requirements
- 6. Architecture Overview
- 7. Documents and Specifications
- 8. Implementation
- 9. Baseline Integration Status
- 10. Telcos
- 11. Overview of Wiki Pages
- 12. Component Specification
- 13. Test Plan
- 14. Yocto BB recipe

1. Team

GENIVI representatives:

- Gunnar Andersson (EG-SI Lead)
- David Yates (Topic Lead Lifecycle)

2. Statement of Work

This execution project aims at transforming the logical Node State Management into the specific components Node State Manager and Node State Machine by delivering an implementation to be integrated into the GENIVI Foton release.

This wiki page provides an overview of all information generated as part of the project. This includes the project schedule, requirements, an overview of the architecture, pointers to the implementation, documents and technical specifications as well as progress reports in the form of telco minutes.

3. Project Schedule
4. Weekly Status Reports

Weekly status reports can be found within the EG-SI Weekly Meetings

5. Requirements

- Requirements Specification
- Excalibur Dependencies

6. Architecture Overview

- Functional Scope
- Architecture Overview
- Internal Software Architecture
- System shutdown concept
- NSM Data

7. Documents and Specifications

- Lifecycle presentations

8. Implementation

The source code repository for the Node State Manager is available here:

- Node State Manager Code

Working with GENIVI Git repositories is described this wiki page.

As a part of the review to promote the NodestateManager from "Abstract P2" to "Specific P1", the question arose, how the glib usage influences the startup. Therefore, measurements with strace have been made. The result indicates that shared objects are not loaded at once, but only the symbols that are accessed. The usage of large shared objects, in this case glib, does not necessarily slow the startup down.

The output of the strace measurement can be found here.

9. Baseline Integration Status

This section will summarise the integration of the NSM implementation into the GENIVI baseline.

10. Telcos

11. Overview of Wiki Pages

The Lifecycle domain is described in a number of Wiki pages and you can navigate through the domain pages using the arrow links on the bottom of the page.

12. Component Specification

Here is the latest version of the Component Specification for the Node State Manager.

This is based on the latest version of the EA Model and the content of the Wiki pages.

13. Test Plan

The attached document node_state_manager Test Plan.doc describes the test plan for the Node State Manager. This matches the code that is stored within the GIT repository.

14. Yocto BB recipe
Node State Manager Description

This is the draft proposal of the public web page text for NSM.

Node State Manager (NSM), part of the Lifecycle subsystem in GENIVI

Introduction

The node state management is the central function for information regarding the current running state of the embedded system. The Node State Manager (NSM) component provides a common implementation framework for the main state machine of the system. The NSM collates information from multiple sources and uses this to determine the current state(s).

The NSM notifies registered consumers (applications or other platform components) of relevant changes in system state. Node state information can also be requested on-demand via provided D-Bus interfaces.

The node state management also provides shutdown management, so one part of the information which is provided is the shutdown request notification to the consumers.

The node state management is the last/highest level of escalation on the node and will therefore command the reset and supply control logic. It is notified of errors and other status signals from components that are responsible for monitoring system health in different ways.

Internally, node state management is made up of a common generic component, Node State Manager (NSM), and a system-specific state machine (NSMC) that is plugged into the Node State Manager. Through this architecture there can be a standardized solution with stable interfaces towards the applications, which still allows for product-specific behavior through the definition of the specific state machine.

Full Text

The node state management is the central function for information regarding the current running state of the embedded system. The Node State Manager (NSM) component provides a common implementation framework for the main state machine of the system. The NSM collates information from multiple sources and uses this to determine the current state(s).

The NSM notifies registered consumers (applications or other platform components) of relevant changes in system state. Node state information can also be requested on-demand via provided D-Bus interfaces.

In addition to states, the NSM supports the notion of sessions which are temporary running conditions of the system. Sessions are independent and orthogonal to the node state. There are a number of predefined session types corresponding to what all IVI systems typically use. However, a key difference between sessions and states, is that any application can also dynamically add its own session type. In an automotive IVI system, an example of a typical session is an ongoing phone call. The specific state machine could be implemented such that this affects how the state machine reacts to, for example, a shutdown request while the session is active.

The NSM also provides shutdown management. The boot management functionality (primarily implemented by the Node Startup Controller) only boots the node. Shutdown is achieved by the Node State Manager signalling applications to shut down. Please note however that a key difference in the GENIVI Lifecycle handling compared to typical Linux systems is that components shall normally be designed to simply go into a quiet state. Typically this means to store persistent data and make sure file systems are in a consistent state and so on. An application is however not shut down in the normal sense, i.e. it does not free memory or unload itself unnecessarily. This is to allow for the "cancelled shutdown" feature which allows the system to quickly resume into a running state if this was requested by the user at a time before the system has reached a full shut down. If shutdown completes however, the applications must have entered a state where it is safe to perform a reset or power-off.

This will be done later. Go ahead with the text above

Under the full text I would suggest that we include the majority of the contents of

https://collab.genivi.org/wiki/display/genivi/SysInfraEGLifecycleConcept

to really understand how the Node State Manager fits in the Lifecycle Concept.

Subdomain Lifecycle Documentation

- A list of former/legacy documentations and specifications
- Lifecycle in system context
- Requirements analysis (the compliance requirements, use cases and scenarios)
- Functional analysis (based on use cases and scenarios)
- Design
- Integration
- Tests / Compliance
A list of former/legacy documentations and specifications

The following picture shows an overview of lifecycle components.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name of item</th>
<th>Purpose of item</th>
<th>Author (Name,Company)</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Platform_State_Manager-Req.pdf</td>
<td>Platform State Management Requirement Specification (ver 0.17)</td>
<td>Mark Hatle, Wind River</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>2</td>
<td>PSMPresentation.ppt</td>
<td>Top Level presentation describing the Platform State Manager (03/11/2009)</td>
<td>Mark Hatle, Wind River</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>3</td>
<td>DFS-PowerStateManagement.pdf</td>
<td>Overview of Demo (K3) version of the Program State Management (v0.17 - 29/05/2009)</td>
<td>Wind River</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>4</td>
<td>GENIVI_Platform_Lifecycle.pdf</td>
<td>GENIVI Platform Lifecycle Specification Draft Version 0.2 (19/06/2009)</td>
<td>Mark Hatle, Wind River</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>5</td>
<td>System_Resource_Management_Requirements_v1.0.doc</td>
<td>The capture of Resource Management requirements for the GENIVI system (Draft Version 1.0 - 23/06/2009)</td>
<td>Unknown</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>6</td>
<td>DFS-ControlGroupsHealthMonitor-R2_0.pdf</td>
<td>Overview of the K3 Demo for the System Health Monitor (Version 1.2 - 22/04/2009)</td>
<td>Brian Gunn, Wind River</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>7</td>
<td>System_Resource_Management-collated-091709.xls</td>
<td>System Resource Management: Requirements review</td>
<td>Various</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>8</td>
<td>BootInfrastructure</td>
<td>(Fast) Boot Infrastructure: proposed boot process, power states and requirements</td>
<td>Mark Hatle Wind River</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>9</td>
<td>FastBootMoblin</td>
<td>Fast Boot Infrastructure: Moblin optimizations</td>
<td>Gerard Maloney Intel</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>10</td>
<td>SIBootTimings</td>
<td>Fast Boot Infrastructure: requirements</td>
<td>Manfred Bathelt BMW</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>11</td>
<td>systemd - system initialization daemon</td>
<td>Systemd: system initialization and session management daemon</td>
<td>Michael Kerrisk jambit</td>
<td>To-be-checked</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>GENIVI_1.0_RC_Architecture_Implementation_Overview.7z</td>
<td>Architecture Implementation Overview: GENIVI 1.0</td>
<td>GENIVI</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>13</td>
<td>100803_GENIVIStakeholdersNeeds_1.52.pdf</td>
<td>Stakeholders Needs Document: Apollo Reference document</td>
<td>GENIVI</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>14</td>
<td>ResourceManager</td>
<td>Resource Manager: Wiki - component description, requirements, implementation</td>
<td>Mark Hatle Wind River</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>15</td>
<td>SystemHealthMonitor</td>
<td>System Health Monitor: Wiki - component description and requirements</td>
<td>Mark Hatle Wind River</td>
<td>To-be-checked</td>
</tr>
<tr>
<td>16</td>
<td>PowerStateManager</td>
<td>Power State Manager: Wiki - component description and requirements</td>
<td>Mark Hatle Wind River</td>
<td>To-be-checked</td>
</tr>
</tbody>
</table>

**Lifecycle in system context**

- System border (SysInfraEGLifecycleConceptSystemBorder)
- General concept (SysInfraEGLifecycleConcept)
  - Boot management Split (SysInfraEGLifecycleConceptBootmanagementSplit)
    - Boot Management - Startup concept (SysInfraEGLifecycleConceptBootmanagementStartup)
    - Boot Management - Shutdown concept (SysInfraEGLifecycleConceptBootmanagementShutdown)
- Multi-node concept (SysInfraEGLifecycleConceptMultiNode)
- Concept refinement (SysInfraEGLifecycleConceptRefinement)

**Requirements analysis (the compliance requirements, use cases and scenarios)**

- Requirements (SysInfraEGLifecycleRequirements)
  - Vehicle Requirements
    - GENIVI Model -> Requirements View -> System Infrastructure -> Lifecycle
  - SW Platform Requirements
    - GENIVI Model -> Logical View -> SW Platform Requirements -> System Infrastructure -> Lifecycle

$ : The Lifecycle directory refers to the subdomain Lifecycle and will itself be split it into the required subdirectories and hence will contain all requirements. Existing subdirectories at this level referring to Lifecycle will be removed (i.e. System Health Monitor etc)

- Use cases and scenarios (SysInfraEGLifecycleRAUsecases)
  - GENIVI Model -> Use Case View -> System Infrastructure -> Lifecycle

**Functional analysis (based on use cases and scenarios)**

- Black box realization of use cases (SysInfraEGLifecycleFABlackBoxUcRealization)
  - Static views
    - GENIVI Model -> Logical View -> Clusters -> Base Layer -> System Infrastructure -> Lifecycle
  - Dynamic views
    - GENIVI Model -> Logical View -> Use Case Realizations -> System Infrastructure -> Lifecycle

**Design**

- Structure definition (SysInfraEGLifecycleDesignStructDef)
  - Blocks / Components
    - GENIVI Model -> Logical View -> Logical Components
  - Activity diagrams
  - IMPLEMENTATION Model -> System Infrastructure -> Lifecycle -> Sequence Diagrams
- Public interfaces
- Design analysis (white box realization of use cases)(SysInfraEGLifecycleDesignWhiteBoxUcRealization)
Integration

- Dependencies
- Deployment

Tests / Compliance

- Test criteria
- Test cases

SysInfraEGLifecycleConcept

Subdomain Lifecycle Concept

Level 1 logical view of Lifecycle

The objectives are, which driving this first-level structure definition:

- Complexity reduction
- Clear responsibility split
- De-central organization with less dependencies
- Offering GENIVI SW platform interfaces/attributes
- Extendable with product specifics

The following figure presents the level 1 logical view of the concept for Lifecycle.

Responsibility Description

Supply management

Abstract:
The supply management will monitor supply specific signals/escalations provided/abstracted by the plug-ins organized in a tree structure and will notify registered consumers of state changes. Each plug-in is a supply commander which will try first to solve issues by its own. Only in case if it is not resolvable it will escalate the issue to the upper tree node. The root of this tree is the supply management.

It will be up to the registered consumer and the plug-ins to decide what action to take within the node (i.e. logging errors, reducing functionality in the system,...) if the supply condition is going worse.

There is one special consumer the node state management which is interesting in 3 mapped supply platform states (GOOD, POOR, BAD).

- If the condition is GOOD, no further action is needed.
- If the condition is POOR, the node state management will request a shutdown of the system/ECU, but if the system is in a state (like SW update), the request won't be satisfied.
  - (System lifetime protection)
- If the condition is BAD, the node state management will force the system/ECU to shutdown immediately. This request can not be overruled.
  - (System corruption protection)

Configuration:
The supply management will use a configurable state machine. A default state machine will be provided by GENIVI, but this can be replaced/extended at the product development.

Plug-in:
The plug-ins has the role/responsibility to abstract the product specific HW/network information about supply conditions and will already react on issues where it is able to.

Thermal management

Abstract:
The thermal management will monitor thermal specific signals/escalations provided/abstracted by the plug-ins organized in a tree structure and will notify registered consumers of thermal changes. Each plug-in is a thermal commander which will try first to solve issues by its own. Only in the case it is not resolvable will it escalate the issue to the upper tree node. The root of this tree is the thermal management.

It will be up to the registered consumer and the plug-ins to decide what action to take within the node (i.e. activating of fans in the system and/or reducing the volume,...) if the thermal condition is getting worse.

There is one special consumer the node state management which is interested in 3 mapped thermal platform states (GOOD, POOR, BAD).

- If the condition is GOOD, no further action is needed.
- If the condition is POOR, the node state management will request a shutdown of the system/ECU, but if the system is in a state (like SW update), the request won't be satisfied.
  - (System lifetime protection)
- If the condition is BAD, the node state management will force the system/ECU to shutdown immediately. This request can not be overruled.
  - (System corruption protection)

Configuration:
The thermal management will use a configurable state machine. A default state machine will be provided by GENIVI, but this can be replaced/extended at the product development.

Plug-in:
The plug-ins has the role/responsibility to abstract the product specific HW/network information about supply conditions and will already react on issues where it is able to.

[Markus Boje 2011-04-14: Supply and Thermal Management looks very similar. The difference seems to be in the plug-in tree. Can this be subsumed by a kind of OuterConditionManager?]
[Torsten Hildebrand 2011-04-27: On concept level we scoping on different topic, the manifest could be in one implementation running in two instances]
Power management

Abstract:
The power management will receive power event/messages from product specific plug-ins which are providing the connection to the node/vehicle network, also a plugin for abstraction of the wake-up logic will have to be provided by the product development. The power management will filter and map them to GENIVI predefined events/messages, which will be forwarded to the node state management. As the internal state-machine of the node state management contains product specifics, raw power events/messages can also be send to it. Currently there is no direct interface planned at the power management to access internal information by applications. All application relevant information shall be provided on the node state management.

Configuration:
Product to platform mapping information has to be provided.

Plug-in:
Plug-ins will be used to provide all requested information to the power management, developed by product development.

Node state management

Abstract:
The node state management is the central repository for information regarding the states/sessions inside the node. It collates information from multiple sources and uses this to determine the current states (there are different states existing for different purposes). This information will be delivered to registered consumers or can be requested via the provided interface. All the raw data gathered by the node state management is also be available on request to interested consumers.

The node state management also provides the shutdown management, so one part of the information which is provided is the shutdown request notification event/message to the consumers. The boot management only boots the node and has to stop the boot activity in case of a high priority shutdown request, which is initiated by the node state management. That means boot management is just a consumer of the node state management.
The node state management is the last/highest level of escalation on the node, therefore it will command the reset and supply control logic. Additionally if the system is a multi-node system the node state management has included a slave (done as a consumer) which knows the specifics of this configuration and knows about what event/message need to be transferred also to other node(s).
**Configuration:**

The node state management will use a configurable state machine. A default state machine will be provided within GENIVI, but this can be replaced/extended by the product development.

**Plug-in:**

The node state management will use a plug-in to initiate supply off/cycle events/command in the system, pending if the control logic is locally or located on another node.

**Data**

The node state management will access and control data from a number of sources. Further information on this topic can be found here.

**Boot management**

**Abstract:**

The boot management contains all the functionality required to start a GENIVI based product including low level drivers, services, automotive applications and third party add-ons. This configuration has to be easily manageable, especially the included dependencies and boot order information. The boot management shall handle boot dependencies as much as possible by itself automatically to reduce the complexity and the integration effort for the development team.

The most important goal of boot management is to start-up the node as fast as possible to a requested functionality state. To satisfy this goal parallelization of component start-up is provided to avoid unused CPU power (idle time). Additionally it will include the runtime prioritization during start-up of components based on user driven events (e.g. the last user context information is read to dictate the boot path). In some cases a cancel of the start-up activity is needed like a "BAD" thermal condition. This is one of the reasons why the boot management is a consumer of the node state management. Finally the boot management is responsible to setup resource management configurable and observable partitions.

**Configuration:**

All SW components to be started on the node must provide information regarding their internal dependencies which will be used by the boot management during the system start-up including the partition base information.

For certain components which are used for dictating the boot path additional configuration is needed.

**Plug-in:**

Not applicable

**Resource management**

**Abstract:**

Resource management contains the functionality to ensure that the node runs in a stable and defined manner. To do this it will monitor and limit different aspects of SW component behavior including system resources (like CPU load and memory) and critical run-time observation.

Resource allocation (CPU and memory) will be configurable on a component basis and will provide run time dynamic alterations in the area of resource allocation when requested by applications in predefined resource-hungry use cases.

In case of critical run-time observation failure, the following escalation strategies could be considered to repair the situation :

- Restart application
- Restart node
- Rollback of application to previous version
- Rollback of all user updates to baseline state
- Deletion of applications persistence data
- Deletion of all user persistence data

The exact escalation strategy employed will be configurable and determined by the OEM.

Over and above the described functionality, the resource management stores this situation in the error storage, creates and records a crash dump for future problem analysis.

**Configuration:**

The required configuration data is defined and provided on the component level.

**Plug-in:**

The monitoring of critical or important SW components on the node will be done via plug-ins. A plug-in will be provided to monitor one or many applications on the node along with a recovery strategy for the failing application (i.e. restart single application/restart complete system).

[Christian Muck 2011-04-20: Where does the plug-in gets the information from if an application fails? (Will the plug-in observes the]
application information from systemd, will systemd notify the plug in with the "OnFailure" attribute of a unit file or will it observe the application itself e.g. /proc entries?

[Torsten Hildebrand 2011-04-27: The error reporting will be different between the applications, because they are existing in a lot of cases, therefore a specific plug-in is needed. systemd will only report if an application is unexpected closed and may the application offers systemd based error callback routines to get more information. Additionally we will offer an GENIVI-example plug-in, which can be used for new development.]

[[torsten.hildebrand 2011-10-21] Added the interface for storing the LUC at the Boot Management]

SysInfraEGLifecycleBootMgmtSOW

GENIVI Lifecycle - Boot Management

Statement Of Work

- GENIVI Lifecycle - Boot Management
- Statement Of Work
  - Scope of Work
    - Introduction
    - Organization Of Work
    - Schedule
    - Period of performance
  - Requirements
  - Design
  - Shutdown Concept
  - Use cases
  - Key activities
  - Initial list of expected tasks
    - Solution definition and final design
    - Solution Development
    - Coordination of upstream acceptance
    - Maintenance
  - Completion Criteria
- Scope of Work

Introduction

The purpose of this statement of work is to define the implementation required to fulfill the architecture of the Boot Management package within the Lifecycle sub-domain.

Background

The system architecture for the Lifecycle sub-domain can be found in the GENIVI architecture model and a subset of the architecture is available within the GENIVI wiki.

The functional scope of Lifecycle is:
However for this statement of work we are only interested in the Boot Manager component that refines the Boot Management package:
The Boot Management contains all the functionality required to start a GENIVI based product including low level drivers, services, automotive applications and third party add-ons. This configuration has to be easily manageable, especially the included dependencies and boot order information. The boot management shall handle boot dependencies as much as possible by itself automatically to reduce the complexity and the integration effort for the development team.

The most important goal of boot management is to start-up the node as fast as possible to a requested functionality state. To satisfy this goal parallelization of component start-up is provided to avoid unused CPU power (idle time).

Additionally it will include the runtime prioritization during start-up of components based on user driven events (e.g. the last user context information is read to dictate the boot path). In some cases a cancel of the start-up activity is needed like a “BAD” thermal condition. This is one of the reasons why the boot management is a consumer of the node state management.

Finally the boot management is responsible to setup resource management configurable and observable partitions.

### Organization Of Work

<table>
<thead>
<tr>
<th>Topic</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintenance Mode</td>
<td>Upstream</td>
</tr>
<tr>
<td>License for the contributed code</td>
<td>GNU General Public License</td>
</tr>
<tr>
<td>GENIVI mentor</td>
<td>Continental Automotive</td>
</tr>
<tr>
<td>Relevant upstream projects</td>
<td>Linux systemd Boot Manager libgroup</td>
</tr>
<tr>
<td>Maintainers</td>
<td>Implementer</td>
</tr>
</tbody>
</table>

**Schedule**

The outcome of this SOW is planned to be included in Excalibur. Due to the short timescales and the inherent risks to a a GENIVI distribution synchronization of this work with Discovery is not feasible. NOTE: the planned introduction of systemd into the Discovery release is not effected by this SOW. The usage of systemd within a distribution is still possible without the additional features provided by the Boot Management. The aim is to have a functional solution by the second calendar quarter 2012 (Q2 2012) that can be integrated within the BIT.

**Period of performance**

Start: TBD
### Requirements

The SW requirements to be fulfilled by the Boot Manager can be found in the table below:

<table>
<thead>
<tr>
<th>Alias</th>
<th>Derived from</th>
<th>Area</th>
<th>Originator (Name, Company)</th>
<th>Description</th>
<th>Additional Info</th>
<th>Priority</th>
<th>State</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW-LICY-003</td>
<td>API</td>
<td>Mark Hatle, Wind River</td>
<td>Hardware independent API's</td>
<td>The definition and structure of Lifecycle APIs must be hardware independent</td>
<td></td>
<td>P1</td>
<td>Released</td>
<td>Christian Muck BMW - Okay.</td>
</tr>
<tr>
<td>SW-LICY-009</td>
<td>VH-LICY-027</td>
<td>Component</td>
<td>Mark Hatle, Wind River</td>
<td>DLT Monitoring capabilities</td>
<td>All Lifecycle components must provide monitoring capabilities with DLT for the Test Framework</td>
<td>P1</td>
<td>Released</td>
<td>Christian Muck BMW - Better understanding: Lifecycle Management provide monitoring capabilities with DLT for the Test Framework. Original Req. Lifecycle Management provide monitoring capabilities for the Test Framework.</td>
</tr>
<tr>
<td>SW-LICY-022</td>
<td>RQ 1.14.1</td>
<td>Component</td>
<td>Mark Hatle, Wind River</td>
<td>Track start up of services/application</td>
<td>The Boot Manager must keep track about start up of services/application</td>
<td>P1</td>
<td>Released</td>
<td>David Yates, Continental - Component design and naming not finalized hence reworded. Christian Muck BMW - Okay. Original Req. SHM must keep track about start up of services/application.</td>
</tr>
<tr>
<td>SW-LICY-023</td>
<td>VH-LICY-023</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>Last User Mode persistent data</td>
<td>The Boot Manager provide a mechanism to alter the start order of applications based on Last User Mode persistent data.</td>
<td>P1</td>
<td>Released</td>
<td>Munich Workshop: Reword for LI dependent sti; Original Req. The Lifecycle Management supports different start priorities applications.</td>
</tr>
<tr>
<td>SW-LICY-024</td>
<td>VH-LICY-026</td>
<td>Component</td>
<td>Start/stop applications during run-time</td>
<td>The Boot Manager must provide a mechanism for starting and stopping applications during normal run-time. The Boot Manager will use this for shutdown and cancel shutdown scenarios. Additionally this interface will be used by SWL and System Health Monitor.</td>
<td>P1</td>
<td>Released</td>
<td>Christian Muck, BMW</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
<td>------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
<td>--</td>
<td>--------</td>
<td>-------------------</td>
<td></td>
</tr>
</tbody>
</table>

| SW-LICY-025 | VH-LICY-026 | Component | Critical applications are prioritized | The Boot Manager must ensure that critical applications are prioritized during system start-up to ensure that they are available as early as possible within the system start-up sequence | P1 | Released | David Yates, Continental |

| SW-LICY-042 | VH-LICY-004 | Component | Configurable shutdown procedure | The Boot Manager must provide a configurable dynamic shutdown procedure for legacy and 3rd party applications in the system | P1 | Released | David Yates, Continental |

| SW-LICY-050 | VH-LICY-011 | Component | User input alters start-up sequence | The Boot Manager must be able to dynamically alter the start-up sequence of the Head Unit to reflect operations on user input devices i.e. If the user presses the "Navi" button during the system start-up then the Navigation SW must be given priority during the start-up sequence | P1 | Released | David Yates, Continental |
The Lifecycle Management must support different levels of functionality dependent on active node sessions.

NOTE: the active node sessions agreed in the requirement actually refers to the "Node Application Mode", i.e. Transport, SWL etc.

Design

As detailed previously this statement of work covers the implementation required for the Boot Manager. The standard flow for the Boot Manager is as follows:

- Register with the Node State Manager as a consumer for the Shutdown State
- Register with the Node State Manager as a consumer for the Node Application Mode
- Trigger the Node State Manager about the current Lifecycle State
- Evaluating the current "Node Application mode" to determine subsequent actions in the Lifecycle
  - Normal & Parking mode
    - The Boot Manager needs to evaluate the current user and read the Last User Context
    - Trigger systemd with the applications to be given priority in the current lifecycle
  - Transport, SWL & Factory Mode
    - Simply signal systemd about completion as default SW should be started in these modes

Additional responsibilities are:

- Providing an interface that can be used to trigger application start/stop within systemd
- Multiple registration with the NSM to allow for the shutdown of legacy applications

Shutdown Concept

The basic idea is the following:

- During start-up phase the applications,... will register at the Node State Management which are interested to be involved during a shut-down phase.
- The order of registration in start-up is the inverted order of the execution in shut-down phase. Currently we think that we do not need a specific specification of the shut-down behavior. There will be a protocol defined between the registered consumers and the Node State Management.
- There will be specific consumers which are responsible to set a shutdown target or a list of conflicts to systemd, which enables the de-initialization of (legacy) components via systemd.
Use cases

The Boot Manager must fulfill the use cases defined in the two following linked documents:

White Box UC
Black Box UC

Key activities

Initial list of expected tasks

The following table captures the high-level tasks to be performed as part of this SOW. They are all rather interdependent.

Solution definition and final design

<table>
<thead>
<tr>
<th>Work</th>
<th>Project</th>
<th>Name</th>
<th>Short description</th>
<th>Priority</th>
<th>Work Estimation (in days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.01</td>
<td>General</td>
<td>Documentation</td>
<td>Document project plan</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1.02</td>
<td>General</td>
<td>Documentation</td>
<td>Document design of Boot Manager</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1.03</td>
<td>General</td>
<td>Community adoption</td>
<td>Communicate with the maintainers sub-system maintainers upstream early to engage the dialog and increase the likelihood of adoption.</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1.04</td>
<td>General</td>
<td>Mitigate upstream risks</td>
<td>Incorporate upstream inputs into the final design and circle back with the contributors upstream</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1.05</td>
<td>General</td>
<td>Present SOW design</td>
<td>Once finalized, present the SOW design to all relevant GENIVI expert groups to ensure transparency and approval</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1.06</td>
<td>General</td>
<td>Sign off</td>
<td>Work within GENIVI process to ensure component sign off and integration</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

Sum of effort estimations:

Solution Development

<table>
<thead>
<tr>
<th>Work</th>
<th>Project</th>
<th>Name</th>
<th>Short description</th>
<th>Priority</th>
<th>Work Estimation (in days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.01</td>
<td>Linux</td>
<td>Implementation</td>
<td>Implement work package and deliver into GIT Hub for testing within existing systemd POC</td>
<td>1</td>
<td>80</td>
</tr>
</tbody>
</table>

Sum of effort estimations:
Coordination of upstream acceptance

<table>
<thead>
<tr>
<th>Work</th>
<th>Project</th>
<th>Name</th>
<th>Short description</th>
<th>Priority</th>
<th>Work Estimation (in days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.01</td>
<td>Linux</td>
<td>final delivery preparation</td>
<td>Prepare compliance documents required for acceptance into GENIVI Bit team</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3.02</td>
<td>Linux</td>
<td>Delivery support</td>
<td>Support the BIT team with the integration of the package</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Sum of effort estimations:

Maintenance

Supplier will take on the responsibility to maintain the solution during the acceptance period. Once upstream, supplier will coordinate maintenance with the upstream maintainers and sub-system maintainers necessary for the respective projects.

Completion Criteria

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Project</th>
<th>Short description</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.01</td>
<td>Design document</td>
<td>Document that describes the architecture of the module</td>
<td>GENIVI Copyright</td>
</tr>
<tr>
<td>4.02</td>
<td>Boot Mgr module</td>
<td>Actual Linux implementation of Boot Manager in source form</td>
<td>GPLv2</td>
</tr>
<tr>
<td>4.03</td>
<td>General</td>
<td>Test application (inc. systemd unit files) that can be started via the Boot Manager</td>
<td>GPLv2</td>
</tr>
<tr>
<td>4.04</td>
<td>General</td>
<td>Manual pages to reflect implementation</td>
<td>Misc.</td>
</tr>
<tr>
<td>4.05</td>
<td>General</td>
<td>Successful integration of the module inside a GENIVI release</td>
<td>GENIVI Compliance</td>
</tr>
</tbody>
</table>

SysInfraEGLifecycleConceptBootmanagementSplit

Subdomain Lifecycle Boot Management- Concept -Split

Introduction

The split of boot management

Why do we have that split?

It is possible to perform a shutdown using systemd, however it would stop and unload the components in its shutdown concept. This results in a large lead time to make them functional again in case of a cancel shutdown situation. An IVI system must be able to come back fast
without losing any context if a reboot can be avoided for that cancel shutdown.

Node State Management will only call registered consumers in the shutdown phase. Those consumers will drive the components into a stable state and ensure that everything has been stored which will be needed for the next start-up.

Note: The components won't be shutdown (Exceptions are existing like the flash filesystem). Therefore in addition the shutdown management concept will include/use the systemd shutdown concept as well where it makes sense and supports legacy/adapted components.

---

**SysInfraEGLifecycleConceptBootmanagementShutdown**

**Subdomain Lifecycle Bootmanagement- Shutdown Concept**

This page is under construction.

Table of content:

- Introduction
- About the concept
- Shutdown preparation in start-up phase
- Shutdown in execution
- Analysis of concept

---

**Introduction**

In automotive the aim in a shut-down phase is to trigger real needed activities only. That means a lot of standard/traditional activities for shut-down are not in the focus in an IVI-system. If you can archive the requested start-up KPI with a cold boot the IVI system will be switched off at the end of the shut-down phase.

The main activities which are need in shut-down are the following:

- log off the user
- write / specify the last user context
- unregister at the vehicle network
- bring hardware in a safe state (drives,...)
- mute audio
- switch off display(s)
- store application and system persistence data

**About the concept**

The basic idea is the following:

- During start-up phase the applications,... will register at the Node State Management which are interested to be involved during a shut-down phase.
- The order of registration in start-up is the inverted order of the execution in shut-down phase. Currently we think that we do not need a specific specification of the shut-down behavior. There will be a protocol defined between the registered consumers and the Node State Management.
- There will be specific consumers which are responsible to set a shutdown target or a list of conflicts to systemd, which enables the de-initialization of (legacy) components via systemd.

---

**Shutdown preparation in start-up phase**
During the start-up phase components that are interested in being notified about impending shutdowns will register with the Node State Manager. They must do the registration with the Node State Manager before they notify systemd that they have finished initialization as the Node State Manager will then benefit from the dependency ordering provided by systemd in the startup sequence.

### Shutdown in execution

To allow for the shutdown of Legacy/3rd Party applications that do not want to use the Node State Manager interface the Boot Manager will be able to register with the Node State Manager as a consumer for the component in question. This will be achieved by using the ExecStatePost tag in the component's unit file.

During the shutdown sequence the Node State Manager will call the registered consumers in the reverse order of registration to preserve any dependencies that may exist.

### Analysis of concept

- Shutdown activities are triggerable without unloading the components.
- Legacy components can be shut down as their traditional way.
SysInfraEGLifecycleConceptBootmanagementStartup

Subdomain Lifecycle Boot Management Start-up Concept

Table of content:

- Introduction
  - Rough systemd performance analysis
- The concept
  - Overview of the GENIVI boot phases (target clustering)
  - Focused.target
    - BootManager
    - Focused Application Types
    - Application Design Constraints
    - Example flow
  - Unfocussed.target(s)
    - Usage
  - Lazy target
    - Usage
- An example
  - Example of Subsystem Start-up Order using systemd + BootManager
- Appendix
  - Special system units

Introduction

To understand the Bootmanagement Concept based on systemd, it is necessary to understand the principles of systemd. If you already know systemd and for what unit files are useful, you can skip the introduction on SysInfraEGSystemd - otherwise please take some minutes to learn more about systemd (reading or watching video).

Each unit will have a configuration file that can be used to define critical information about the unit that systemd will use during the start-up process.

For instance here is the configuration file (.service) for crond:

```
[Unit]
Description=Command Scheduler
After=syslog.target

[Service]
EnvironmentFile=/etc/sysconfig/crond
ExecStart=/usr/sbin/crond -n $CRONDARGS

[Install]
WantedBy=multi-user.target
```

Of interest for us at this point are the After and the WantedBy keywords. These show us that the crond must be started after syslog.target and is part of the multi-user.target

One type of unit is a target. A target is a collection of sub units that will all be started when that target is activated. They allow for easier grouping of units and control of the system dependencies. Additionally they typically mirror the standard Linux run-levels.

The following diagram (Example subtarget deployment on Fedora 15) shows a typical dependency graph for the main targets.
The problem with this approach for GENIVI is that we need the graphical front-end earlier and there are components in the Mandatory section that are not needed so early in the start-up. Therefore we need to change the target clustering.

Rough systemd performance analysis

Using Fedora 15 and a Virtual Machine as a baseline we analyzed the system start-up to evaluate how and what systemd was starting.

- Systemd trace output tells us the following:
  - Start-up finished in 2s 124ms (kernel) + 2s 331ms (initrd) + 35s 426ms (userspace)

  and analysis of the systemd plot shows us more high-level data: which service took how much time to initialize, and what needed to wait for it. Clearly there are a lot of components started that are not needed within an Automotive solution and the ordering of the components is not optimal for us.

  By changing the unit configuration files we were able to move components to the start of the dependency graph as we would need to for PDC/RVC and also were able to remove components from the start-up.

  Therefore we saw no reason to believe that an Automotive start-up could not be possible using systemd. However we identified that we need to alter the target clustering to meet our requirements and would need to use an embedded libc to reduce the initial overhead.

The concept

Overview of the GENIVI boot phases (target clustering)
Focused.target

BootManager

The focused.target will contain the GENIVI Lifecycle Boot Manager. It will be responsible for the following actions:

- reading the LastUserContext (LUC) persistent information
- parsing the LUC data to determine which unit or service to start in the current phase
- requesting systemd to start the required units (based on unit name)
- signalling systemd when it has requested all the units that it requires (i.e. when Boot Mgr has finished initialization)
- providing an interface that can be used to update the LUC persistent information
- informing the Node State Manager about the status of the current system startup
- Option: Triggering the Resource Management to update the cgroup configuration in the system

*)unit: Could be a target unit or service unit(s).

Focused Application Types

To determine the focused and unfocused applications in the system it is proposed to create the following three types of focused applications:

- Audible
  - This specifies the current audible source within the Head Unit (i.e. Radio)
- Foreground
  - This specifies the application that is currently in focus on the HMI (i.e. Browser)
- Background
  - This specifies an application that is running in the background (i.e. Navigation)

It is proposed that an interface will be provided that will allow for applications to be registered to each of the focused types listed above.

This interface can be used by the HMI, applications themselves or some other controlling component to fill the LUC persistent data.

To parse the LUC persistent data optimally it is proposed that the complete unit name (i.e. navigation.target/service) is used on this interface. This will then be used by the Boot Manager on the next start-up to determine what should be started in the focused target phase.

Application Design Constraints

The application is responsible for delaying the standard systemd initialized signal. It must only signal completion when it is completely up and running (i.e. for Navigation when address input is possible or route planning is complete) and not when initialization is complete.

If the application was to signal systemd when the initialization was complete, then systemd could continue with the unfocused.target potentially blocking high priority applications.
If target-units are used to control the focused boot phase:
It is proposed that a “target” is created for the main Automotive features (i.e. navigation.target, mediaplayer.target etc.) and this unit name is used during the registration. In this way it is easy to include all dependent components for the main application. Alternatively it would be possible to create a “target” file during the integration/build phase for each of the target configurations possible in the system. In this way it would be possible to have a more precise resource management split dependent on the three applications filling the three LUC focused types.

If service-units are used to control the focused boot phase:
The registered application must ensure that the registered unit is configured in a way to pull in all components that it is dependent on (i.e. a top level feature unit)

Example flow

1. systemd accesses focused.target.wants directory to find required components
2. systemd starts the BootMgr as part of the focused.target
3. BootMgr reads the focused application string, i.e. navigation.target
4. BootMgr tells systemd to start navigation.target
5. systemd parses navigation.target for dependencies
6. systemd parses hmi.service, gps.service and tuner.service for dependencies
7. systemd starts HMI, GPS and Tuner in parallel
8. HMI, Tuner and GPS applications signal systemd that they are running
9. systemd starts navigation
10. Navigation runs up to point where route is calculated and displayed
11. Navigation signals systemd that it is running
12. Boot Mgr reads the audible application string, i.e. nothing
13. BootMgr tells systemd to start radio.service
14. systemd parses radio.service for dependencies
15. systemd starts Radio
16. Radio signals systemd that it is running
17. Bootmanager signals systemd that it is finished
18. systemd moves on to unfocussed.target

Note: Targets and services can be used. The policy what need to be started first (Background / Audible / Foreground) is OEM specific.

Unfocussed.target(s)

Usage

The unfocussed.target(s) will include all core applications that are not already included within the Mandatory boot phase.

This target will be used to start all applications that the user is/was not currently using. The target itself will include applications that have already been started as part of the focused.target but these will be ignored by systemd.

All components will be started in parallel where possible.

Lazy.target

Usage

The lazy.target will include background applications that can be delayed until everything else in the system has been started. Additionally the signal “FULLY_RUNNING” will allow blocked lazy activities to start (i.e. disk defragmentation).

An example

Example of Subsystem Start-up Order using systemd + BootManager
Here, using systemd, it can be seen that the Boot Manager evaluates the LUC and starts the Navigation. This in turn pulls in Positioning, Tuner and HMI.

### Appendix

#### Special system units

- **basic.target**
  - A special target unit covering early boot-up.
- **ctrl-alt-del.target**
  - systemd starts this target whenever Control+Alt+Del is pressed on the console. Usually this should be aliased (symlinked) to reboot.target.
- **dbus.service**
  - A special unit for the D-Bus system bus. As soon as this service is fully started up systemd will connect to it and register its service.
- **default.target**
  - The default unit systemd starts at boot-up. Usually this should be aliased (symlinked) to multi-user.target or graphical.target.
- **display-manager.service**
  - The display manager service. Usually this should be aliased (symlinked) to xdm.service or a similar display manager service.
- **emergency.target**
  - A special target unit that starts an emergency shell on the main console. This unit is supposed to be used with the kernel command line option systemd.unit= and has otherwise little use.
- **graphical.target**
  - A special target unit for setting up a graphical login screen. This pulls in multi-user.target.
  - Units that are needed for graphical login shall add Wants dependencies for their unit to this unit (or multi-user.target) during installation.
- **halt.target**
  - A special target unit for shutting down and halting the system.
  - Applications wanting to halt the system should start this unit.
- **kbrequest.target**
  - systemd starts this target whenever Alt+ArrowUp is pressed on the console. This is a good candidate to be aliased (symlinked) to rescue.target.
- **local-fs.target**
  - systemd automatically adds dependencies of type After to all mount units that refer to local mount points for this target unit. In addition, systemd adds dependencies of type Wants to this target unit for those mounts listed in /etc/fstab that have the auto and comment=systemd.mount mount options set.
  - systemd automatically adds dependencies of type After for this target unit to all SysV init script service units with an LSB header referring to the $local_fs facility.
- **mail-transfer-agent.target**
  - The mail transfer agent (MTA) service. Usually this should pull-in all units necessary for sending/receiving mails on the local host.
  - systemd automatically adds dependencies of type After for this target unit to all SysV init script service units with an LSB header referring to the $mail-transfer-agent or $mail-transport-agent facilities, for compatibility with Debian.
- **multi-user.target**
  - A special target unit for setting up a multi-user system (non-graphical). This is pulled in by graphical.target.
  - Units that are needed for a multi-user system shall add Wants dependencies to this unit for their unit during installation.
- **network.target**
  - systemd automatically adds dependencies of type After for this target unit to all SysV init script service units with an LSB header referring to the $network facility.
- **nss-lookup.target**
  - systemd automatically adds dependencies of type After for this target unit to all SysV init script service units with an LSB header referring to the $named facility.
- **poweroff.target**
  - A special target unit for shutting down and powering off the system.
  - Applications wanting to power off the system should start this unit.
- **reboot.target**
  - A special target unit for shutting down and rebooting the system.
  - Applications wanting to reboot the system should start this unit.
- **remote-fs.target**
  - Similar to local-fs.target, but for remote mount points.
  - systemd automatically adds dependencies of type After for this target unit to all SysV init script service units with an LSB header referring to the $remote_fs facility.
- **rescue.target**
  - A special target unit for setting up the base system and a rescue shell.
- **rpcbind.target**
  - systemd automatically adds dependencies of type After for this target unit to all SysV init script service units with an LSB header referring to the $rpcbind facility.
- **time-sync.target**
  - systemd automatically adds dependencies of type After for this target unit to all SysV init script service units with an LSB header referring to the $time facility.
- **shutdown.target**
  - A special target unit that terminates the services on system shutdown.
  - Services that shall be terminated on system shutdown shall add Conflicts dependencies to this unit for their service unit, which is implicitly done when DefaultDependencies=yes is set (the default).
  - systemd automatically adds dependencies of type Conflicts to this target unit for all SysV init script service units that shall be terminated in SysV runlevels 0 or 6.
- **sigpwr.target**
  - A special target that is started when systemd receives the SIGPWR process signal, which is normally sent by the kernel or UPS daemons when power fails.
- **sockets.target**
  - A special target unit that sets up all service sockets.
  - Services that can be socket-activated shall add Wants dependencies to this unit for their socket unit during installation.
- **sysinit.target**
  - A special target unit covering early boot-up scripts.
  - systemd automatically adds dependencies of the types Wants and After for all SysV service units configured for runlevels that are not 0 to 6 to this target unit. This covers the special boot-up runlevels some distributions have, such as S or b.
- **syslog.target**
  - systemd automatically adds dependencies of type After for this target unit to all SysV init script service units with an LSB header referring to the syslog facility.
- **systemd-initctl.service**
  - This provides compatibility with the SysV /dev/initctl file system FIFO for communication with the init system.
  - This is a socket-activated service, see system-initctl.socket.
- **systemd-initctl.socket**
  - Socket activation unit for system-initctl.service.
- **systemd-shutdown.service**
  - This is internally used by systemd to provide syslog logging to the processes it maintains.
  - This is a socket-activated service, see system-loggger.socket.
- **systemd-shutdown.socket**
  - Socket activation unit for system-shutdown.service.
- **umount.target**
  - A special target unit that umounts all mount and automount points on system shutdown.
  - Mounts that shall be unmounted on system shutdown shall add Conflicts dependencies to this unit for their mount unit, which is implicitly done when DefaultDependencies=yes is set (the default).

---

SysInfraEGLifecycleConceptCgroupsInput

cgroups input collection

This page collects various inputs concerning cgroups. If you want to add something create a new heading with the title of your post and put your name below.

See genivi-dev mail for details of the origin.

Use Cases for system behavior

How do you want the system to behave? Come up with use cases. (see also https://collab.genivi.org/issues/browse/GT-555)
cgroups deployment and experience

Share your experience with cgroups configuration of systems and how you would like deployed GENIVI Linux systems to look like.

The following figure shows a node cgroup hierarchy for CPU-load as a proposal, which will be used in the F2F in Ismaning Nov 2011:

The diagram shows a node cgroup hierarchy with the following hierarchy:
- **Unlimited, ROOT**
  - PDC
  - View Cameras
  - SW Loading
  - Infra Services
  - Vehicle Network
  - Positioning
  - Comm Stacks
  - Diag

- **AUTOMOTIVE**
  - Radio
  - NAV
  - Speech
  - Media
  - Phone

- **APPS**
  - 3rd party
  - other
  - APPS
  - Browser

- **3rd party, BGND**
  - Background tasks

- **CPU shares = 50, runtime = 1000, period = 2000**

Multicore behavior of cgroups

We especially would like to get information or involvement in respect to multicore systems and cgroup behavior - if you have experience or want to investigate this topic please contact Torsten Hildebrand.

Collection of links

- Lennart's general behavior recommendations on cgroups
- [Christian Muck 2011-08-26] man page for systemd cgroup settings
- [Christian Muck 2011-08-26] systemd cgroups limit patch

SysInfraEGLifecycleConceptMultiNode

Subdomain Lifecycle Multi-Node Concept
Overview of a multi-node deployment

The following figure shows an abstract 3 node deployment with some aspects from Lifecycle.

**Description**

The multi-node concept for Lifecycle is based on the idea that each node has its own commander, so each node is first of all independent (like a standalone node). That means each node has its own state-machine which is driven by all incoming internal and external event/messages. Those messages originally are generated by the ECUs connected to the vehicle bus, by ECU owned hard- and software and by the ECU user. There is no master - slave relation between nodes and no states are somehow mirrored on another node.

For GENIVI application Lifecycle will offer platform own-defined states, sessions, events and messages which can also be used for inter node communication (INC). Because of OEM specific business logic also the raw (OEM defined) states, sessions, events and messages are provided to each node.

Common resources like the power supply have its own commander (realized as a plug-in), controlled by the local state-machine. If another node has got some requests like shutdown or reboot, it will send an event/message to the power supply command node and this node will decide based on internal state / session and request priority if the request can be satisfied.

In fact that the Linux operating system is not used on all nodes (because they are small performance CPUs, no MMU is supported,...) the requested interface of the Lifecycle management has to be fulfilled. The realization of this interface will be a message catalog based on the INC.

**Rationale**

The component implementation of Lifecycle shall be totally independent from the deployment decision (one or more node based system/ECU). Local events shall be handled locally. Changes or extensions of typical local events should not request updates/changes on other nodes. Additionally a decentralized management reduces the complexity a lot. A state-machine owner need only to take care about the local events and the defined events for shared resources (if they are existing at this level).
Subdomain Lifecycle Concept Refinement

The orange blocks are the first level clustering of the subdomain Lifecycle. The blue blocks are the logical components of the subdomain Lifecycle. This figure defines the refinement of the first level clustering and the logical components.

Subdomain Lifecycle System Border / Context

Lifecycle as a black box from application point of view

The following figure shows the interfaces of Lifecycle to the rest of the system. Provided interfaces which can be used by applications or others and requested interfaces which has be either fulfilled or stubbed by the product development. Additionally Lifecycle will be dependent on the subdomains Persistency and Log&Trace.
SysInfraEGLifecycleDesignStructDef

Subsystem Lifecycle Design Structure Definition

- Subsystem Lifecycle Structure Scope (current)
- Node State Management Structure Definition
- Boot Management Structure Definition

Subsystem Lifecycle Structure Scope (current)

The following figure shows the current scope of the structure definition. That will be changed updated next with the resource management.
Node State Management Structure Definition

The following figure shows the structure definition including the proposals for:

- the refinement
- the GENIVI compliance stereotype
- the component license
- the GENIVI priority
- the interface relations
- the dependencies

Boot Management Structure Definition

The following figure shows the structure definition including the proposals for:

- the refinement
- the GENIVI compliance stereotype
- the component license
- the GENIVI priority
- the interface relations
- the dependencies
In the above diagram the "Application Responsibility" components are placeholder components that are not part of Boot Management but rather are to be provided by each Application in the system. Every Application that is to be started by systemd will need to notify/signal systemd when it has completed its initialization. This signal will then be used by systemd to control the dependency synchronization during the start-up sequence.

They are included in the Boot Management compliancy diagrams as we want to ensure that this application requirement/design constraint is fulfilled as needed.

---

**SysInfraEGLifecycleDesignWhiteBoxUcRealization**

**Subdomain Lifecycle Design - White Box Use Case Realization**

- Scope of this page
- Boot Management
- Resource Management
- System Health Monitor
- Thermal Management
- Node State Management
- Questions and Answers
  - Application blocking system shutdown
  - Cancel ongoing system shutdown
  - Consumer blocking system shutdown
  - System Start-up using Last user mode
  - System Start-up with User Selection
  - System Recovery from an application failure

**Scope of this page**

The scope of this work package is the definition of White Box Use Cases for the Subdomain Lifecycle.

Below you will find a table for each of the Logical Components within the Subdomain Lifecycle. Each table will document the currently available Use Cases for that component and include a status column. The absolute latest version of the use cases are in Enterprise Architect but after major changes the attached exported version will be updated.

Below the tables is a space for people to add questions for any use cases.

**Boot Management**
<table>
<thead>
<tr>
<th>Alias</th>
<th>Originator (Name,Company)</th>
<th>Use Case</th>
<th>State</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC_LCBM_001</td>
<td>David Yates, Continental</td>
<td>System Start-up - Last User Mode</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_002</td>
<td>David Yates, Continental</td>
<td>System Start-up - User Selection</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_003</td>
<td>David Yates, Continental</td>
<td>System Shutdown - Application blocking shutdown</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_004</td>
<td>David Yates, Continental</td>
<td>System Shutdown - Consumer blocking shutdown</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_005</td>
<td>David Yates, Continental</td>
<td>System Shutdown - Cancel ongoing shutdown</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_006</td>
<td>David Yates, Continental</td>
<td>System Restart - Synched reboot into SWL mode</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_007</td>
<td>David Yates, Continental</td>
<td>System Shutdown - Cancel shutdown request arrives too late to cancel the shutdown sequence</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_008</td>
<td>David Yates, Continental</td>
<td>System Shutdown - Fast shutdown requested during Coding Phase</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_009</td>
<td>David Yates, Continental</td>
<td>System Shutdown - Normal Shutdown</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_010</td>
<td>David Yates, Continental</td>
<td>System Shutdown - Incoming call forces shutdown cancellation</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_011</td>
<td>David Yates, Continental</td>
<td>System Start-up - Over temperature issue causes shutdown during start-up</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCBM_012</td>
<td>David Yates, Continental</td>
<td>System Start-up - Hangup during start-up causes system reboot</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
</tbody>
</table>

**Resource Management**

<table>
<thead>
<tr>
<th>Alias</th>
<th>Originator (Name,Company)</th>
<th>Use Case</th>
<th>State</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC_LCRM_001</td>
<td>David Yates, Continental</td>
<td>Partition System Resources</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCRM_002</td>
<td>David Yates, Continental</td>
<td>Application requests additional resources</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
</tbody>
</table>

**System Health Monitor**

<table>
<thead>
<tr>
<th>Alias</th>
<th>Originator (Name,Company)</th>
<th>Use Case</th>
<th>State</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC_LCSH_001</td>
<td>David Yates, Continental</td>
<td>System recovery from application failure</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCSH_002</td>
<td>David Yates, Continental</td>
<td>System recovery from failed filesystem in system startup</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
<tr>
<td>UC_LCSH_003</td>
<td>David Yates, Continental</td>
<td>Software update process and the interaction with Lifecycle</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
</tbody>
</table>

**Thermal Management**

<table>
<thead>
<tr>
<th>Alias</th>
<th>Originator (Name,Company)</th>
<th>Use Case</th>
<th>State</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC_LCTM_001</td>
<td>David Yates, Continental</td>
<td>Thermal management requires system shutdown</td>
<td>Released</td>
<td>David Yates, Continental -Reviewed and released</td>
</tr>
</tbody>
</table>

**Node State Management**
### Alias | Originator (Name,Company) | Use Case | State | Comment
--- | --- | --- | --- | ---
UC_LCNM_001 | David Yates, Continental | Session state change requested | Released | David Yates, Continental - Reviewed and released

#### Questions and Answers

**Application blocking system shutdown**

*Question Christian Muck, 2011-05-26:* I think the node state manager has the feasibility to cancel the shutdown sequence before the synchronous call (response?) before "turn off power" will be executed. Is this "validate shutdown allowed"?

*Answer David Yates, 2011-05-26:* I would say that the interface has to be synchronous as the consumer must know when the data has actually been updated before proceeding.

*Question Christian Muck, 2011-05-26:* Is "persist data" from consumers to persistency synchronous or asynchronous?

*Answer David Yates, 2011-05-26:* I would say that the interface has to be synchronous as the consumer must know when the data has actually been updated before proceeding.

**Cancel ongoing system shutdown**

*Question Christian Muck, 2011-05-26:* Is the respond "state change notification(accept) and "state change accept(shutdown)" basically the same after the call "state change request(shutdown)"?

*Question Christian Muck, 2011-05-26:* Why is in this sequence chart the call "persist data" from a consumer to persistency? Can this done only by special consumers or is this normally the task of the node state manager?

**Consumer blocking system shutdown**

*Question Christian Muck, 2011-05-26:* I think the node state manager has to validate the "point of no return" before "turn off power"

*Answer David Yates, 2011-05-26:* I do not think that that is relevant for this use case. The Lifecycle only cares about the "point of no return" in the cancel shutdown use case but you are right that this is not shown in that diagram.

*Answer David Yates, 2011-05-26:* I do not think that that is relevant for this use case. The Lifecycle only cares about the "point of no return" in the cancel shutdown use case but you are right that this is not shown in that diagram.

*Question Christian Muck, 2011-05-26:* "state_change_request(shutdown)" is called from node state manager to persistency. When the persistency will be flushed. Should we adopt this part or why is it here only "persist data"?

*Answer David Yates, 2011-05-26:* You are right this is inconsistent to the other use cases. The handling should be the same.

*Question Christian Muck, 2011-05-26:* Is "persist data" from consumers to persistency synchronous or asynchronous? - Same question at App Blocking Shutdown

*Answer David Yates, 2011-05-26:* From my perspective it has to be synchronous as the consumers must know when the data has actually been written before proceeding.

**System Start-up using Last user mode**

*Question Christian Muck, 2011-05-12:* Apps signal systemd that they are initialized. We should explain the interfaces of systemd on SysInfraEGSystemd. Did Lennart say something about interfaces of systemd?


*Question Christian Muck, 2011-05-12:* What is exactly the difference between non-active and lazy apps?

*Question Christian Muck, 2011-05-12:* When a non active app gets the signal from user systemd to initialize, should it "run to a pre-working state", similar as the active app "run to working state"?

*Answer David Yates, 2011-05-16:* Yes, we would envisage that a Non-active apps will run to a "quiet" state whereby it can start quickly if needed but where it does not take up too many resources. This was the reason for checking with Lennart whether it was possible to easily pass a parameter via systemd to the component that could be used to tell it which state to initialize in. As this does not seem to be that easy it will be up to the Application to read the Last User Mode during initialization to determine its running state.

**System Start-up with User Selection**

*Question Christian Muck, 2011-05-12:* At SystemInfraEGLifecycleDesignWBLastUserMode there is always a response from an app to systemd 'initialized'. Did Lennart say something about interfaces of systemd?

*Answer David Yates, 2011-05-16:* Correct, the component should always send a signal to systemd, that was just an oversight from my side. I will update this in the future.

**System Recovery from an application failure**
SysInfraEGLifecycleFABlackBoxUcRealization

Subsystem Lifecycle FA Black Box Use Case Realization

- Lifecycle Management Black Box export from Enterprise Architect

Lifecycle Management Black Box export from Enterprise Architect

For the latest architecture regarding the black box view of the Lifecycle Management please check out the Infrastructure branch of the Enterprise Architect model. However, here is an export of the last baseline snapshot from 12th July 2011.

SysInfraEGLifecycleNSMData

Subdomain Lifecycle - Node State Manager

Component Overview

The node state management is the central repository for information regarding the states/sessions inside the node. It collates information from multiple sources and uses this to determine the current states (there are different states existing for different purposes). This information will be delivered to registered consumers or can be requested via the provided interface. All the raw data gathered by the node state management is also available on request to interested consumers.

The node state management also provides the shutdown management, so one part of the information which is provided is the shutdown request notification event/message to the consumers. The boot management only boots the node and has to stop the boot activity in case of a high priority shutdown request, which is initiated by the node state management. That means boot management is just a consumer of the node state management.

The node state management is the last/highest level of escalation on the node, therefore it will command the reset and supply control logic. Additionally if the system is a multi-node system the node state management has included a slave (done as a consumer) which knows the specifics of this configuration and knows about what event/message need to be transferred also to other node(s).

Node State Information

The following table details the data that will be used/controlled by the Node State Manager.

| Data Name | name of the data | Data Values | Values that the data can have | Source | This is the person who will change this data item to this particular value of the data | NSM State Relevant | This defines whether the NSM will take into account this particular value of the data when evaluating node state changes | NSM Trigger Event | If the data item is owned by the NSM then this is the interface that will be used to update the value of the data | Distributed | This specifies whether the Data Item will be distributed by the NSM to interested consumers | User, Seat or Node | This determines whether the data item relates to the Node, a particular Seat/View or for a particular User |
## Comment -> General comments about the data value

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Data Values</th>
<th>Source</th>
<th>NSM State Relevant</th>
<th>NSM Trigger Event</th>
<th>Distributed</th>
<th>Data consumer</th>
<th>User, Seat or Node</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Boot Mode</td>
<td>SWL, Application, Test Software (TSW)</td>
<td>Node State Manager</td>
<td>Yes</td>
<td>Boot mode change only performed by Node State Manager trigger by applications.</td>
<td>Yes</td>
<td>Bootloader, NSM and some other Lifecycle components</td>
<td>Node</td>
<td>Recommendation is that session state hand should be used and is the key data use applications. Boot Mode is only used by specific enabled applications. Constraint here is that persistency must be available in early boot phase. Used by the boot loader to decide which kernel image to start in the system. Maybe not needed NSM as each kernel image will have its state machine. Question is whether to include the current boot mode in the information distributed by the NSM or whether this should always be read from Persistency. Advantages for NSM is that all consumers would get the current boot mode and not the currently stored boot mode.</td>
</tr>
<tr>
<td>Node Application Mode</td>
<td>Parking, Factory, Transport, SWL</td>
<td>Node State Manager (triggered by Diagnostics, SWL or SHM)</td>
<td>Yes</td>
<td>Node Application mode change I/F</td>
<td>Yes</td>
<td>NSM, BMGR and other applications</td>
<td>Node</td>
<td>The NSM will support a selection of &quot;application modes&quot; which when selected can result in different levels of application functionality and state handling. i.e. Transport mode - only PDC and RVC applications are required, all other functionality is disabled.</td>
</tr>
<tr>
<td>Node State</td>
<td>Start-up</td>
<td>Default value</td>
<td>Yes</td>
<td>NSM initialization automatically sets this value</td>
<td>Yes</td>
<td>NSM, SHM</td>
<td>Node</td>
<td>Default value and is taken until Fully Running until Fully Running.</td>
</tr>
<tr>
<td>Node State</td>
<td>Base Running</td>
<td>Boot Manager</td>
<td>Yes</td>
<td>Node State change I/F</td>
<td>Yes</td>
<td>NSM, SHM</td>
<td>Node</td>
<td>Base Running means that mandatory components have been started; the Boot Manager is just been called for initialization.</td>
</tr>
<tr>
<td>Node State</td>
<td>Last User Context (LUC) Running</td>
<td>Boot Manager</td>
<td>Yes</td>
<td>Node State change I/F</td>
<td>Yes</td>
<td>NSM, SHM</td>
<td>Node</td>
<td>LUC Running will be entered when the LUC Manager has started components needed within the focused phase.</td>
</tr>
<tr>
<td>Node State</td>
<td>Fully Running</td>
<td>Boot Manager</td>
<td>Yes</td>
<td>Node State</td>
<td>Yes</td>
<td>NSM, SHM and some Lifecycle consumers</td>
<td>Node</td>
<td>Fully Running means all applications requested are active but some operational functionality is disabled (i.e. defragmentation)</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Node State</td>
<td>Fully Operational</td>
<td>Boot Manager</td>
<td>Yes</td>
<td>Node State</td>
<td>Yes</td>
<td>NSM, SHM and may some Lifecycle consumers</td>
<td>Node</td>
<td>Fully Operational if system is complete and running</td>
</tr>
<tr>
<td>Node State</td>
<td>Degraded Power</td>
<td>NSM (via Supply Mgmt)</td>
<td>Yes</td>
<td>Node State</td>
<td>Yes</td>
<td>SHM and some Lifecycle consumers</td>
<td>Node</td>
<td>This state will be active when the NSM receives a &quot;Poor&quot; state from Supply Management but shutdown is not currently possible</td>
</tr>
<tr>
<td>Node State</td>
<td>Shutdown Delay</td>
<td>Node State Manager</td>
<td>Yes</td>
<td>Clamp change/Network activity</td>
<td>No</td>
<td>Node</td>
<td>A shutdown due to Clamp Change/Net activity has been initiated. The Head will delay for a configurable (OEM specific) period of time before initiating the shutdown. This also allows for a &quot;Poor&quot; state from Supply Management.</td>
<td></td>
</tr>
<tr>
<td>Node State</td>
<td>Shutting Down</td>
<td>Node State Manager</td>
<td>Yes</td>
<td>Clamp change timeout, Diagnostic request, SW request, System Health Monitor, Thermal/Supply Mgmt</td>
<td>Yes</td>
<td>All shutdown consumers (incl. BMGR, SHM,...)</td>
<td>Node</td>
<td>System shutdown has been received or clamp change timeout from state &quot;Shutdown Delay&quot; has occurred</td>
</tr>
<tr>
<td>Node State</td>
<td>Fast Shutdown</td>
<td>Node State Manager</td>
<td>Yes</td>
<td>Clamp change, Diagnostic request, SW request, System Health Monitor, Thermal Mgmt, Supply Mgmt</td>
<td>Yes</td>
<td>All fast shutdown consumers (limited)</td>
<td>Node</td>
<td>System is in process of performing a fast shutdown. This will normally be called on a Diagnosis request when it is known that there is no persistent data to be written. This would result in no consumers being called in the shutdown sequence</td>
</tr>
<tr>
<td>Supply State</td>
<td>Good, Poor, Bad</td>
<td>Supply Manager</td>
<td>Yes</td>
<td>Generic HW State I/F</td>
<td>No</td>
<td>Node</td>
<td>Good = Nothing to do, Poor = Shutdown if possible, Bad = Shutdown immediate</td>
<td></td>
</tr>
<tr>
<td>Thermal State</td>
<td>Good, Poor, Bad</td>
<td>Thermal Manager</td>
<td>Yes</td>
<td>Generic HW State I/F</td>
<td>No</td>
<td>Node</td>
<td>Good = Nothing to do, Poor = Shutdown if possible, Bad = Shutdown immediate</td>
<td></td>
</tr>
<tr>
<td>Node Session State</td>
<td>Heating, Ventilation, AirCon (HEVAC) Active</td>
<td>Application</td>
<td>Yes</td>
<td>Session State change I/F</td>
<td>Yes</td>
<td>NSM and some applications</td>
<td>Seat</td>
<td>This is used by certain OEMs and will also to be a wake-up reason as it defines we start to a point where we can control the HEVAC</td>
</tr>
<tr>
<td>Node</td>
<td>Session</td>
<td>Component/Mode</td>
<td>Status</td>
<td>Changed I/F</td>
<td>NSM and some applications</td>
<td>Seat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------</td>
<td>----------------------------------------------------</td>
<td>--------</td>
<td>-------------</td>
<td>---------------------------</td>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVC</td>
<td>Rear View Camera (RVC) Active</td>
<td>RVC</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM and some applications</td>
<td>Seat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDC</td>
<td>Park Distance Control (PDC) Active</td>
<td>PDC</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM and some applications</td>
<td>Seat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMI</td>
<td>HMI Active</td>
<td>HMI</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM and some applications</td>
<td>Seat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>Network Passive State active</td>
<td>Network Manager</td>
<td>Yes</td>
<td>No</td>
<td>NSM and some applications</td>
<td>Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network active</td>
<td>Network Manager</td>
<td>Yes</td>
<td>No</td>
<td>NSM and some applications</td>
<td>Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td>Phone active</td>
<td>Phone Application</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM and some applications</td>
<td>Seat</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Permanent / Entertainment Mode active</td>
<td>NSM</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM and some applications</td>
<td>Node</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Loading (SWL)/Flash Mode active</td>
<td>SWL/Diagnosis</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM, SHM, SRM, SWL appl. and some applications</td>
<td>Node</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is a product decision but it is likely that the RVC active will overrule a Poor and non critical failure.

This is a product decision but it is likely that the PDC active will overrule a Poor and non critical failure.

This is an open set state that can be used differently for different products. Traditionally it is expected that this session state will be used by the HMI when the vehicle is completely run up and all graphics have been rendered on the appropriate layer. This could for instance be used to determine when to enable the display or to switch a Splashscreen to the real HMI to ensure the user does not see the HMI before it is completely ready.

This state will be true when there is Network Activity but the Head Unit is in a user perceived off state and therefore is not directly using the Network.

This state will be true when there is Network Activity and the Head Unit is in a user perceived on state and therefore is directly reliant on the Network. Indicates that a phone call is in progress and would normally delay the NSM from delaying the system shutdown.

Permanent/Entertainment mode is normally active when the user has started the target via the Power On button and the clamp state is not active. This mode would normally allow the target to run for a configurable period of time before automatic shutdown occurs.

When SWL is in progress we would need to handle reboot requests and recovery requests differently.
<table>
<thead>
<tr>
<th>Node Session</th>
<th>Diagnostics Active</th>
<th>Diagnostics</th>
<th>No</th>
<th>Session State change I/F</th>
<th>Yes</th>
<th>Session State change I/F</th>
<th>Yes</th>
<th>Node State</th>
<th>Session State change I/F</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Session</td>
<td>Additional Product Session States</td>
<td>Product Extensions</td>
<td>Yes</td>
<td>Session State change I/F</td>
<td>Yes</td>
<td>NSM and some applications</td>
<td>Seat</td>
<td>Predefined product states with no defined GENIVI meaning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>Unknown</td>
<td>NSM</td>
<td>No</td>
<td>Yes</td>
<td>NSM, BMGR, Error Mgmt, DIAG SHM and may some applications</td>
<td>Node</td>
<td>This is the default value and would be used catch all for unhandled exceptions/restarts systems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>First Switch To Power (FSTP)</td>
<td>NSM</td>
<td>No</td>
<td>Yes</td>
<td>NSM, BMGR, DIAG SHM and may some applications</td>
<td>Node</td>
<td>This would be the state when the user performed first switch to power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>CAN Activity</td>
<td>Power Event Controller</td>
<td>No</td>
<td>Power Event I/F</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td>This is the normal wake-up reason and would initiate a standard system start-up</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>Software</td>
<td>Power Event Controller</td>
<td>No</td>
<td>Power Event I/F</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td>A system restart request via software (SWL, Diagnosis, SHM etc) would result in a &quot;Software&quot; wake-up reason. The NSM would handle it the same way as the &quot;CAN activity&quot; reason.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>Eject</td>
<td>NSM (Front Panel)</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td>An &quot;eject&quot; wake-up reason would be used in the NSM as the typical use case is that the target is not completely run-up. Typically we would run up enough to allow for the eject to be handled. We would then shutdown again with media has been removed completely from the drive.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>Media Insertion</td>
<td>NSM (VuC/PIC)</td>
<td>No</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td>Media insertion is relevant to the NSA machine as it does alter the Last User From the NSM point of view a &quot;Media Insertion&quot; would trigger the same handling as the &quot;ECU Power On button&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>ECU Power On button</td>
<td>NSM (Front Panel)</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td>This wake-up reason would trigger the change of Node Session State to Entertainment Mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>HEVAC</td>
<td>NSM (Front Panel)</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td>Input via the HEVAC wake up the Head (this is OEM option and would only act) the HEVAC session with a reduced set functionality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>Phone</td>
<td>NSM (via Network)</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td>In-car phone can wake up the Head Unit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
<td>-------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------------------------------</td>
<td>------</td>
<td>--------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up Reason</td>
<td>Unexpected Reset</td>
<td>NSM</td>
<td>Yes</td>
<td>Yes</td>
<td>NSM, BMGR, Error Mgmt and some applications</td>
<td>Node</td>
<td>Set when the NSM validates the shutdown reason from the prior lifecycle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shutdown Reason</td>
<td>Thermal Poor</td>
<td>Thermal Manager</td>
<td>Yes</td>
<td>Updated by NSM at the point of triggering the shutdown</td>
<td>No</td>
<td>NSM</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shutdown Reason</td>
<td>Thermal Bad</td>
<td>Thermal Manager</td>
<td>Yes</td>
<td>Updated by NSM at the point of triggering the shutdown</td>
<td>No</td>
<td>NSM</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shutdown Reason</td>
<td>Supply State Poor</td>
<td>Supply Manager</td>
<td>Yes</td>
<td>Updated by NSM at the point of triggering the shutdown</td>
<td>No</td>
<td>NSM</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shutdown Reason</td>
<td>Supply State Bad</td>
<td>Supply Manager</td>
<td>Yes</td>
<td>Updated by NSM at the point of triggering the shutdown</td>
<td>No</td>
<td>NSM</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shutdown Reason</td>
<td>Normal (network)</td>
<td>NSM (based on clamp state)</td>
<td>Yes</td>
<td>Updated by NSM at the point of triggering the shutdown</td>
<td>No</td>
<td>NSM</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restart Reason</td>
<td>Application Failure</td>
<td>NSM</td>
<td>Yes</td>
<td>Updated by NSM at the point of triggering a system restart</td>
<td>Yes</td>
<td>NSM, SHM, BMGR and some applications</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restart Reason</td>
<td>SW Loading</td>
<td>NSM</td>
<td>No</td>
<td>Updated by NSM at the point of triggering a system restart</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restart Reason</td>
<td>Diagnostics</td>
<td>NSM</td>
<td>No</td>
<td>Updated by NSM at the point of triggering a system restart</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restart Reason</td>
<td>User</td>
<td>NSM</td>
<td>Yes</td>
<td>Updated by NSM at the point of triggering a system restart</td>
<td>Yes</td>
<td>NSM, BMGR and some applications</td>
<td>Node</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BMW Lifecycle State Chart**

To validate that the GENIVI lifecycle concept in general and the node state management works in real world scenarios, **BMW published its lifecycle state chart**.
This page is under construction.

The following diagram shows the traceability from Vehicle Requirements to Black Box Use cases.
SysInfraEGLifecycleRequirements

Released Subdomain Lifecycle Requirement Definition

Legal Note well
The rules for contributing confidential information are defined in Section 17.1 of the GENIVI Bylaws. Unless information submitted by members to this page is submitted according to those confidentiality rules, all information on this page will be considered non-confidential.

- Scope of this page
- Scope of Lifecycle requirements
- Lifecycle Owned Requirements - Vehicle (1.1.1)
- Lifecycle Owned Requirements - SW Platform (1.2.1)
- Lifecycle Owned Requirements - Mechanism (1.3.1)
- Lifecycle Owned Requirements - Hardware (1.4.1)
- Lifecycle Owned Constraints (1.1.2)
- Lifecycle Requirements to be satisfied outside of Lifecycle Subdomain (2.1.1)
- Lifecycle Constraints to be satisfied outside of Lifecycle Subdomain (2.1.2)

Scope of this page

The scope of this work package is the definition of all requirements and constraints for the Subdomain Lifecycle and all requirements and prerequisites from the Subdomain Lifecycle to someone else.

The structure, the terms and the lifecycle of the requirements and constraints are described here: (SysInfraEGLifecycleRequirementsStructDef)

Scope of Lifecycle requirements

Lifecycle requirements describing the rules of how the ECU is supposed to react to different internal and external conditions including supply voltage level, temperature, vehicle network status, application status, etc. Such requirements are also referred to as Life Cycle, Life Phases,
## Lifecycle Owned Requirements - Vehicle (1.1.1)

<table>
<thead>
<tr>
<th>Alias</th>
<th>Area</th>
<th>Originator (Name, Company)</th>
<th>Description</th>
<th>Additional Info</th>
<th>State</th>
<th>Comment</th>
<th>Assigned To</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH-LICY-001</td>
<td>Component</td>
<td>Mark Hatle, Wind River</td>
<td>Lifecycle Management must override blocking consumers on shutdown</td>
<td>In shutdown situations, Lifecycle Management must timeout after a defined time duration and proceed</td>
<td>Released</td>
<td>David Yates, Continental - Exception needs to added for SWL</td>
<td>NSM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>with shutdown even if one or more consumers or devices fail to respond to a state change request.</td>
<td></td>
<td>(especially in scenario where HU is the master of the SWL Process)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Special handling must be configurable for the Software Loading consumer to prevent potential system</td>
<td></td>
<td>Christian Muck, BMW - Okay. Original Req.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>instabilities.</td>
<td></td>
<td>In shutdown situations, PSM must timeout after a defined time duration</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and proceed with shutdown even if one or more consumers or devices fail to respond to a state</td>
<td></td>
<td>and proceed with shutdown even if one or more consumers or devices</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>change request.</td>
<td></td>
<td>fail to respond to a state change request</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-002</td>
<td>Component</td>
<td>Mark Hatle, Wind River</td>
<td>Supply Management must be able to route hardware power events</td>
<td>Munich Workshop: Hardware power events are part of Supply Management</td>
<td>Released</td>
<td>Munich Workshop: Hardware power events are part of Supply Management</td>
<td>Supply</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Christian Muck, BMW - Not part of GENIVI lifecycle/PSM requirement.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Original Req: Assignment not PSM and should be renamed to Supply</td>
<td></td>
<td>Original Req: Assignment not PSM and should be renamed to Supply</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>David Yates, Continental: Agreed in Telco on 19th May 2011</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-003</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>The system should detect the wake up reason to prevent or modify start-up if</td>
<td></td>
<td>Released</td>
<td></td>
<td>Boot Mgmt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>functionality is not needed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VH-LICY-004</td>
<td>Component</td>
<td><strong>Christian Muck</strong>, BMW</td>
<td>The system functionality to store necessary data is depending on the shutdown mode.</td>
<td>Differentiation between normal shutdown (normal persistence) and fast shutdown, e.g., from diagnostic. In a fast shutdown situation we will not do normal persistence. This will be a product decision on consumers will be called.</td>
<td>Released</td>
<td>David Yates, Continental: Additional information is needed here. Christian Muck, BMW: We differentiate between normal shutdown (normal persistence) and fast shutdown, e.g., from diagnostic. In a fast shutdown situation we will do not normal persistence.</td>
<td>NSM</td>
</tr>
<tr>
<td>VH-LICY-005</td>
<td>Component</td>
<td><strong>Christian Muck</strong>, BMW</td>
<td>In diagnostic or flash sessions the lifecycle is controlled by diagnostic.</td>
<td>Note: Incoming messages over bus will be ignored, e.g. phone call, button pressed. Node internal exceptions from thermal management should not be ignored to save the ECU.</td>
<td>Released</td>
<td>David Yates, Continental: Does this override other exceptions, i.e. Phone call taking place, thermal management etc. Christian Muck, BMW: Incoming messages over bus will be ignored, e.g. phone call, button pressed. Node internal exceptions from thermal management should not be ignored to save the ECU.</td>
<td>NSM</td>
</tr>
<tr>
<td>VH-LICY-006</td>
<td>Component</td>
<td><strong>Fabien Hernandez</strong>, PSA</td>
<td>Before a managed reset/reboot, the system provides a degraded mode in order to maintain main user services.</td>
<td>If diagnosis starts a reset/reboot procedure, system can give a little time to the user to finish actions. This time will be a product configurable value.</td>
<td>Released</td>
<td>David Yates, Continental: Agreed in Telco on 19th May 2011</td>
<td>NSM</td>
</tr>
<tr>
<td>VH-LICY-007</td>
<td>Component</td>
<td><strong>Fabien Hernandez</strong>, PSA</td>
<td>The system has the ability to delay the shutdown depending on application state.</td>
<td>Rational: Ongoing phone call and the driver asks for ignition off (key)</td>
<td>Released</td>
<td>David Yates, Continental: Agreed in Telco on 19th May 2011</td>
<td>NSM</td>
</tr>
<tr>
<td>VH-LICY-008</td>
<td>Component</td>
<td>Fabien Hernandez, PSA</td>
<td>The system has the ability to manage a timeout to shutdown depending on application state.</td>
<td>Rational: When a call is still ongoing a defined period of time (x minutes) after the driver asked for ignition off (key) the call is switch to the phone and the system is shut down.</td>
<td>Released Markus 2011-04-07 Changed wording of rational David Yates, Continental: Agreed in Telco on 19th May 2011</td>
<td>NSM</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-009</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>The system has to follow the global car operation modes.</td>
<td>Ignition on -&gt; system on. Low battery -&gt; system shutdown.</td>
<td>Released David Yates, Continental: Agreed in Telco on 19th May 2011</td>
<td>NSM</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-010</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>There must be a normal operation mode.</td>
<td>Full functionality for all user interactions with ignition on.</td>
<td>Released David Yates, Continental: Agreed in Telco on 19th May 2011</td>
<td>NSM</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-011</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The system can be woken up by the user pressing the Power On button on the ECU</td>
<td></td>
<td>Released David Yates, Added more explicit requirements from Christians original req, Christian Muck, The system can be woken up from network/bus signals.</td>
<td>Boot Mgmt</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-012</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The system can be woken up by the user inserting a CD/DVD in the drive</td>
<td></td>
<td>Released David Yates, Added more explicit requirements from Christians original req, Christian Muck, The system can be woken up from network/bus signals. Torsten: Add DVD eject button</td>
<td>Boot Mgmt</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-013</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The system can be woken up by the user pressing the CD/DVD eject button</td>
<td></td>
<td>Released David Yates, Added following review in F2F meeting</td>
<td>Boot Mgmt</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-014</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The system can be woken up by an active bus signal</td>
<td></td>
<td>Released David Yates, Added more explicit requirements from Christians original req, Christian Muck, The system can be woken up from network/bus signals. Danilo: Add a requirement for internal wake up signal (i.e. internal phone)</td>
<td>Boot Mgmt</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-015</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The system can be shutdown by a specific Clamp State notification</td>
<td>Released</td>
<td>David Yates, Added more explicit requirements from Christians original req. Christian Muck, The system can be woken up from network/bus signals.</td>
<td>NSM</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-016</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The system can be shutdown by the ECU after a product defined timeout when there is no active Clamp State</td>
<td>Released</td>
<td>David Yates, Added more explicit requirements from Christians original req. Christian Muck, The system can be woken up from network/bus signals.</td>
<td>NSM</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-017</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The system can be instructed to shutdown by an external tester Specific diagnostic authentication and protocols will need to be handled before this Diagnosis job is accepted by the ECU</td>
<td>Released</td>
<td>David Yates, Added more explicit requirements from Christians original req. Christian Muck, The system can be woken up from network/bus signals.</td>
<td>NSM</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-018</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>There must be a parking mode. Reduced functionality (i.e. internal HU functionality only) during ignition off but key is present</td>
<td>Released</td>
<td>David Yates, Continental : more information needed, is this mode used when HU is turned on by user but clamp state not active and therefore HU only stays on for 30mins? Simon Barnett, Jaguar: Do not agree with reduced functionality, HU wake-up should wake-up other car components (ECU) for High premium system Danilo, Magneti : More detailed requirement needed to define reduced functionality and flexibility based on OEM requirements Christian Muck, BMW : Okay can be rejected.</td>
<td>Lifecycle</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-019</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>There must be an operation mode for transporting the cars.</td>
<td>No function for the customer (e.g. navigation, entertainment etc.) except safety functions (e.g. PDC) during key presence and ignition on. Different start-up and different functionality to building the cars possible. Only used in factories and service.</td>
<td>Released Torsten Hildebrand: Should be split into two requirements Christian Muck, BMW : Split into two requirements from original: &quot;There must be an operation mode for building and transporting the cars.”</td>
<td>Lifecycle</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-020</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>There must be an operation mode for building the cars in factories.</td>
<td>Reduced functionality (i.e. Diagnosis, PDC, etc.) during key presence and ignition on. Different start-up and different functionality to transporting the cars possible. Only used in factories.</td>
<td>Released Christian Muck, BMW : Split into two requirements from original: &quot;There must be an operation mode for building and transporting the cars.”</td>
<td>Lifecycle</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-021</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>There must be an operation mode for flashing devices.</td>
<td>This mode will be accessible through a protected interface that can only be accessed via authorized applications.</td>
<td>Released David Yates, Continental : is there no customer based updates - has this been agreed with Automotive group? Other OEMs require full system update by end user Christian Muck, BMW : I did not found any Automotive EG requirement for flashing devices. Torsten Hildebrand: Reword still needed to be allow for other applications to trigger the mode. Maybe we need a protected interface that can only be called by certain authenticated consumers</td>
<td>Lifecycle</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-022</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>There must be a thermal management.</td>
<td>Offer interfaces for thermal management and perform proper counter measures if it’s too cold/warm.</td>
<td>Released</td>
<td>Thermal</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-023</td>
<td>Component</td>
<td>Fabien Hernandez, PSA</td>
<td>Following power up, the system is gradually functional</td>
<td>Global requirement related to availability of rear view camera below. We can think also about first picture availability, audio, video, ... (time constraints)</td>
<td>Released</td>
<td>Boot Mgmt</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-024</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The lifecycle must provide watchdog functionality for system applications.</td>
<td>Released</td>
<td>NHM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VH-LICY-025</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>The ECU must handle an immediate power off.</td>
<td>Released</td>
<td>Torsten: We accept this and will drive sub requirements to other domains. Maybe more understandable as &quot;The ECU must handle an emergency power off&quot;</td>
<td>NSM</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-026</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>The lifecycle must ensure safety critical applications can always be started by the driver. It must be shown that at all times in the system that safety critical applications can be started within the same timescales (KPI's) as agreed for a first switch to power. Specifically during the shutdown sequence there can not be a window of opportunity where safety critical applications are blocked from running. NOTE: Exception allowed for when Head Unit is in Flash Mode (i.e. SWL)</td>
<td>Released</td>
<td>David Yates, Continental : Updated to be a true Vehicle requirement and moved original req regarding canceling shutdown into SW Reqs</td>
<td>Boot Mgmt</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-027</td>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>To provide Log&amp;Trace data for problem analysis the DLT must be supported. Each application/component should provide version information and build details in DLT at start-up time.</td>
<td>Released</td>
<td>David Yates, Continental : Reword needed, Lifecycle will use DLT but Additional Info field is wrong Christian Muck, BMW : No additional info field is needed, e.g. the first log messages of a consumer should contain version information and build details.</td>
<td>Lifecycle</td>
<td></td>
</tr>
<tr>
<td>VH-LICY-028</td>
<td>Component</td>
<td>Fabien Hernandez, PSA</td>
<td>Accordingly to its state, the system could generate a reset/reboot</td>
<td>Released</td>
<td>NSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>The system must support a real time clock and an uptime clock.</td>
<td>Open</td>
<td>David Yates, Continental: Discussion needed here, we will forward real time clock information from Vehicle Network but Lifecycle should not be responsible for more. Torsten: This is Infrastructure but is not really Lifecycle, maybe we need a new subdomain to handle this kind of item. Danilo: Will check with Sys Arch team how clocks should be handled in the system.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Fabien Hernandez, PSA</td>
<td>For specific states, the system is not allowed to reboot. Install, upgrade states</td>
<td>Duplicate</td>
<td>David Yates, Continental: Similar requirement above from BMW - exceptions allowed for Thermal Management?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>There must be a communication mode. Full functionality and no system shutdown during phone calls.</td>
<td>Duplicate</td>
<td>Fabien Hernandez: With these requirement do we need to maintain &quot;1&quot; or put it as a SW platform RQ? All, agreed it is a duplicate of other requirements. ZReject</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>There must be a shutdown delay mode. Systems prepares shutdown, no key is present and no functionality is offered to the user.</td>
<td>Duplicate</td>
<td>David Yates, Continental: what is the use case here, who requires this mode? Christian Muck, BMW: Duplicate to which requirement? ZReject</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Fabien Hernandez</td>
<td>The Lifecycle Management can stop applications or reboot the system if a CPU load threshold is reached</td>
<td>Rejected</td>
<td>Munich Workshop: Can be Rejected, agreed with Fabien Markus: 2011-03-31: I think we should create generic check node health requirement instead of this explicit one. ZReject</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Originator (Name, Company)</td>
<td>Description</td>
<td>Additional Info</td>
<td>Priority</td>
<td>State</td>
<td>Comment</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------</td>
<td>-------------</td>
<td>----------------</td>
<td>----------</td>
<td>-------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Christian Muck, BMW</td>
<td>The emergency call must be possible in all situations where the customer is present. Must not be disturbed by entertainment or system shutdown. Note: Exceptions are allowed for SWL and Diagnostic</td>
<td>Rejected</td>
<td>David Yates, Continental : exceptions needed for SWL and Diagnostic sessions Simon Barnett, Jaguar : Emergency calls must be done by another external ECU and therefore emergency calls are out of the scope of IVI Fabian Hernandez, PSA, Maybe it could be fulfilled later with an additional integration of components in the ECU. All, Agreed to reject for now but available later possibly</td>
<td>ZReject</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>David Yates, Continental</td>
<td>The system can be woken up from network/bus signals.</td>
<td>Rejected</td>
<td>David Yates, Added more explicit requirements from Christians original req, Christian Muck, The system can be woken up from network/bus signals.</td>
<td>ZReject</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VH-LICY-029</td>
<td>Component</td>
<td>David Yates, Continental</td>
<td>Lifecycle Management must be able to perform a fast system shutdown when requested</td>
<td>During a fast shutdown it is accepted that persistence data from that lifecycle will not be stored during the shutdown</td>
<td>Released</td>
<td>David Yates, Continental : Requirements added based on F2F meeting in June</td>
<td>NSM</td>
</tr>
</tbody>
</table>

**Lifecycle Owned Requirements - SW Platform (1.2.1)**
<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Component</th>
<th>Description</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW-LICY-001</td>
<td>OEM</td>
<td>Christian Muck, BMW</td>
<td>The Node State Manager of the Automotive Controller must provide mapping between OEM-specific system states and GENIVI States. The Lifecycle Manager API will distribute the OEM and GENIVI states as part of the interface.</td>
<td>P1</td>
<td>Released</td>
</tr>
<tr>
<td>SW-LICY-002</td>
<td>Internal API</td>
<td>Mark Hatle, Wind River</td>
<td>The Node State Manager must send a notification to all registered consumers after a state change has completed.</td>
<td>P1</td>
<td>Released</td>
</tr>
<tr>
<td>SW-LICY-003</td>
<td>API</td>
<td>Mark Hatle, Wind River</td>
<td>The definition and structure of Lifecycle API's must be hardware independent.</td>
<td>P1</td>
<td>Released</td>
</tr>
<tr>
<td>SW-LICY-004</td>
<td>API</td>
<td>Mark Hatle, Wind River</td>
<td>Lifecycle Management API must provide functionality to get a list of all registered consumers.</td>
<td>P1</td>
<td>Released</td>
</tr>
<tr>
<td>SW-LICY-005</td>
<td>VH-LICY-002 API</td>
<td>Mark Hatle, Wind River</td>
<td>The Node State Manager must provide an API to get the current power state.</td>
<td>P1</td>
<td>Rejected</td>
</tr>
<tr>
<td>SW-LICY-006</td>
<td>VH-LICY-002 API</td>
<td>Mark Hatle, Wind River</td>
<td>The Node State Manager must provide an API to switch the power state.</td>
<td>P1</td>
<td>Rejected</td>
</tr>
<tr>
<td>SW-LICY-007</td>
<td>Component</td>
<td>Mark Hatle, Wind River</td>
<td>The Node State Manager must be accessible to multiple consumers.</td>
<td>P1</td>
<td>Released</td>
</tr>
</tbody>
</table>
| SW-LICY-008 | Component | Mark Hatle, Wind River | The Node State Manager must be able to distribute the node state to all consumers | P1 | Released | Christian Muck - Better understanding | Lifecycle Management able to set the state to all consumers. Not part of GE lifecycle to power up/down individual systems and devices. Original Req: Lifecycle Management able to individually power up/down individual systems and devices.

| SW-LICY-009 | VH-LICY-027 | Component | Mark Hatle, Wind River | All Lifecycle components must provide monitoring capabilities with DLT for the Test Framework | P1 | Released | Christian Muck - Better understanding | Lifecycle Management must provide monitoring capabilities with DLT for the Test Framework. Original Req: Lifecycle Management provide monitoring capabilities for the Test Framework.

| SW-LICY-011 | VH-LICY-006 | Component | Mark Hatle, Wind River | The Node State Manager must have the ability to trigger immediate or deferred system reboots | P1 | Released | David Yates, Continental - Component design and naming not finalized hence reworded | Christian Muc - Reworded from schedule to trigger system reboots. Original Req: NHM must have the ability to schedule immediate or deferred system reboots.

| SW-LICY-012 | VH-LICY-026 | Component | Mark Hatle, Wind River | node resource Management must have the ability to restrict and monitor memory levels of processes and notify the appropriate system components | P1 | Released | David Yates, Continental - Component design and naming not finalized hence reworded | NHM should react on Memory information provided by the NRM. Original Req: NHM must have the ability to monitor memory level processes and notify the appropriate system components.
| SW-LICY-013 | VH-LICY-026 | Component | David Yates, Continental | node resource Management must have the ability to restrict and monitor CPU usage levels of processes and notify the appropriate system components | P1 | Released |
| SW-LICY-014 | VH-LICY-024 | Component | Mark Hatle, Wind River | node health Monitor must provide software watchdog functionality to User Space applications | P1 | Released |
| SW-LICY-015 | VH-LICY-024 | Component | Mark Hatle, Wind River | node health watchdog timeout reaction must be configurable | P1 | Released |
| SW-LICY-016 | VH-LICY-024 | Component | Christian Muck, BMW | The node health Manager shall be configurable with regards to restarting failing processes | P1 | Released |
| SW-LICY-017 | VH-LICY-024 | Component | Mark Hatle, Wind River | The node resource Manager must monitor memory consumption and initiate a configurable recovery mechanism if the complete system memory is below a configurable threshold | P1 | Released |
| SW-LICY-019 | VH-LICY-024 | Component | Fabien Hernandez | The node resource Manager must monitor the CPU load of the complete system and of individual processes | P1 | Released |
| SW-LICY-071 | VH-LICY-024 | API | David Yates, Continental | The node health Manager must provide an interface whereby Applications can register plugins for observation | P1 | Released |
| SW-LICY-070 | VH-LICY-024 | Component | Mark Hatle, Wind River | The node health Manager must provide a configurable recovery/escalation policy in case that the NHM detects an abnormal start up behavior. For instance if a safety critical application is not started correctly then the system should be restarted. However, if a user/3rd party application does not start then a restart is not necessary. | P1 | Released |
| SW-LICY-020 | VH-LICY-024 | Component | Christian Muck, BMW | The node health Manager response to System/Process failures must be configurable. System failures monitored by the Lifecycle Management will include for example exceptions, excessive resource usage and heartbeat failures. The configurable actions for the Lifecycle Management shall include ignore, restart the process and triggering a reboot of the system. This configuration will be possible at the OEM/Product level. | P1 | Released |

Munich Workshop:
Reword to be more obvious
David Yates, Continental - Definition of policies/settings required plus component design/naming should be renamed.

BMW - Definition what should be tracked? What is an abnormal start up behavior?

Original Req. NHM must react according to configured system policies in case that the NHM service/application tracker detects an abnormal start up behavior.

Manfred Bath (BMW): propose to introduce an abstract component responsible for consistency checks of applications (e.g. to detect abnormal startup), to provide proper specific input to NHM. Such functionality should not be core responsibility.
<table>
<thead>
<tr>
<th>Issue</th>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW-LICY-021</td>
<td>VH-LICY-027</td>
<td>The node health Manager must trigger a dump of debug information in case of failures.</td>
</tr>
<tr>
<td>SW-LICY-022</td>
<td>RQ 1.14.1</td>
<td>The NSC must keep track about start up of services/application</td>
</tr>
<tr>
<td>SW-LICY-023</td>
<td>VH-LICY-023</td>
<td>The NSC provide a mechanism to alter the start order of applications based on Last User Mode persistent data.</td>
</tr>
<tr>
<td>SW-LICY-066</td>
<td>VH-LICY-023</td>
<td>An API must be provided for applications to register themselves in the LUC</td>
</tr>
<tr>
<td>SW-LICY-072</td>
<td>VH-LICY-023</td>
<td>An API must be provided to read the contents of the current LUC</td>
</tr>
<tr>
<td>SW-LICY-024</td>
<td>VH-LICY-026</td>
<td>The NSC must provide a mechanism for starting and stopping applications during normal run-time</td>
</tr>
<tr>
<td>SW-LICY-067</td>
<td>VH-LICY-026</td>
<td>The Node State Manager must be able to trigger the cancelling of an ongoing shutdown</td>
</tr>
<tr>
<td>SW-LICY-025</td>
<td>VH-LICY-026</td>
<td>The NSC must ensure that critical applications are prioritized during system start-up to ensure that they are available as early as possible within the system start-up sequence</td>
</tr>
</tbody>
</table>
| SW-LICY-026 | VH-LICY-024 | Component | Mark Hatle, Wind River | The node resource Manager must provide monitoring of the RAM usage and the "Out of Memory" signal. | The NRM will monitor and report when applications fail to get more memory (i.e. they have reached their allocated max.) but will not take any further actions. It is intended that this trace output can be used by a System Integrator to configure a system as required. | P1 | Released | Munich Workshop: replace memory with ram
David Yates, Continental - needed for clarification
Christian Muck - Not needed.
Original Req. Middleware must provide monitoring of the RAM usage and the "Out of Memory" signal. |

| SW-LICY-027 | VH-LICY-024 | Component | Mark Hatle, Wind River | The node resource Manager must provide a mechanism to control the CPU usage on a process basis | P1 | Released | Munich Workshop: Replace system with CPU
David Yates, Continental - needed for clarification
Christian Muck - Reword needed for clarification
Original Req. Middleware must provide a mechanism to control the CPU usage on a process basis. |

| SW-LICY-029 | VH-LICY-002 | API | Mark Hatle, Wind River | Supply Management must be able to dispatch hardware power events to registered consumers | P1 | Released | Munich Workshop: Covered under Supply Management
Christian Muck - Not part of GENIVI lifecycle requirement.
Original Req: Lifecycle Management must be able to dispatch hardware power events to the related Lifecycle consumers. |

| SW-LICY-030 | VH-LICY-002 | API | David Yates, Continental | The Supply management API must provide a mechanism for Supply Management plug-ins to communicate power events | P1 | Released | Munich Workshop: Covered under Supply Management
Christian Muck - Not part of GENIVI lifecycle requirement.
Original Req: Lifecycle Management able to dispatch hardware power events to the Lifecycle consumer. |

| SW-LICY-031 | VH-LICY-002 | Component | David Yates, Continental | The Supply management must be able to map power events received from plug-ins to Platform Supply Management states | P1 | Released | |

<p>| SW-LICY-032 | VH-LICY-002 | Component | David Yates, Continental | The Supply management must communicate Platform Supply Events to the Node State Manager | P1 | Released | |</p>
<table>
<thead>
<tr>
<th>Document ID</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW-LICY-033</td>
<td>VH-LICY-022</td>
<td>API David Yates, Continental The Thermal Management API must provide a mechanism for Thermal Management plug-ins to communicate thermal events</td>
</tr>
<tr>
<td>SW-LICY-034</td>
<td>VH-LICY-022</td>
<td>Component David Yates, Continental The Thermal management must be able to map power events received from plug-ins to Platform Thermal Management states</td>
</tr>
<tr>
<td>SW-LICY-035</td>
<td>VH-LICY-022</td>
<td>API David Yates, Continental The Node State Manager must provide an interface with which the Thermal Manager can write the Platform Thermal State</td>
</tr>
<tr>
<td>SW-LICY-036</td>
<td>VH-LICY-001</td>
<td>API David Yates, Continental The Node State Manager must provide an API mechanism for a consumer to delay the system shutdown</td>
</tr>
<tr>
<td>SW-LICY-037</td>
<td>VH-LICY-001</td>
<td>Component David Yates, Continental The Node State Manager must provide a mechanism to configure how long a consumer can block the system shutdown</td>
</tr>
<tr>
<td>SW-LICY-069</td>
<td>VH-LICY-001</td>
<td>Component David Yates, Continental The Node State Manager must enforce that no consumer can block the system shutdown for more than the product configured time</td>
</tr>
<tr>
<td>SW-LICY-039</td>
<td>VH-LICY-003</td>
<td>Component David Yates, Continental Boot Management must provide a dynamic start-up mechanism based on the reset reason</td>
</tr>
<tr>
<td>SW-LICY-040</td>
<td>VH-LICY-023</td>
<td>API David Yates, Continental The Node State Manager must provide a public interface whereby the wakeup reason can be read</td>
</tr>
<tr>
<td>SW-LICY-041</td>
<td>VH-LICY-004</td>
<td>Component David Yates, Continental The Node State Manager must provide a configurable dynamic shutdown procedure based on the shutdown reason and node session state Based on the target configuration consumers might not be called before initiating a system shutdown</td>
</tr>
<tr>
<td>SW-LICY-042</td>
<td>VH-LICY-004</td>
<td>Component</td>
</tr>
<tr>
<td>SW-LICY-043</td>
<td>VH-LICY-004 VH-LICY-017</td>
<td>Component</td>
</tr>
<tr>
<td>SW-LICY-044</td>
<td>VH-LICY-017</td>
<td>API</td>
</tr>
<tr>
<td>SW-LICY-045</td>
<td>VH-LICY-016</td>
<td>API</td>
</tr>
<tr>
<td>SW-LICY-046</td>
<td>VH-LICY-016</td>
<td>API</td>
</tr>
<tr>
<td>SW-LICY-047</td>
<td>VH-LICY-012 VH-LICY-013 VH-LICY-014 VH-LICY-015</td>
<td>API</td>
</tr>
<tr>
<td>SW-LICY-049</td>
<td>VH-LICY-011 VH-LICY-016</td>
<td>Component</td>
</tr>
<tr>
<td>SW-LICY-050</td>
<td>VH-LICY-011</td>
<td>Component</td>
</tr>
<tr>
<td>Component</td>
<td>SW-LICY-051</td>
<td>VH-LICY-026</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Component</td>
<td>SW-LICY-052</td>
<td>VH-LICY-012</td>
</tr>
<tr>
<td>Component</td>
<td>SW-LICY-053</td>
<td>VH-LICY-018</td>
</tr>
<tr>
<td>API</td>
<td>SW-LICY-054</td>
<td>VH-LICY-018</td>
</tr>
<tr>
<td>Component</td>
<td>SW-LICY-057</td>
<td>VH-LICY-005</td>
</tr>
<tr>
<td>API</td>
<td>SW-LICY-058</td>
<td>VH-LICY-012</td>
</tr>
<tr>
<td>Component</td>
<td>SW-LICY-059</td>
<td>VH-LICY-012</td>
</tr>
<tr>
<td>Component</td>
<td>SW-LICY-060</td>
<td>VH-LICY-006</td>
</tr>
<tr>
<td>SW-LICY-062</td>
<td>VH-LICY-012</td>
<td>Component</td>
</tr>
<tr>
<td>SW-LICY-064</td>
<td>VH-LICY-013</td>
<td>API</td>
</tr>
<tr>
<td>SW-LICY-068</td>
<td>VH-LICY-013</td>
<td>Component</td>
</tr>
<tr>
<td>-</td>
<td>API</td>
<td>Christian Muck, BMW</td>
</tr>
<tr>
<td>RQ 9.1.5</td>
<td>API</td>
<td>Mark Hatle, Wind River</td>
</tr>
<tr>
<td>RQ 9.1.11</td>
<td>API</td>
<td>Mark Hatle, Wind River</td>
</tr>
<tr>
<td>RQ 9.2.3</td>
<td>API</td>
<td>Mark Hatle, Wind River</td>
</tr>
<tr>
<td>RQ 9.2.6</td>
<td>API</td>
<td>Mark Hatle, Wind River</td>
</tr>
<tr>
<td>Component</td>
<td>Fabien Hernandez</td>
<td>The Lifecycle Management must handle Out Of Memory signal to perform specific tasks</td>
</tr>
</tbody>
</table>
### Lifecycle Owned Requirements - Mechanism (1.3.1)

<table>
<thead>
<tr>
<th>Alias</th>
<th>Derived from</th>
<th>Area</th>
<th>Originator (Name, Company)</th>
<th>Description</th>
<th>Additional Info</th>
<th>State</th>
<th>Comment</th>
<th>Assigned To</th>
</tr>
</thead>
</table>

**RQ 9.1.4** Component
Mark Hatle, Wind River
Lifecycle Management must be capable of remote management of consumers
Rejected
Munich Workshop:
Rejected
David Yates, Continental - What is meant by "remote" and the controller?
Christian Muck - Not needed.

**RQ 9.1.6** Component
Mark Hatle, Wind River
Lifecycle Management should perform authentication during registration of consumers
Rejected
Munich Workshop:
Can be Rejected
What is the rc cause of this requirement?
Christian Muc - Not needed.

**RQ 9.1.7** Component
Mark Hatle, Wind River
PSM should perform authentication during registration of devices
Rejected
Munich Workshop:
Can be Rejected
What is the rc cause of this requirement?
Christian Muc - Not needed.

**RQ 9.1.12** Component
Mark Hatle, Wind River
PSM must be able to handle multiple states
Rejected
Munich Workshop:
Can be Rejected
David Yates, Continental - Requirement not a problem
Currently no definition of state available. Initi "States" prop be made by Continental at the Architects and reviewed System Level
Christian Muc - What do you is your under about this topic Mapping of m than 1 OEM's GENIVI state

**RQ 9.1.18** Component
Mark Hatle, Wind River
PSM must support remote power state management of devices
Rejected
Munich Workshop:
Can be Rejected
David Yates, Continental - meant by "rer and who is th controller?
Christian Muc - Not part of C lifecycle/PSM requirement.

**RQ 1.8.41** Component
Mark Hatle, Wind River
Multi-tasking support must provide process monitoring capabilities
Rejected
Munich Workshop:
Can be Rejected
David Yates, Continental - meant here?
Christian Muc - Not needed.
## Lifecycle Owned Requirements - Hardware (1.4.1)

<table>
<thead>
<tr>
<th>Alias</th>
<th>Area</th>
<th>Originator (Name, Company)</th>
<th>Description</th>
<th>Additional Info</th>
<th>State</th>
<th>Comment</th>
<th>Assigned To</th>
</tr>
</thead>
</table>
| Power   | Fabien Hernandez, PSA | In the event of computer unit reset, the input and output of the system are reset by default to guarantee:  
• "protected" state of input  
• "neutral" or "passive" state for output (no supply or current). | Open | David Yates, Continental : Clarification needed is this not a HW requirement, if not then more information needed | Hardware |

## Lifecycle Owned Constraints (1.1.2)

<table>
<thead>
<tr>
<th>Alias</th>
<th>Category</th>
<th>Area</th>
<th>Originator (Name, Company)</th>
<th>Description</th>
<th>Additional Info</th>
<th>State</th>
<th>Comment</th>
<th>Assigned To</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ 9.2.7</td>
<td>SW Platform</td>
<td>API</td>
<td>Mark Hatle, Wind River</td>
<td>PSM API should provide functionality to modify the set of power events an already-registered consumer is interested in</td>
<td>Agreed</td>
<td>Munich Workshop: Design Constraint Christian Muck, BMW - Not needed.</td>
<td>NSM</td>
<td></td>
</tr>
</tbody>
</table>

## Lifecycle Requirements to be satisfied outside of Lifecycle Subdomain (2.1.1)

<table>
<thead>
<tr>
<th>Alias</th>
<th>Area</th>
<th>Originator (Name, Company)</th>
<th>Description</th>
<th>Additional Info</th>
<th>State</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH-LICY-003</td>
<td>Subdomain</td>
<td>David Yates, Continental</td>
<td>The Bootloader must be able to pass the reset reason as a parameter to the Kernel</td>
<td>Reassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VH-LICY-012</td>
<td>Subdomain</td>
<td>David Yates, Continental</td>
<td>The Bootloader must be able to pass the wake-up reason as a parameter to the Kernel</td>
<td>Reassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VH-LICY-013</td>
<td>Subdomain</td>
<td>David Yates, Continental</td>
<td>The Bootloader must be able to pass the wake-up reason as a parameter to the Kernel</td>
<td>Reassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VH-LICY-014</td>
<td>Subdomain</td>
<td>David Yates, Continental</td>
<td>The Bootloader must be able to pass the wake-up reason as a parameter to the Kernel</td>
<td>Reassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ</td>
<td>Subdomain</td>
<td>Author</td>
<td>Requirement</td>
<td>Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VH-LICY-018</td>
<td>Subdomain</td>
<td>David Yates, Continental</td>
<td>The Bootloader must pass the the current application mode to dictate what components to start during the current boot phase</td>
<td>Reassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ 1.14.1</td>
<td>Subdomain</td>
<td>Mark Hatle, Wind River</td>
<td>Configuration Registry (CR) must be accessible from beginning of the Middleware start-up stage, to provide configuration settings</td>
<td>Reassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ 1.20.9</td>
<td>Subdomain</td>
<td>Mark Hatle, Wind River</td>
<td>User manager must support export of user profiles</td>
<td>Reassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ 1.20.14</td>
<td>Subdomain</td>
<td>Mark Hatle, Wind River</td>
<td>User manager must support export of user profiles</td>
<td>Reassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Christian Muck</td>
<td>It must be possible to set factory configurations for vehicle specific functions. These configuration settings (protected by an OEM specific authentication/authorization process) can't be changed by the end user.</td>
<td>Reassigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Fabien Hernandez</td>
<td>After a system reset/reboot, the user should recover its last settings or context.</td>
<td>Reassigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ 1.8.39</td>
<td>System</td>
<td>Mark Hatle, Wind River</td>
<td>Multi-tasking processes must support rights management that provides the ability to set node resource restrictions</td>
<td>Rejected</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ 10.2.1</td>
<td>System</td>
<td>Mark Hatle, Wind River</td>
<td>The boot and init sequence must be optimized and fast enough to meet the start-up performance requirements</td>
<td>Review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ 1.1.4</td>
<td>Subdomain</td>
<td>Mark Hatle, Wind River</td>
<td>Rear view camera picture must be available 5s after cold boot</td>
<td>Rejected</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ 1.16.3</td>
<td>System</td>
<td>Mark Hatle, Wind River</td>
<td>Middleware must be capable to restrict device access from User Space applications on user and group level</td>
<td>Rejected</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RQ</td>
<td>Subdomain</td>
<td>System</td>
<td>Mark Hatle, Wind River</td>
<td>Description</td>
<td>Status</td>
<td>Reassigned By</td>
</tr>
<tr>
<td>----</td>
<td>-----------</td>
<td>--------</td>
<td>-----------------------</td>
<td>-------------</td>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td>RQ 1.16.4</td>
<td>System</td>
<td>Middleware must be able to restrict access to hardware components from User Space applications on user and group level</td>
<td>Mark Hatle, Wind River</td>
<td>Rejected</td>
<td>Christian Muck, BMW</td>
<td></td>
</tr>
<tr>
<td>RQ 1.16.6</td>
<td>System</td>
<td>Middleware must be able to restrict access to the file system on user and group level</td>
<td>Mark Hatle, Wind River</td>
<td>Rejected</td>
<td>Christian Muck, BMW</td>
<td></td>
</tr>
<tr>
<td>RQ 9.1.13</td>
<td>System</td>
<td>All registered PSM consumers should acknowledge a state change request before this change will be performed</td>
<td>Mark Hatle, Wind River</td>
<td>Review</td>
<td>David Yates, Continental</td>
<td></td>
</tr>
<tr>
<td>RQ 10.1.8</td>
<td>Subdomain</td>
<td>Boot loader, in the normal mode, must load software images to make the system fully functional. It must load operating system and application software into memory and start execution</td>
<td>Mark Hatle, Wind River</td>
<td>Reassigned</td>
<td>David Yates, Continental</td>
<td></td>
</tr>
<tr>
<td>RQ 10.1.11</td>
<td>Subdomain</td>
<td>The start-up of boot loader and kernel must not interfere with the devices and services that are initialized and enabled by the HW Configuration</td>
<td>Mark Hatle, Wind River</td>
<td>Reassigned</td>
<td>David Yates, Continental</td>
<td></td>
</tr>
<tr>
<td>RQ 10.1.12</td>
<td>Subdomain</td>
<td>Boot loader must validate the MD5sum of the kernel image before running it</td>
<td>Mark Hatle, Wind River</td>
<td>Reassigned</td>
<td>David Yates, Continental</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>System</td>
<td>The system must provide an interface to allow the node health Monitor to dump debug information (stack traces, call stacks, etc.,) in the event of failures.</td>
<td>David Yates, Continental</td>
<td>Reassigned</td>
<td>Christian Muck, BMW</td>
<td></td>
</tr>
</tbody>
</table>

Christian Muck, BMW - Not needed.
Christian Muck, BMW - Rejected
Christian Muck, BMW - No activities started on boot loader yet. Pushed when such an activity starts. Christian Muck, BMW - Pushed to Sys Arch Team
Christian Muck, BMW - Boot loader topic and not part of lifecycle requirements.
Christian Muck, BMW - No activities started on OS/Kernel yet. Pushing when such an activity starts. Christian Muck, BMW - Pushed to Sys Arch Team

David Yates, Continental - Registration should be without handshake but still allow additional handshake.
Christian Muck, BMW - Understanding: the lifecycle at the automotive controller initiates the state change command is sent to the PSM set the state to all
Christian Muck, BMW - Boot loader requirement not be in lifecycle Subdomain topic and not part of life requirements.
Christian Muck, BMW - started on boot loader when such an activity starts. Christian Muck, BMW - Sys Arch Team
Christian Muck, BMW - renamed as HW Configuration. Should be split into two as well.
Christian Muck, BMW - topic and not part of life requirements.
Christian Muck, BMW - started on boot loader when such an activity starts. Christian Muck, BMW - Sys Arch Team
David Yates, Continental - Boot loader requirement not be in lifecycle Subdomain topic and not part of life requirements...but what happens if the check fails. Christian Muck, BMW - started on boot loader when such an activity starts. Christian Muck, BMW - Sys Arch Team

| RQ  | System | Mark Hatle, Wind River | The GENIVI system must provide Data Storage Quota Management on User level to limit Data Storage Usage per User. Original Req. Middleware must provide Data Storage Quota Management on User level to limit Data Storage Usage per User. | Reassigned | Christian Muck, BMW - UserMgtPersonalization

| RQ  | System | Mark Hatle, Wind River | Quota Management must be configurable to restrict user quota on each available data storage in the system | Reassigned | Christian Muck, BMW - UserMgtPersonalization

| RQ  | System | Mark Hatle, Wind River | Quota Management must support a notification mechanism to inform user space applications about reached user quota limits | Reassigned | Christian Muck, BMW - UserMgtPersonalization

| RQ  | System | Mark Hatle, Wind River | Middleware must provide user profile management capabilities. | Reassigned | David Yates, Continental - What is the definition of Middleware in context of GENIVI? Christian Muck, BMW - Understanding that user management stores cust settings. Not a PSM req Christian Muck, BMW - UserMgtPersonalization

| RQ  | System | Mark Hatle, Wind River | Operating System must support multiple users | Reassigned | Christian Muck, BMW - UserMgtPersonalization

| System | Christian Muck, BMW | The developer of an application/daemon provides information about the application dependencies. | Reassigned | Munich Workshop: Move to bottom table ar Integration Christian Muck, BMW - started on Integration yet when such an activity st Christian Muck, BMW - Sys Arch Team

| System | Fabien Hernandez | Each application launched provides maximum memory, mass storage and CPU load needed | Rejected | Munich Workshop: Duplicate to above req regards to integration te

| Component | Christian Muck, BMW | The software update must be able to flash all parts of the system. Firmware, map data, program code... Indeed it is a software loading requirement but it has some derived impact on lifecycle in terms of boot code updates. E.g. when you are executing a kernel in place (flash) and you want to update exactly that kernel you will need a derived SW platform requirement to lifecycle. Even if finally this requirement ends up in Automotive EG vehicle level space it still has some traces to lifecycle SW platform requirements and therefore should be considered. | Reassigned | Fabien Hernandez Does requirement has an imp lifecycle - it's more boot looking for RQ in relatio Lifecycle) Christian Muck, 2011-04-19: This requirement was discus SysInfraEGMinutes20110414#Vehicle level requirements Lifecycle Boje Indeed it is a softw requirement but it has s impact on lifecycle in ter code updates. E.g. whe executing a kernel in pi you want to update exa you will need a derived requirement to lifecycle. this requirement ends u Automotive EG vehicle i still has some traces to platform requirements a should be considered.) Christian Muck, BMW - Sys Arch Team
| Component | Christian Muck, BMW | When the software update fails or is interrupted the system must return to a stable state which allows a retry. | Reassigned | Fabien Hernandez Does this requirement have an impact on Lifecycle? It's more boot related (we're looking for RQ in Lifecycle)

David Yates, Continental: requirement with a SW lifecycle in the area of reliability and system state

All: Add a requirement to go always be able to state for SW

Christian Muck, BMW - Sys Arch Team |
| Component | Christian Muck, BMW | Security relevant functions must be available two seconds after start-up. For example PDC and rear view cameras. | Reassigned | If go through that kind of constraint we need also to take into account time for: audio (e.g., parking functionality), video (e.g., functionality), start-up complete system ...

David Yates, Continental: system requirement in Lifecycle Needs to be pushed into team

Danilo: Has tried this but interest/ownership in these issues

Christian Muck, BMW - started yet. Pushing when activity starts.

Christian Muck, BMW - Sys Arch Team |
| Component | Fabien Hernandez, PSA | After network wake-up, the welcome page is displayed within 1 second. Constraint? | Reassigned | David Yates, Continental: system requirement in Lifecycle

Christian Muck, BMW - started yet. Pushing when activity starts.

Christian Muck, BMW - Sys Arch Team |
| Component | Fabien Hernandez, PSA | After network wake-up, the MMI system included Visual, Audio, Phone, Video... is available in less than 4 seconds Constraint? | Reassigned | David Yates, Continental: system requirement in Lifecycle

Christian Muck, BMW - started yet. Pushing when activity starts.

Christian Muck, BMW - Sys Arch Team |
| RQ 2.1.1 | System | Mark Hatle, Wind River | Framework must be running for High-Priority audio within 1s from start-up initiation | Rejected | Christian Muck, BMW - |
| RQ 2.1.2 | System | Mark Hatle, Wind River | Framework must be running for normal audio within 3s from start-up initiation | Rejected | Christian Muck, BMW - |
| RQ 2.1.4 | System | Mark Hatle, Wind River | Framework's time of audio unavailability from system shutdown 'point-of-no-return' back to High-Priority Audio Availability must not exceed High Priority Audio Availability time constraints | Reassigned | Christian Muck, BMW - of the whole system.

Christian Muck, BMW - started yet. Pushing when activity starts.

Christian Muck, BMW - Sys Arch Team |
RQ 1.1.20 System Mark Hatle, Wind River HMI operability must be available in less then 4s Rejected Christian Muck, BMW -

RQ 1.8.1 System Mark Hatle, Wind River Must use a Linux Operating System Rejected David Yates, Continental - Not needed Christian Muck, BMW -

Component Christian Muck, BMW The system must support several graphical layers. Graphical Layers maybe used for full-screen video, HMI, browser or navigation map rendering. Reassigned Fabien Hernandez, PSA we are in the owned ch; David Yates, Continental Lifecycle Christian Muck, BMW - MediaGraphis/AudioEG Christian Muck, BMW - answer: Normally the L1 to do that.


RQ 1.1.7 Subdomain Mark Hatle, Wind River Early audio: last entertainment source must be playing after start-up within 3s Reassigned Christian Muck, BMW - of audio topic and not lifecycle requirements. Christian Muck, BMW - MediaGraphis/AudioEG Christian Muck, BMW - answer: Requirements / (now it's < 2s and not 3s)

RQ 2.1.3 System Mark Hatle, Wind River Framework must be running for High-Priority audio during system shutdown Reassigned Christian Muck, BMW - Christian Muck, BMW - MediaGraphis/AudioEG Christian Muck, BMW - answer: Requirements /

VH-LICY-025 Component Christian Muck, BMW Persistency must monitor the power interrupt line and trigger the flushing of critical data when an immediate power off occurs Review David Yates, Added bas requirements

Lifecycle Constraints to be satisfied outside of Lifecycle Subdomain (2.1.2)

<table>
<thead>
<tr>
<th>Alias</th>
<th>Area</th>
<th>Originator (Name, Company)</th>
<th>Description</th>
<th>Additional Info</th>
<th>State</th>
<th>Comment</th>
<th>Assigned To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Fabien Hernandez, PSA</td>
<td>Time between a network command (coming from network) and MMI feedback (visual, audio): &lt; 70ms</td>
<td>Rejected Fabien Hernandez, PSA. Also a constraint. David Yates, Continental: This is not Lifecycle</td>
<td>System</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>The system has to provide user feedback on user interaction within 100ms.</td>
<td>Reassigned Fabien Hernandez, PSA. It looks like a constraint. May be not only linked to lifecycle? David Yates, Continental: This is not Lifecycle Christian Muck, BMW - No activities started yet. Pushing when such an activity starts. Christian Muck, BMW - Pushed to Sys Arch Team</td>
<td>System</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Christian Muck, BMW</td>
<td>The system must provide a video watchdog functionality.</td>
<td>The video source is checked for video signal availability and current transmission.</td>
<td>Reassigned</td>
<td>Fabien Hernandez, PSA. Global system constraint? Christian Muck, BMW - No activities started yet. Pushing when such an activity starts. Christian Muck, BMW - Pushed to Sys Arch Team</td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

---

**SysInfraEGSystemd**

**systemd**

*For Beginners*

If you are new to this Expert Group start here: [SysInfraEGStarterKit](#).

**General Information**

*Short Introduction*

systemd is a system and service manager for Linux, compatible with SysV and LSB init scripts. systemd provides aggressive parallelization capabilities, uses socket and D-Bus activation for starting services, offers on-demand starting of daemons, keeps track of processes using Linux cgroups, supports snapshotting and restoring of the system state, maintains mount and automount points and implements an elaborate transactional dependency-based service control logic. It can work as a drop-in replacement for sysvinit.

It's called **systemd**, not system D or System D, or even SystemD.

<table>
<thead>
<tr>
<th>Maintainer:</th>
<th>Lennart Poettering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latest stable release:</td>
<td>189</td>
</tr>
<tr>
<td>License:</td>
<td>LGPL2.1+ (changed from GPLv2 license change commit)</td>
</tr>
<tr>
<td>EG Responsible:</td>
<td>David Yates</td>
</tr>
</tbody>
</table>
Quick Links

General links

- systemd-man - http://0pointer.de/public/systemd-man/
- systemd-Project - http://0pointer.de/blog/projects/systemd.html
- Git Web Front-end - http://cgit.freedesktop.org/systemd/
- Lennart's terse slides from linux.conf.au 2011, A video of his talk
- Mailing Lists - General Development and Discussion Mailing List
- Bug Reports - Existing Bug Reports
- IRC - #systemd on irc.freenode.org
- Follow systemd on Google+

Publications

- Control Centre: The systemd Linux init system
- Booting up: Tools and tips for systemd, a Linux init tool

systemd documentation for developers

- Part I, socket activation
- Part II, socket activation II
- On systemd-Hostname
- On systemd-timedated
- On systemd-localed
- On systemd-multiseat
- PaxControlGroups - How to behave nicely in the cgroupfs trees
- How to Write syslog Daemons Which Cooperate Nicely With systemd
- Incompatibilities with SysV/LSB
- systemd and Storage Daemons for the Root File System
- Interface Portability And Stability Chart
- systemd optimizations
- The Container Interface of systemd
- The initrd Interface of systemd

systemd documentation for admins

- Part I, verifying bootup
- Part II, Which Service Owns Which Processes
- Part III, how do I convert a sysv init script into a systemd service file
- Part IV, killing services
- Part V, the three levels of "off"
- Part VI, changing roots
- Part VII, the blame game
- Part VIII, the new configuration files
- Part IX, On /etc/sysconfig and /etc/default
- Part X, Instantiated services
- Part XI, Converting initd services
- Part XII, Securing Your Services
- Part XIII, Log and Service Status
- Part XIV, The self-explanatory boot
- Part XI, Watchdogs
### Table of contents

- systemd
  - General links
  - Publications
  - systemd documentation for developers
  - systemd documentation for admins
- Introduction
  - Detailed
    - Process identifier 1
    - Parallelizing socket services
      - Necessary for improvement
      - Solution
    - Syslog example
    - Parallelizing bus services
      - Example
    - Parallelizing services summary
    - Parallelizing file system jobs
    - Keeping the first user PID small
    - Keeping track of process
    - Activation methods
      - Unit files
    - Comparison of sysvinit, upstart and systemd
  - Video introduction
    - Lennart Poettering - Hacking Core OS (@OSEC Barcamp)
    - Video - systemd, beyond init
    - Video - Interview with Lennart
    - Slide - Open source development
  - man pages
  - Packages
  - Daemons and systemd
    - Writing and packaging system daemons
    - Reference implementation of various APIs for new-style daemons
    - Example for socket activation
    - Example of start-up notification
  - Writing systemd unit/service files
  - Installing systemd service files
    - Kernel
    - Root FS
  - Configure uboot
  - Convert SysVInitScripts to systemd service files
  - How to debug systemd problems
  - Analyzing boot-up
  - systemd commands
  - Native systemd configuration files
  - SysVInit to systemd cheatsheet
    - Services
    - Runlevels/Targets
    - Changing runlevels;
    - Figure out current runlevel
  - systemd-devel
  - systemd performance analysis
  - Optimization
  - systemd questions and answers
  - Proof of Concept (POC)
  - Sources
  - Systemd installation guides etc

### Introduction

**Detailed**

**Process identifier 1**

See [Lennart's blog story](#) for a longer introduction, and the two status updates since then. Also see the [Wikipedia article](#).

For a fast and efficient boot-up two things are crucial

- to start less
  - Starting fewer or deferring the starting of services until they are actually needed
- to start more in parallel
  - We should not serialize start-up (such as sysvinit does). Run all at the same time, so that the available CPU and disk IO bandwidth is maxed out
Most current systems that try to parallelize boot-up still synchronize the start-up of the various daemons. Example: since Avahi needs D-Bus, D-Bus is started first, and only when D-Bus signals that it is ready, Avahi is started too. This start-up synchronization results in the serialization of a significant part of the boot process.

Parallelizing socket services

Necessary for improvement

Understanding what exactly daemons require from each other and why their start-up is delayed

- Traditional UNIX daemons
  - Waiting until the socket of the other daemon offers its services and is ready for connections
  - D-Bus clients example: /var/run/dbus/system_bus_socket

Solution

systemd creates the listening sockets before the actually daemon is started and then just pass the socket to the daemon

- systemd can create all sockets for all daemons in one step
- Next step run all daemons at once

If a service needs another and the other service isn't fully started up: That's completely okay!

- The connection is queued in the providing service and the client will potentially block on that single request
- But only that one client will block and only this single request

Kernel socket buffers helps to maximize parallelization

- Ordering, buffering and synchronization requests is done by the kernel without any further management from userspace

Dependency management becomes redundant (or at least secondary)

- Daemon already started: immediately success
- Daemon is in process of being started: success (unless a synchronous request)
- Daemon is not running: success (it can be auto spawned)
- From the requester daemon-perspective: no difference

Is this kind of logic new?

- Apple's launchd idea
  - The listening of the sockets is pulled out of all daemons and is done by launchd
  - The services can all start up in parallel
  - The idea is older than launchd

- inetd
  - Listens on designated ports used by Internet services such as FTP, POP3 etc.
  - When a TCP or UDP packet arrives with a particular destination port number, inetd launches the appropriate server program to handle the connection
  - inetd focus: Internet services

Syslog example

Assumption: systemd created all sockets

- systemd starts syslog and various syslog clients all at the same time
- The messages of the clients will be added to the /dev/log socket buffer
- As long as that buffer doesn't run full, the clients will not have to wait in any way and can immediately proceed with their start-up
• As soon as syslog itself finished start-up, it will dequeue all messages and process them

**Parallelizing bus services**

Modern daemons on Linux: provide services via D-Bus instead of sockets. Apply of the same parallelizing boot logic as for traditional socket services possible?

• D-Bus already has all the right hooks for it
  • Using bus activation a service can be started the first time it is accessed

**Example**

• CUPS uses Avahi to browse for printers
  • Starting Avahi at the same time as CUPS is possible
  • If CUPS is quicker than Avahi, D-Bus queues the request until Avahi manages to establish its service name

**Parallelizing services summary**

Socket-based service activation and the bus-based service activation enables us to start all daemons in parallel, without any further synchronization.

Activation also allows us to do lazy-loading of services if a service is rarely used, we can just load it the first time, somebody accesses the socket or bus name, instead of starting it during boot.

**Parallelizing file system jobs**

Current distributions

• More synchronizations points than just daemon start-ups
  • Most prominently: file-system related jobs

On boot-up a lot of time is spent idling to wait until all devices that are listed in /etc/fstab/ show up in the device tree and are then fsck’ed, mounted,...

If that is finished we go on and boot the actual services. Improvement through autofs mount point.

• Like a connect() call shows that a service is interested in another service, an open() (or similar call) shows that a service is interested in a specific file or file-system.

Interested apps wait only if a file-system they are looking for is not yet mounted and readily available

• systemd sets up an autofs mount point and when the file-system finished fsck and quota due to normal boot-up it will be replaced by the real mount
  • While the file-system is not ready yet, the access will be queued by the kernel and the accessing process will block
  • But only that one daemon and only that one access

Integrating autofs in an init system is fragile and weird?

• Example: application tries to access an autofs file-system
  • Take very long to replace it with the real file-system, it will hang in an interruptible sleep -> you can safely cancel it
  • If the mount point should not be mountable in the end (maybe because fsck failed), autofs returns a clean error code

**Keeping the first user PID small**

**Shell is evil. Shell is fast and shell is slow**

• It is fast to hack, but slow in execution
  • Classic sysvinit boot logic is modeled around shell scripts
  • Every time commands like grep, awk, cut, sed and others are called, a process is spawned, the libraries searched, some start-up
stuff and set up an and more

- After seldom doing more than a trivial string operation the process is terminated again
- Result: incredible slow

Systemd wants to get rid off shell scripts in the boot process

- Figure out what they are currently actually used for
- Systemd perspective: quite boring things, most of the scripting is spent on trivial setup and tear-down of services and should be rewritten in C
- Either in separate executables, or moved into the daemons themselves, or simply be done in the init system

Rewrite shell scripts in C takes time, in a few case does not really make sense and sometimes shell scripts are just too handy to do without

**Keeping track of process**

Central part of a system that start up and maintains services should be process babysitting

- It should watch services
- Restart them if the shut down
- If they crash, collect information about them and keep it around for the administrator

Systemd uses cgroups

- Cgroups allow the creation of a hierarchy of groups of processes
- Cgroups let you enforce limits on entire groups of process, e.g. limit the total amount of memory or CPU
- The hierarchy is directly exposed in a virtual file-system and hence easily accessible
- The group names are basically directory names in that file-system
- If a process belonging to a specific cgroup fork(), its child will become a member of the same group
- Unless the child is privileged and has access to the cgroup file system, the child can not escape its group

**Activation methods**

- **activation on boot**
  Want to make sure a service or other unit is activated automatically on boot it is recommended to place a symlink to the unit file in the .wants/ directory of either multi-user.target or graphical.target, which are normally used as boot targets at system start-up.

- **socket-based activation**
  Special target unit sockets.target. It is recommended to place a WantedBy=sockets.target directive in the Install

- **bus-based activation**
  Example

- **device-based activation**
  It is possible to bind activation to hardware plug/unplug events.

- **path-based activation**
  Provide a way to bind service activation to file system changes

- **timer-based activation**
  Some daemons that implement clean-up jobs that are intended to be executed in regular intervals benefit from timer-based activation

**Unit files**

Systemd provides a dependency system between various entities called „Units“. Units encodes information about

- a service
- a socket
- a device
- a mount point
- an automount point
- a swap file or partition
- a start-up target
- a file system path
- a timer controlled

Units are named as their configuration files. Instead of bash scripts, systemd uses unit configuration files. Configuration file syntax is inspired by XDG Desktop Entry Specification .desktop files, which in turn are by Microsoft Windows .ini files

_TODO: More details about unit files_

**Comparison of sysvinit, upstart and systemd**


**Video introduction**

Lennart Poettering - Hacking Core OS (@OSEC Barcamp)

Lennart Poettering: "It's a very long recording (1:43h), but it's quite interesting (as I'd like to believe) and contains a bit of background where we are coming from and where are going to. Anyway, please have a look. Enjoy!"
**Video - systemd, beyond init**
Source: http://www.youtube.com/watch?v=9UnEV9SPuw8

**Video - Interview with Lennart**
Source: http://www.youtube.com/watch?v=qPiVNjUq_hU

**Slide - Open source development**
Source: https://docs.google.com/present/view?id=dD4d9j2z_1R8fjKqC7

**Booting userspace in less than 1 second**
Koen’s slides from ELC-E/T-DOSE 2011 on systemd in the embedded world

Source of video: http://www.youtube.com/watch?feature=player_embedded&v=RFVlbaDql8&noredirect=1

**man pages**
Here’s the full list of all man pages: http://0pointer.de/public/systemd-man/

**Packages**
- Fedora (Where we are: http://lists.fedoraproject.org/pipermail/devel/2010-July/138855.html)
- OpenSUSE (Instructions: http://en.opensuse.org/SDB:Systemd)
- Debian (Wiki: http://wiki.debian.org/systemd)
- Gentoo
- ArchLinux
- Ubuntu

**Daemons and systemd**

**Writing and packaging system daemons**
Read http://0pointer.de/public/systemd-man/daemon.html for
- What a daemon
- Writing new style Daemons for systemd
- Daemon activation types in systemd (socket-, bus-, device-, path-, timer- other-based activation)
- Porting existing daemons to new style systemd daemons

**Reference implementation of various APIs for new-style daemons**
- http://0pointer.de/public/systemd-man/sd_booted.html
- http://0pointer.de/public/systemd-man/sd_is_fifo.html
- http://0pointer.de/public/systemd-man/sd_listen_fds.html
- http://0pointer.de/public/systemd-man/sd_notify.html

The reference implementation from Lennart Poettering was distributed under the MIT license.

**systemd for developers**

**Example for socket activation**

General links about units, services and sockets
- Writing and packaging system daemons
- man pages
- Reference implementation of various APIs for new-style daemons
- systemd for developers

You can download a simple example project to get familiar with systemd and for quick demonstrations. Features of the project:
- compatible with systemd/none systemd systems
- socket based activation
- automatic start-up of the service on boot
- both, whatever happens first on start-up

**Example of start-up notification**

Start-up notification to signal systemd when a daemon finished starting up, use the following command (used the reference implementation of Lennart):

```c
sd_notify(0, "READY=1")
```

**Extended start-up notification**

```c
sd_notifyf(0, "READY=1\n STATUS=Processing requests...\n MAINPID=%lu", (unsigned long) getpid());
```

It's important that the "Type=notify" in the unit file.

**Writing systemd unit/service files**

Please read the section "Integration with systemd - Writing systemd service files" - http://0pointer.de/public/systemd-man/daemon.html

- http://0pointer.de/public/systemd-man/systemd.unit.html
- https://fedoraproject.org/wiki/Systemd_Packaging_Draft#Writing_unit_files

**Installing systemd service files**

**Kernel**

Because of kernel 2.6.35.9 you have to use the patch

http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commitdiff;h=676db4af043014e852f67ba0349dae0071bd11f3

```bash
cp -rpv <source> <target> # Copy your whole kernel source directory
cd <target>
vi ../Patch.systemd.patch # Delete the beginning and copy the rest of the patch to here
patch -p1 < ../Patch.systemd.patch # Include the patch (see above) if necessary
vi arch/arm/configs/<YourBoard>_defconfig # for e.g. CONFIG_QUOTA=y
export ARCH=arm
export CROSS_COMPILE=/home/<yourPath>/arm-2010.09/bin/arm-none-linux-gnueabi-
made distclean
make <YourBoard>_defconfig
make menuconfig # see https://wiki.edubuntu.org/systemd#Kernel_requirements
make -j8 uImage
cp arch/arm/boot/uImage
/tftpboot/<YourBoard>_uImage_<properName>_discovery.quota.systemd
```

**Root FS**

Start with an Ubuntu 11.10 minimal FS. All steps for this are not part of systemd.

See (Ubuntu 11.10 Oneiric) https://wiki.ubuntu.com/ARM/RootfsFromScratch
Install the systemd ECO system on your ARM SDB:

```
apt-get install intltool # 46 packages
apt-get install gperf
apt-get install libcap-dev
apt-get install pkg-config
apt-get install libudev-dev
apt-get install libdbus-1-dev
apt-get install make
```

Get the following packages from http://packages.debian.org/sid/libkmod2

```
dpkg -i libkmod2*
dpkg -i libkmod-dev*
```

Before starting ./configure of systemd, you have to install udev-181 ECO system on your ARM SDB:

```
apt-get install uuid-dev
dpkg -i libblkid1_*
dpkg -i libblkid-dev*
```


```
export TARGET=udev-181
unxz $TARGET.tar.xz
tar xvf $TARGET.tar
cd $TARGET
./configure --disable-introspection \
  --disable-gudev
make -j4
sudo make install
```

Get newest systemd from http://www.freedesktop.org/software/systemd (here version 43)
export TARGET=systemd-43
unxz $TARGET.tar.xz
tar xvf $TARGET.tar
cd $TARGET
./configure --localstatedir=/var 
   --disable-manpages 
   --with-dbuspolicydir=/etc/dbus-1/system.d 
   --with-dbussessionservicedir=/usr/share/dbus-1/services 
   --with-dbussystemservicedir=/usr/share/dbus-1/system-services 
   --with-dbussysteminterface=/usr/share/dbus-1/interfaces
make
sudo make install

Configure uboot
If your uboot from SD card looks similar like (pri):

bootargs=console=ttySC0,115200n8n rootwait rw noinitrd root=/dev/mmcblk0p2
ip=143.103.22.48

You have to add: init=/usr/lib/systemd/systemd

set bootargs console=ttySC0,115200n8n rootwait rw noinitrd
init=/usr/lib/systemd/systemd root=/dev/mmcblk0p2 ip=143.103.22.48
sav

Please read the section "Integration with systemd - Installing systemd service files" - http://0pointer.de/public/systemd-man/daemon.html

Convert SysVinitScripts to systemd service files
Please read Lennart's blog Part III, how do i convert a sysv init script into a systemd service file

How to debug systemd problems

Analyzing boot-up

- Change the log level of systemd - Verbose mode on
  Modify /etc/systemd/system.conf to increase verbosity. Example:

  LogLevel=debug            <--- Uncomment this line and use *debug* (default: commented and *info*)
  LogTarget=syslog-or-kmsg  <--- Uncomment this line (default: commented)
  SysVConsole=yes           <--- Uncomment this line (default: commented)

  (Change the log level to debug has an impact on your start-up performance)

- systemd writes automatically a log message with the time it needed to syslog/kmsg when it finished booting up and sends a broadcast message (D-Bus). Example:

  systemd[1]: Start-up finished in 2s 65ms 924us (kernel) + 2s 828ms 195us (initrd) + 11s 900ms 471us (userspace) = 16s 794ms 590us.

- Please read http://0pointer.de/blog/projects/blame-game.html to avoid misunderstanding of the command "systemd-analyze blame":

  systemd-analyze blame
Example:

```bash
$ systemd-analyze blame
6207ms udev-settle.service
5228ms cryptsetup@luks\x2d9899b85d\x2df790\x2d4d2a\x2da650\x2d8b7d2fb92cc3.service
735ms NetworkManager.service
642ms avahi-daemon.service
600ms abrtd.service
517ms rtkit-daemon.service
478ms fedora-storage-init.service
396ms dbus.service
390ms rpcidmapd.service
346ms systemd-tmpfiles-setup.service
322ms fedora-sysinit-unhack.service
316ms cups.service
310ms console-kit-log-system-start.service
309ms libvirtd.service
303ms rpcbind.service
298ms ksmtuned.service
288ms lvm2-monitor.service
281ms rpcgssd.service
277ms sshd.service
276ms livesys.service
267ms iscsid.service
236ms mdmonitor.service
234ms nfslock.service
223ms ksm.service
218ms mcelog.service
...```

- systemd-analyze plot shows more high-level data: which service took how much time to initialize, and what needed to wait for it. Example:

  ```bash
  systemctl dot | dot -Tsvg > systemd.svg
  ```

- Creates a dependency graph of the units. Example:

  ```bash
  systemctl plot > plot.svg
eog plot.svg
  ```

**systemd commands**

Control the systemd system and service manager
http://0pointer.de/public/systemd-man/systemctl.html

Analyze system boot-up performance
http://0pointer.de/public/systemd-man/systemd-analyze.html

Query the systemd journal
http://0pointer.de/public/systemd-man/journalctl.html

Special journal fields
http://0pointer.de/public/systemd-man/systemd.journal-fields.html

**Native systemd configuration files**

- Add a hostname
  Example:

  ```
  File: /etc/hostname
  MyHostname
  ```

- Console and keymap settings
  The /etc/vconsole.conf file configures the virtual console, i.e. keyboard mapping and console font. Example:
File: /etc/vconsole.conf

- OS info
  /etc/os-release contains data that is defined by the operating system vendor and should not be changed by the administrator.
  Example:

  ```
  File: /etc/os-release
  
  NAME=Archlinux
  ID=arch
  PRETTY_NAME=Arch GNU/Linux
  ANSI_COLOR=1;34
  ```

- Locale settings (read man locale.conf for more options)
  Example:

  ```
  File: /etc/locale.conf
  
  LANG=en_US.utf8
  LC_COLLATE=C
  ```

- Configure kernel modules to load during boot
  systemd uses /etc/modules-load.d/ to configure kernel modules to load during boot in a static list. Each configuration file is named in the style of /etc/modules-load.d/<program>.conf. The configuration files should simply contain a list of kernel module names to load, separated by newlines. Empty lines and lines whose first non-whitespace character is # or ; are ignored. Example:

  ```
  File: /etc/modules-load.d/virtio-net.conf
  
  # Load virtio-net.ko at boot
  virtio-net
  ```

- Configure kernel modules blacklist
  systemd uses /etc/modprobe.d/ to configure kernel modules blacklist. Each configuration file is named in the style of /etc/modprobe.d/<program>.conf. Empty lines and lines whose first non-whitespace character is # or ; are ignored. Example:

  ```
  File: /etc/modprobe.d/snd_hda_intel
  
  blacklist snd_hda_intel
  ```

### SysVinit to systemd cheatsheet

Source: [https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet](https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet)

### Services

<table>
<thead>
<tr>
<th>sysvinit Command</th>
<th>systemd Command</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>service frobozz start</td>
<td>systemctl start frobozz.service</td>
<td>Used to start a service (not reboot persistent)</td>
</tr>
<tr>
<td>service frobozz stop</td>
<td>systemctl stop frobozz.service</td>
<td>Used to stop a service (not reboot persistent)</td>
</tr>
<tr>
<td>service frobozz restart</td>
<td>systemctl restart frobozz.service</td>
<td>Used to stop and then start a service</td>
</tr>
<tr>
<td>service frobozz reload</td>
<td>systemctl reload frobozz.service</td>
<td>When supported, reloads the config file without interrupting pending operations.</td>
</tr>
<tr>
<td>service frobozz condrestart</td>
<td>systemctl condrestart frobozz.service</td>
<td>Restarts if the service is already running.</td>
</tr>
<tr>
<td>service frobozz status</td>
<td>systemctl status frobozz.service</td>
<td>Tells whether a service is currently running.</td>
</tr>
</tbody>
</table>
### Runlevels/Targets

Source: [https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet](https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet)

<table>
<thead>
<tr>
<th>sysvinit Runlevel</th>
<th>systemd Target</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>runlevel0.target, poweroff.target</td>
<td>Halt the system.</td>
</tr>
<tr>
<td>1, s, single</td>
<td>runlevel1.target, rescue.target</td>
<td>Single user mode.</td>
</tr>
<tr>
<td>2, 4</td>
<td>runlevel2.target, runlevel4.target, multi-user.target</td>
<td>User-defined/Site-specific runlevels. By default, identical to 3.</td>
</tr>
<tr>
<td>3</td>
<td>runlevel3.target, multi-user.target</td>
<td>Multi-user, non-graphical. Users can usually login via multiple consoles or via the network.</td>
</tr>
<tr>
<td>5</td>
<td>runlevel5.target, graphical.target</td>
<td>Multi-user, graphical. Usually has all the services of runlevel 3 plus a graphical login.</td>
</tr>
<tr>
<td>6</td>
<td>runlevel6.target, reboot.target</td>
<td>Reboot</td>
</tr>
<tr>
<td>emergency</td>
<td>emergency.target</td>
<td>Emergency shell</td>
</tr>
</tbody>
</table>

### Changing runlevels:

<table>
<thead>
<tr>
<th>sysvinit Command</th>
<th>systemd Command</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>telinit 3</td>
<td>systemctl isolate multi-user.target (OR systemctl isolate runlevel3.target OR telinit 3)</td>
<td>Change to multi-user run level but has no effect on the next boot.</td>
</tr>
<tr>
<td>sed s/^id:.*:initdefault:/id:3:initdefault:/</td>
<td>In -sf /lib/systemd/system/multi-user.target /etc/systemd/system/default.target</td>
<td>Set to use multi-user runlevel on next reboot</td>
</tr>
</tbody>
</table>

### Figure out current runlevel

Note that there might be more than one target active at the same time. So the question regarding the runlevel might not always make sense. Here's how you would figure out all targets that are currently active:

```
$ systemctl list-units --type=target
```

If you are just interested in a single number, you can use the venerable runlevel command, but again, its output might be misleading.

### systemd-devel

The following table gives a short overview what's missing in systemd for using in GENIVI, e.g. missing functionality or patches.

<table>
<thead>
<tr>
<th>Area</th>
<th>Requirement</th>
<th>Patch Status</th>
<th>Thread</th>
<th>Short description of patch</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHM</td>
<td>LC_VEH_024</td>
<td>Contributed outside of GENIVI</td>
<td>restart patch 1</td>
<td>In case a service failed to start properly (it exited with a code not 0 - not successful) it has to be retried, to be restarted a predefined number of times, until it ends successfully or until the predefined number of restarts is reached. For that, there is the option &quot;MaxRestartRetries&quot; which accepts an integer = number of restarts. This applies only to &quot;.service&quot; units.</td>
<td></td>
</tr>
</tbody>
</table>
### systemd performance analysis

This section will be used to detail performance results achieved from tests using systemd.

### Optimization


< TODO: Insert examples >

### systemd questions and answers

This section will be used to detail questions and answers with regards to systemd. The table can be used by anyone to add questions/answers for systemd issues.

<table>
<thead>
<tr>
<th>Area</th>
<th>Originator</th>
<th>Question</th>
<th>State</th>
<th>Comments/Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>David Yates</td>
<td>Which system lib is planned to be used finally in GENIVI? i.e. uclibc, glibc or...</td>
<td>Open</td>
<td>The reason this is important for systemd is that it has a large impact on the system start-up as systemd is dependent on this library and must load this before starting anything else.</td>
</tr>
<tr>
<td>Dependencies</td>
<td>Christian Muck</td>
<td>When does systemd exactly resolve the dependencies between units and do we have access(reading) to the dependency graph?</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>Dependencies</td>
<td>David Yates</td>
<td>Is it possible for systemd to dynamically alter the start-up order (dependency graph) during the start-up based on run time events</td>
<td>Closed</td>
<td>Lennart, AMM Dublin</td>
</tr>
<tr>
<td>Dependencies</td>
<td></td>
<td></td>
<td></td>
<td>This could be handled within systemd via CPU prioritization however this functionality is not currently available. This can be added to the systemd road map but this will not be a short term delivery</td>
</tr>
<tr>
<td>Dependencies</td>
<td></td>
<td>How can user interaction being communicated to systemd?</td>
<td>Closed</td>
<td>Lennart, AMM Dublin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>It is just done by starting a service by e.g. a socket request. David, AMM Dublin E.g. the HMI could send a d-bus message to tuner and this would trigger priority change.</td>
</tr>
<tr>
<td>Dependencies</td>
<td></td>
<td>Is the cancellation of an ongoing start-up procedure possible?</td>
<td>Closed</td>
<td>Lennart, AMM Dublin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>With systemd there is no start-up phase any longer from systemd standpoint because we have bus activation. So if you want to cancel a service you can send a signal. So if you write your services right the service can be terminated at any time</td>
</tr>
<tr>
<td>Dependencies</td>
<td>How is the shutdown with systemd realized?</td>
<td>Closed</td>
<td>Lennart, AMM Dublin</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------</td>
<td>--------</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In systemd there is no shutdown. So it works like that you just have a target conflicting with everything else so everything first gets closed down and then the system is shut down (target is called shutdown). So if you use this and adopt it you can control which services should go down and which just should keep on until the shut down is realized. So what systemd does is that it kills all the processes. You could just change the description file and change the shutdown signal to SIGHUP or something and react in your service adequately.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Communication</th>
<th>Is it possible that a service can send something back to systemd informing that shutdown is finished?</th>
<th>Closed</th>
<th>Lennart, AMM Dublin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>What systemd should not be is a generic message broadcast service. So this does sound like you probably use something else. This reminds me to something we wanted to have for suspend on desktops so that something can be delayed. So this is not implemented but we were thinking also about to have a very high level module doing this kind of stuff. The question is if we can get this in the next cycle done.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Units</th>
<th>Torsten Hildebrand</th>
<th>Can you talk about snapshots and systemd instances</th>
<th>Closed</th>
<th>Lennart, AMM Dublin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Snapshots are dynamic and can be taken at any time. You can store it in memory under a name, do any change on your system and afterwards return to the snapshot. You guys wanted to have this on disk. So this would be a little fit and convert the snapshots to a target and seems to be a good feature so I will add this. And I just implemented a feature that you can have an option that you can ignore a service within a snapshot. It is not dynamically. You can also create units on the fly and solve the problem by that. These are called generators. So generators run very early in the boot process. So you can get different systemd instances like system-systemd and user-systemd. So it is already implemented to run the systemd as a user. So what we want to do is make systemd responsible for user logs so that we can exchange the proprietary solutions with a generic one. So this user systemd would just run as a normal service from the view of the system systemd. So they would be completely separate. This would also fit to automotive because you also need user management.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>Applications should change their behavior in different start modes</th>
<th>Closed</th>
<th>Lennart, AMM Dublin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>So you have several options: you can use generators, you can create several service configurations, you can use instantiated services by having one unit file and create multiple instances so you can have variables in the service file and pass your variable as parameters to create real instances.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Proof of Concept (POC)**

To verify the currently proposed usage of systemd within the GENIVI consortium it is proposed that a Proof of Concept (POC) is created to validate the possible startup times when using systemd.

This POC will be detailed on this sub-page along with startup figures when they are available.

**Sources**

The content of this page is a collection of various wiki pages and Lennart’s blog.

**Systemd installation guides etc**

- Installing an Ubuntu 11.10 based file system with systemd on Freescale i.MX53 QSB
- Installing an Ubuntu 11.10 based file system with systemd in a Virtual Machine

**SysInfraEGSystemdPOC**

**systemd POC**
Introduction

As part of the proposal to use systemd within the GENIVI consortium as defined by the Lifecycle Management architecture it is proposed that a Proof of Concept (POC) is created.

This will, as far as possible, validate the expected system start-up times with defined measurement points.

All interested GENIVI Members are free to create their own POC and add test configurations and figures to this page. It is proposed that this page details the standard measurement points and provides a location for any tests results and descriptions.

Obviously it is intended to only record information for POC's where optimized systemd configurations are in place as opposed to standard distributions, i.e. Fedora 15.

Proposed Measurement Points

The following measurement points should be recorded:

- Network Initialization - CAN/MOST initialization and transmission/reception of a message
- Welcome Screen - Display of a static screen on the display
- First Audio - This is the point where we can output any audible source (mimics the starting of the Park Distance Control (PDC))
- First Video - This is the point where a video source can be displayed on the screen (mimics the displaying of the Rear View Camera (RVC))
- BASE_RUNNING - This is the point where all mandatory components (base & early features are started)
- LUC_RUNNING - This is the point where components required for the Last user Context are up and running (for this POC it is proposed that the LUC requires three applications to be started)
- Shell - This is the point where the shell is available for first input
- FULLY_RUNNING - This is the point where the start-up is complete, i.e. we have started all components required in the system

For an explanation of the systemd concept and the contents of some of these measurement points please look here.

NOTE: The contents of the point FULLY_RUNNING can not be completely defined as it will always be dependent on the product, however it is measured to simply get a feeling for how much more in the system needs to be started from the previous boot phase.

Continental systemd POC

A proof of concept was created based on the i.MX53 Quick Start board. This was presented at the San Jose AMM and the following use case definition details the start-up process and the timings achieved.
The unit files used in the POC are attached to this page. The root directory of the tar file is "systemd" and was located in the directory /lib.