Persistence Management Overview

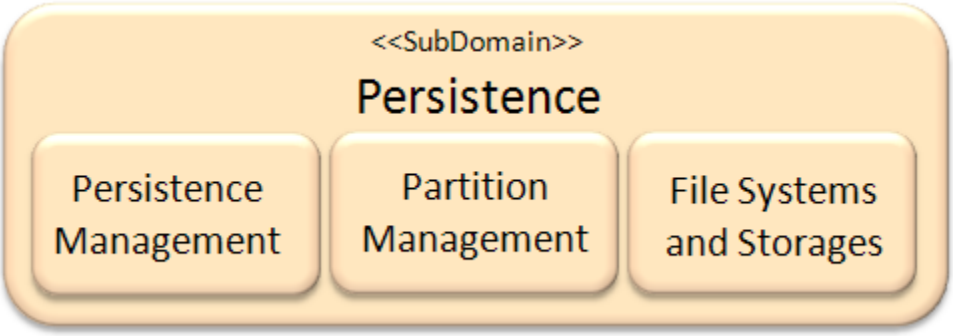
Scope of Work Item

Persistence is about storing data permanently, to manage FLASH constraints and to manage persistency within automotive lifecycle constraints.

The scope of this work package is a full development activity of Persistence. 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 Persistence system architecture specification parts will be located in the GENIVI Enterprise Architecture model. The software and the design specification you can find in the implementation model section of the GENIVI model.

The Functional Scope of Persistence



In Scope

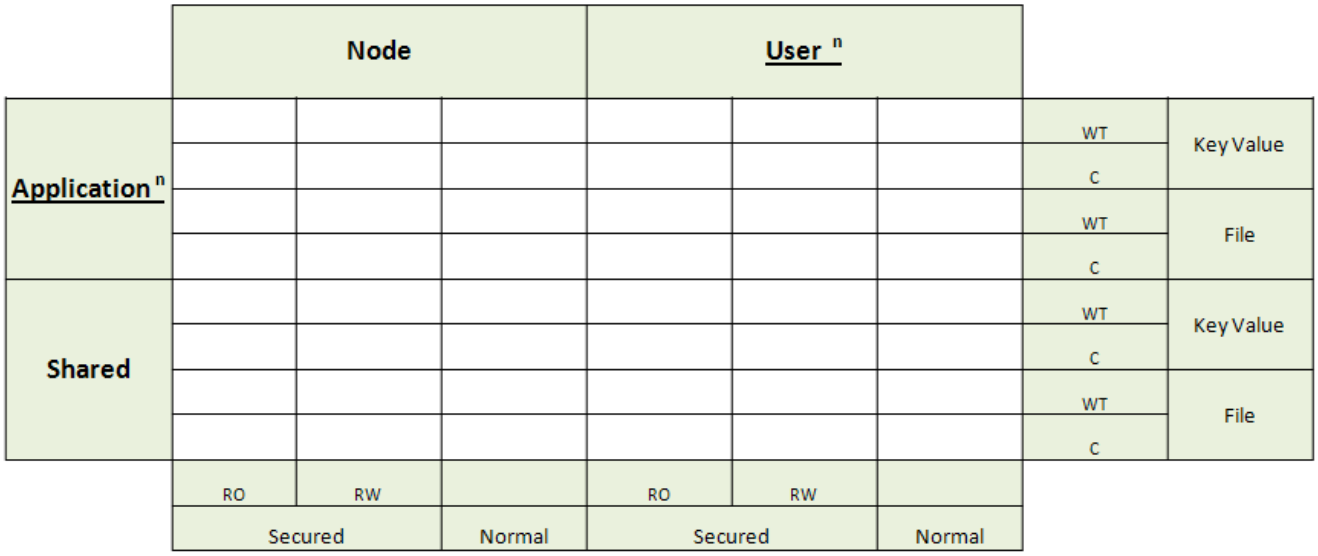
- Persistence Management
  - Key/Value Pair Management
  - File\- and Folder Structure Management
  - User Data Management
  - Shared Data Management
- File Systems and Storages
  - Flash storages (focus NAND)
  - Persistence plug-in for Lifecyle (Health monitoring and repair)

Not in Scope

- Partition Management
- File Systems and Storages
  - other storages (except NAND flash)

Which or what type of bits and bytes inside flash memory are in scope of Persistence?

Because there is not a simple answer this page gives you an introduction of Persistence Data.
... but finally we end up that also data items which are not code should be handled by Persistence. To summarize the characteristics of this data this table is created:



The table need to be read in the following way:

- The "specific" matrix
  - Node: Node specific data (not user related)
  - User: User specific data
  - Application: Application specific data (neither shared within a group nor public)
  - Shared: Shared data within a group or public
- The "policy" matrix (that is the level on which the application will work with the data and when it is persist)
  - File
  - Key - Value
  - WT: Write Through (immediately persist, synchronously)
  - C: Cached data in the RAM (will be stored/persist in flash in the shutdown phase of the system handled by persistence independently)
- The "right and availability" matrix
  - Normal (typical character of a filesystem or database)
  - Secure (very early available <1s and specific security for access/rights). This type of data can only be written by the software loading procedure or by diagnostics.

Now the task is to find for each white cell a solution and define what GENIVI is covering and what is still needed to be be implemented by the Trier 1s or the OSVs.

In the Architecture Documentation section below the answers are given, but before we go into that some additional information about Persistence is provided...

Issues about "Persistent data"

Reliability

The problem with FLASH (NOR and/or NAND) devices is linked to reliability of semiconductor storage devices. In particular flash devices are sensitive to voltage drops during write cycles; reliability of flash devices heavily depends on the device driver, but also on the applications.

Lifetime

Managing persistent data is to avoid storing data during runtime in order to increase the lifetime of the FLASH device, because the total number of program-erase cycles per block is typically limited. Most available FLASH memory devices are guaranteed to have around 100,000 cycles, before the wear begins to deteriorate the integrity of the storage. The number of program-erase cycles can be extended by a technique which is called wear leveling. Wear leveling can be done in software or it is already integrated in the hardware of the flash storing device. The FLASH storing device must work properly at least during the lifetime of a car which is at minimum 10 years.

Exemplary estimation of the number of required program-erase cycles:

Mileage of a car per lifetime: 200.000km Average mileage of one lifecycle: 10km Number of lifecycles per lifetime: 20.000 Max program-erase cycles per lifecycle: 5

Defragmentation

Because of the physical limitation of manageable size (write-able size -> page, erase-able size -> block (NAND) or sector (NOR)) defragmentation is a known issue of Persistence data. Different solutions need to be offered to avoid forcing defragmentation already from concept point of view right from the beginning.

Availability during Start-up of the node/system

Typically to grant reliability persistent data must be checked before use it; this check can be a time-consuming process. But this time cannot be accepted for IVI system. Therefore more enhanced concepts are needed. Considering that persistency storage medium may have long start-up times (e.g. Hard Disks) the applications consuming persistent data must be informed when data can be accessed.

Getting everything stored in shutdown phase

To get a consistent system snapshot, power off must be delayed until all applications have their persistent data written. After all persistent data was written, the filesystem must be unmounted to ensure that the flash device is not accessed anymore. Unmounting the filesystem may also be a time consuming process, since it may include erase and/or write operations.