Persistence Subsystem

2015.10.21 15:00 to 16:00

Ingo Huerner
Persistence Topic Lead (EG-SI Expert Group)
Mentor Graphics

Agenda

- Introduction
- Persistence Subsystem Overview
- Basic Concepts for Persistence Architecture
- Persistence Common Object
- Setup of Persistence Data
- Guidelines
- Open Source Project Information

Introduction – Problems to solve

The problem to solve

- Provide a mechanism to application to load and store persistent data
- Guarantee the memory device works correctly the complete lifetime of the IVI system
- Provide a solution for everybody
  - Including OSS and legacy components

Introduction

Why do we invent something new

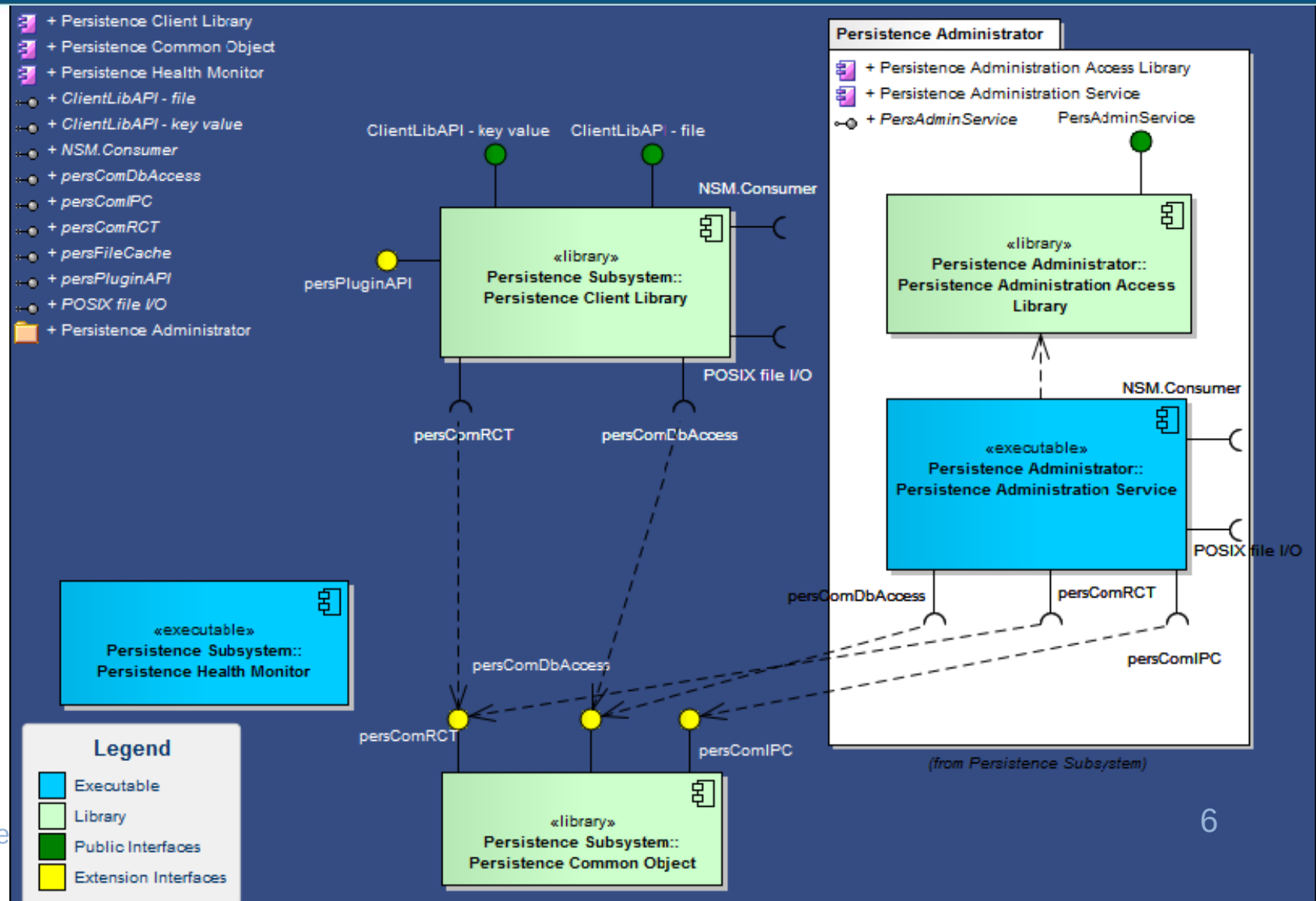
- Automotive requirements
  - System startup ➔ early data like LUC
  - System shutdown ➔ normal/fast
- Security issues
- Simple and easy to use interface
- Extendable
  - Plugin API to implement different storage backends

Introduction - Reliability

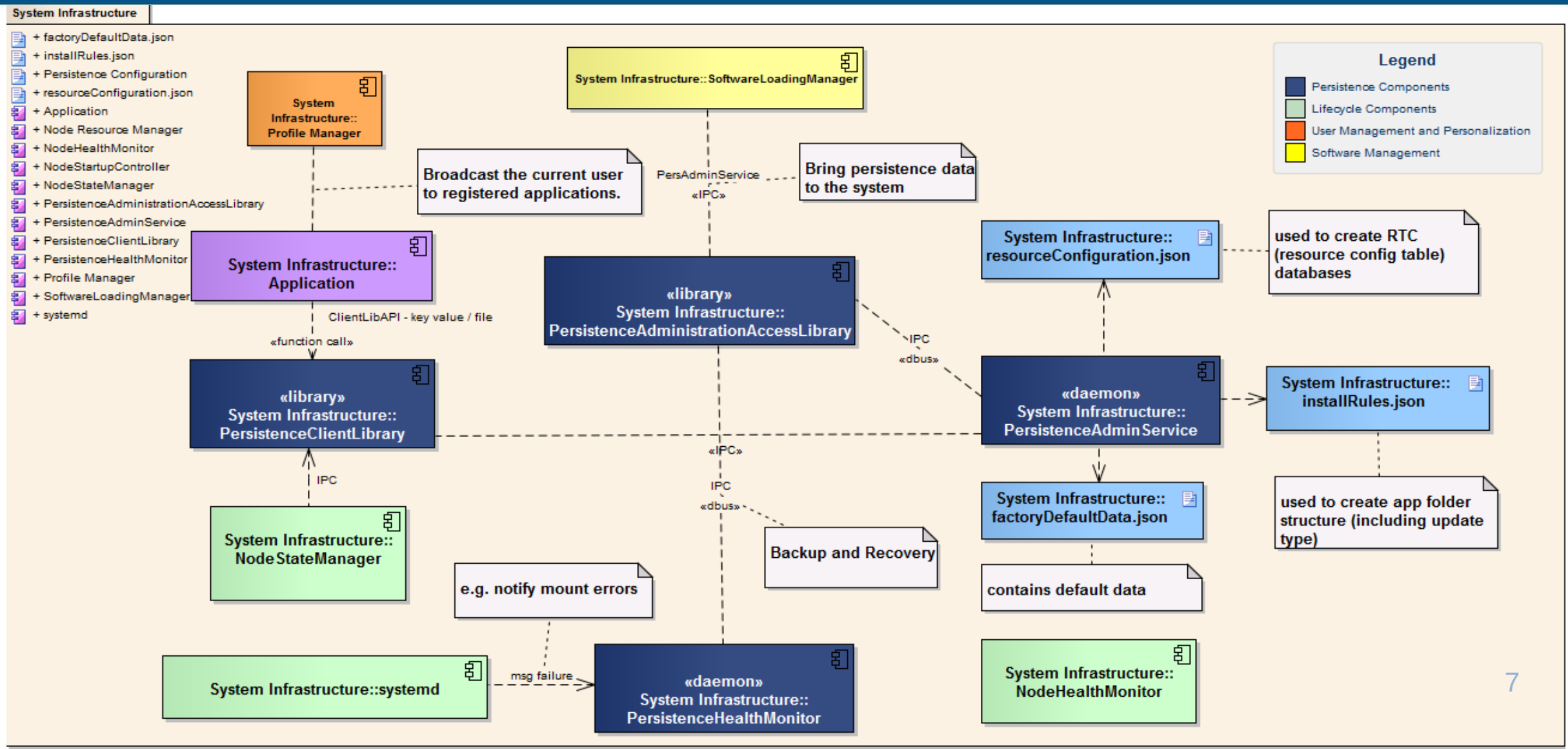
- Power cut‘s
  - Data must be stored power fails save
- Flash memory issues
  - Limited program erase cycle (max 100.000 times)
  - Flash is organized in erase blocks (8 to 16 MB)
    - First the complete block will be erased
    - Data will be written
    - ➔ write data sequentially matching the size of the erase blocks

Components – Persistence Context

- Client Library
- Administration Service
- Common Object
- Health Monitor



Components – System Context



Components – System Context

- Persistence Client Library / Persistence Admin Service
  - Use Persistence Common Object (database access)
- Application (e.g. Tuner/Navigation/MediaPlayer)
  - Use Persistence Client Library (read/write data)
- Software Loading
  - Use Persistence Administration Service (setup data)
- „systemd“
  - Sends error notifications Persistence Health Monitor

Components – Persistence Common Object

Key-value store backend

- Used open source database kissdb as basis
  - https://github.com/zerotier/kissdb
- Added features like
  - Data caching
  - Shared access
  - Backup and recovery mechanism
- By default Itazm/C backend is still used
  - Use configure with „--with-database=key-value-store”

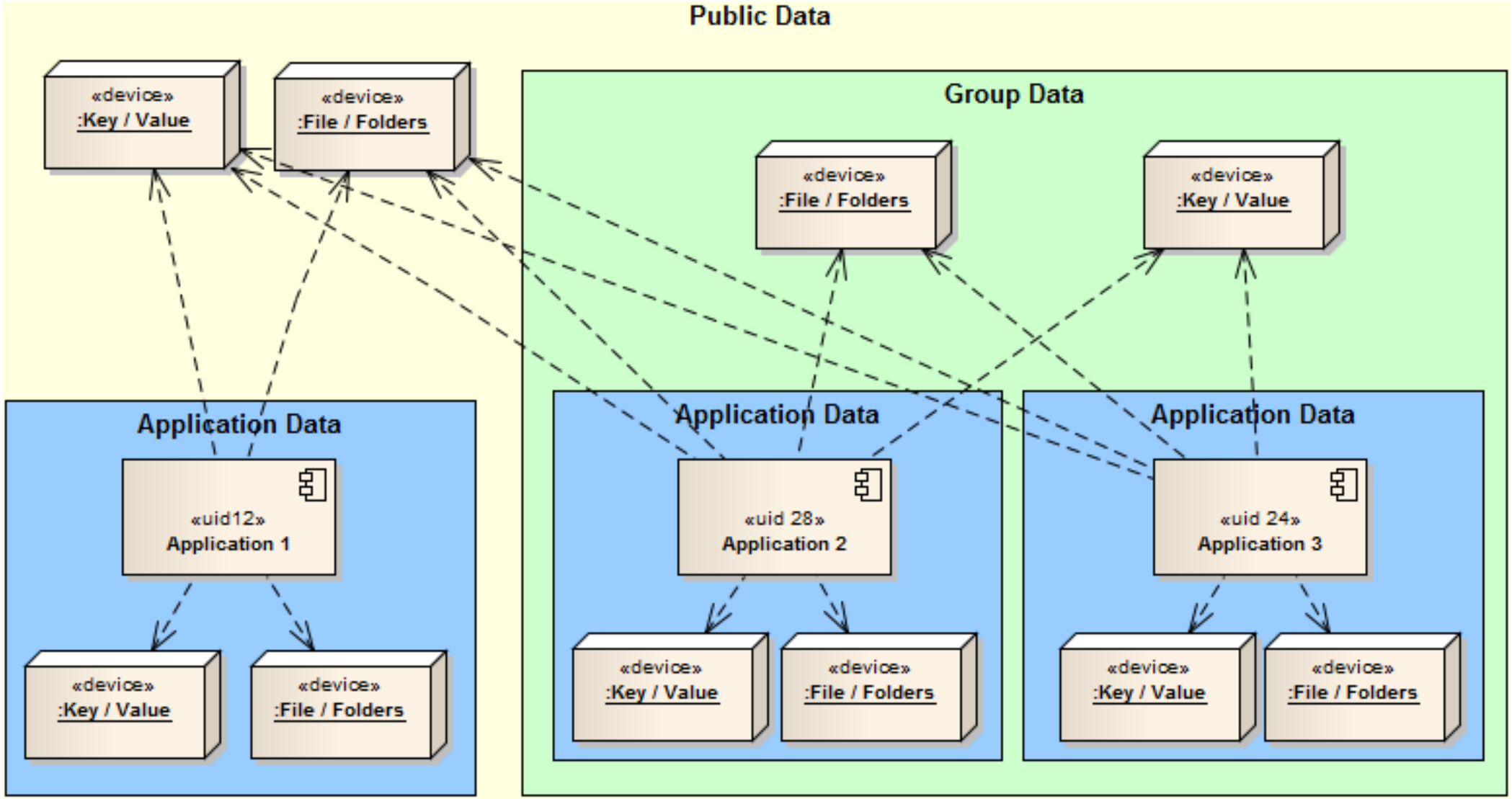
Components – Interface Examples

- Persistence Client Library (C-API)
  - int **pclKeyWriteData**(unsigned int **ldbid**, const char* **resource_id**, unsigned int **user_no**, unsigned char **seat_no**, unsigned char* **buf**, unsigned long **size**)
- Persistence Administrator Service (Client Library C-API)
  - **persAdminDataBackupRecovery**(PersASSelectionType_e **type**, const char* **backup_name**, const char* **applicationID**, unsigned int **user_no**, unsigned int **seat_no**)

Concept – Data Separation

- **Local data**
  - Access is limited to the application itself
- **Group shared data**
  - Shared by a group of applications
- **Public shared data**
  - Shared by any application within the system
- Data will be separated as it is stored in different databases

Concept – Data Separation



Concept - Data Separation

- Provides the possibility to apply access control
- Each app has a different Linux user ID
- Local user data
  - Only the application has access
- Public or group data
  - Public data, everybody is in this group
  - Only some applications are in this group
  - Every member has read access
  - One “master” that has read/write access

Concept – Resource Configuration Table

- Central place of data configuration
    - Application doesn‘t need to care about
    - System integrator needs to setup this table
- Configuration files using JSON
- Security reasons
  - Resource can only be used in the configured way
  - E.g. a resource is configured as read-only, it can’t be modified.

Setup Persistence Data - PAS

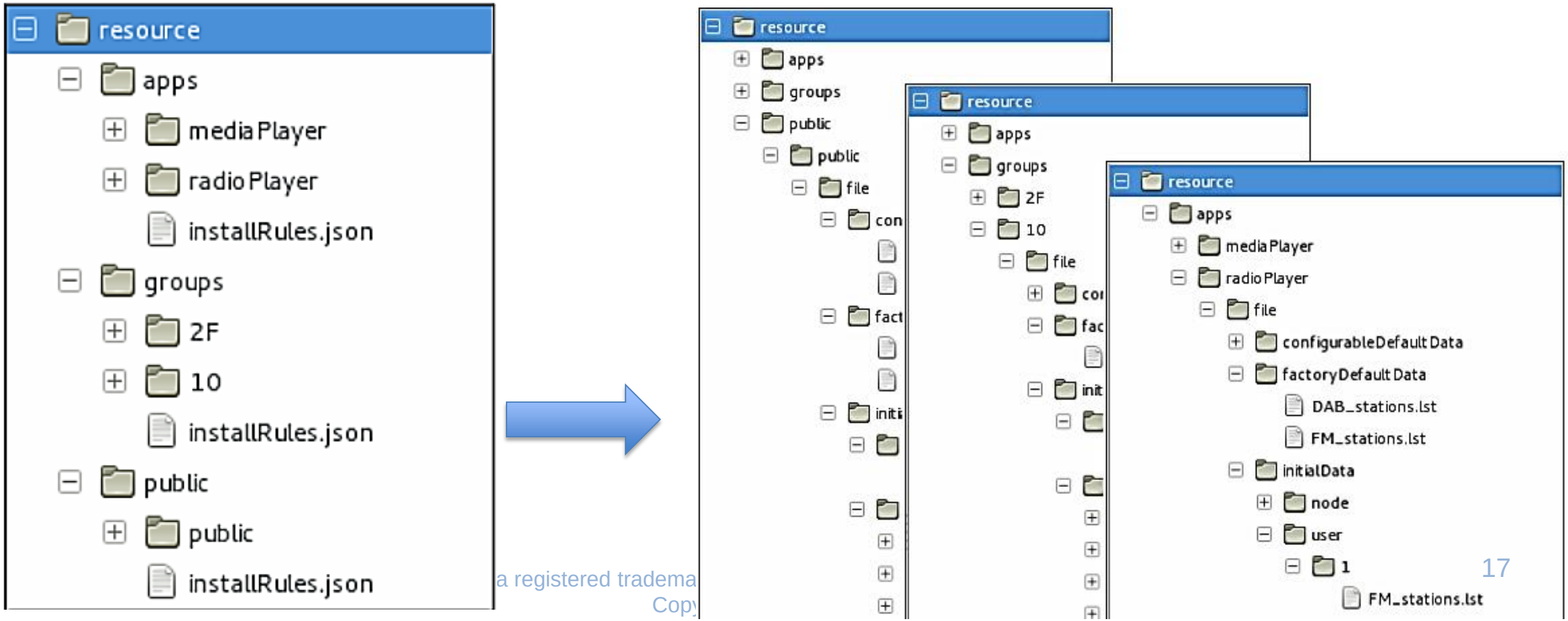
- Create default application folders including links to shared data
- Deploy the default content
- Create local database for each application
- Create shared databases
- Provide application specific links to shared databases
  - (group/ public)
- Setup of application file system access policies
- Delete, copy, backup and restore files (files and databases)

Setup Persistence Data - Scope

- Data organization of installation medium is similar to the internal persistence file system structure
- Format used for installation should be flexible
  - Installation of new application data
  - Update/uninstall of application data (as whole)
  - Install/update/uninstall of individual resources
  - Configuration of single or many apps
  - Partial updates of resources using masks (key types)

Setup Persistence Data - Organization

Update medium: 3 different folders (app/group/public)



Setup Persistence Data – Application Configuration

JSON rule for apps

- Type of installation
  - New install
  - Uninstall
  - Update defaults
  - Skip factory defaults
  - Skip config defaults

Format:

```
{
  "[APPLICATIONNAME]":"[RULE]",
  …
}
```

Example:

```
{
  "Navigation":"PersAdminCfgInstallRules_NewInstall",
  "AppX":"PersAdminCfgInstallRules_Uninstall",
}
```

Setup the persistence data – Resource configuration

JSON rule for RCT

Resource

Configuration

Table

```
{
  "config_appl" : “[APPLICATIONNAME]",
  "version" : “[VERSION]",
  "resources" : {
  “[ENTRYNAME]" : {
  "policy" : “[POLICY]",
  "permission" : “[PERMISSION]",
  "storage" : “[STORAGE]",
  “type”:”[TYPE]”,
  "max_size" : “[MAXSIZE]",
  "responsible" : “[APPLICATIONNAME]",
  “customPlugin” : [PLUGINNAME],
  "customID" : “[HASHVALUE]“
}
```

Setup the persistence data – Resource Configuration

RCT configuration example

```
{  "config_appl":“Navigation",
  "version":"0.1.0",
  "resources":  {
   “last_position":   {
      "policy":"cached",          ➔ Cached/write through
      "permission":"RW",          ➔ Read only or read/write
      "storage":"local",          ➔ Local/shared or custom
      "max_size":"2048",          ➔ max size of the content
      "responsible":"Navigation",➔ Responsible application
      "custom_name":"na",         ➔ if custom storage type
       "type":"key",              ➔ Key or file resource
       "customID":"edf1bc", },
```

Setup Persistence Data – Default Data

JSON rule data

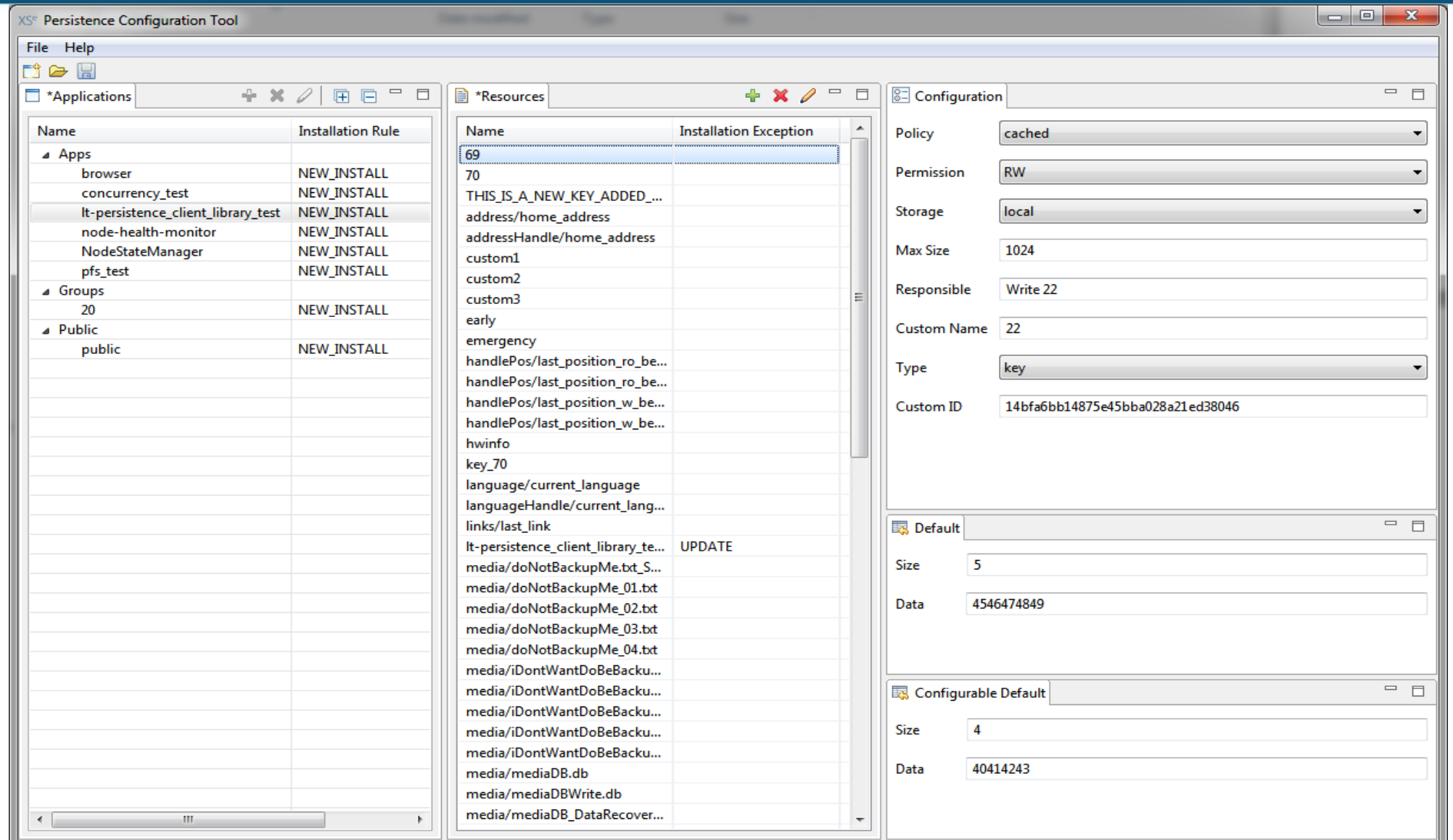
- Factory default data
- Configurable default data

```
{
"config_appl" : “[APPLICATIONNAME]",
"version" : “[VERSION]",
"resources" :
  {
    “[ENTRYNAME]" :
     {
        “size" : “[SIZE]",
        “data" : “[DATA]",
      },
```

Setup Persistence Data – Configuration Tool

- For larger projects handling the JSON files is getting complicated
- Provided a GUI tool to easily add and modify entries
- Tool is Eclipse based
- Will be soon released as an open GENIVI project

Setup Persistence Data – Configuration Tool



Guidelines - Persistence Usability Rules

Why different usability rules

- Only one solution may not fit all needs
- Legacy or OSS components must be integrated
- Intended to be used when storing data directly to files
- Extension of the file API
- A way to integrate other database engines

Guidelines - Persistence Usability Rules

- **Complete integrated usage**
  - Use persistence file API functions
  - Persistence takes care about backup and recovery
- **Partially intgrated usage**
  - Store data in the location persistence provides
  - Backup/recovery is in the responsibility of the application
- **Free usage**
  - Store data weherever you want in persistence folders
  - Persistence takes no responsibility

Guidelines – SQLite integration

- There is a demand for supporting more complex queries
- SQLite is a popular choice as database engine in embedded system
- Provides some guidelines how SQLite can be used in a flash friendly way
  - Run the database in a ramdisk
  - Open a database as an in memory database
  - Use the SQLite OS interface

Guideline – Security

- Persistence does not provide a dedicated security concept
- Persistence security be must part of systems security concept
- Nevertheless there are some security guidelines
  - Due to the data separation concept is easy to use Linux users rights concept to limit access to data
  - Additionally Mandatory Access Control can be used
  - Using PCL plugin interface a secure storage can be implemented
  - Encrypted file system can be used to store data

Open Source Project Information

Persistence Project Page

- http://projects.genivi.org/persistence-management

Bugtracker

- http://bugs.genivi.org/enter_bug.cgi?product=Persistence

Mailing list

- http://lists.genivi.org/mailman/listinfo/genivi-persistence

Documentation

- Architecture documentation and users manuals for different components
- http://projects.genivi.org/persistence-management/documentation

GENIVI Compliance status

Persistence Client Library

- Abstract Component – P1
- GENIVI Project
  - Reference implementation is available
    - Developed by Mentor Graphics
  - http://git.projects.genivi.org/?p=persistence/persistence-client-library.git

GENIVI Compliance status

Persistence Administration Service

- Abstract Component – P1
- GENIVI Project
  - Reference implementation is available
    - Developed by Continental AG
  - http://git.projects.genivi.org/?p=persistence/persistence-administrator.git

GENIVI Compliance status

Persistence Health Monitor

- Abstract Component – P2
- GENIVI Project
  - Proof of concept (PoC) is available
    - Developed by Mentor Graphics
  - http://git.projects.genivi.org/?p=persistence/persistence-health-monitor.git

GENIVI Compliance status

Persistence Common Object

- Not in Compliance
- GENIVI Project
  - Different storage backends are available
    - Itzam/C database backend (Continental AG)
      - ➔ not supported anymore
    - key-value store backend (Mentor Graphics)
  - http://git.projects.genivi.org/?p=persistence/persistence-common-object.git

Contributors needed

- Currently two open topics in persistence
  - File caching for the PCL file API
  - SQLite and flash memory
    - How to make work with SQLite in a flash friendly way
- Interested in persistence
  - ➔ GENIVI needs you!!!

Persistence Subsystem

Thank you for your attention!