Persistence Subsystem

27/04/2016 16:15 to 17:30

Guy Sagnes
Persistence Contributor (EG-SI Expert Group)
Continental Automotive GmbH

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 for loading and storing data persistently
- Guarantee the memory device works correctly the complete lifetime of the IVI system
- Provide a solution for everybody
  - Including OSS and legacy components

Introduction

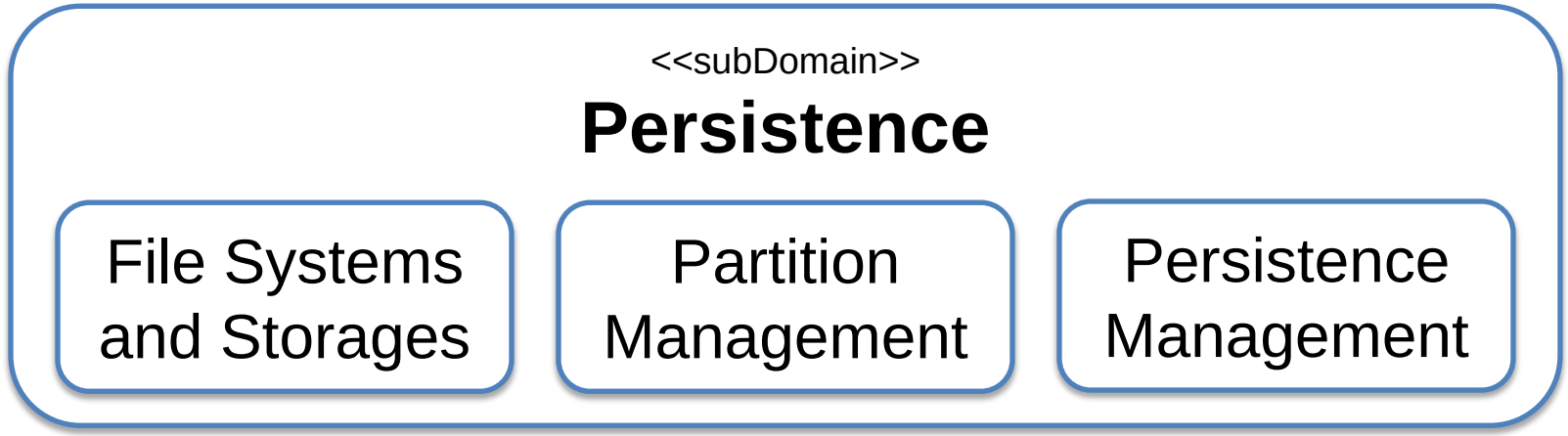
Why do we invent something new?

- Automotive requirements
  - System startup → early data expected like LUC*
  - System shutdown → normal / fast
- Security issues
- Simple and easy to use interface
- Extendable
  - Plug-in API to implement different storage backend

Introduction - Reliability

- Power cut´s
  - Data must be stored power fails save
- Flash memory issues
  - Limited write frequency
    - Limited program erase cycle (max. 100.000 times)

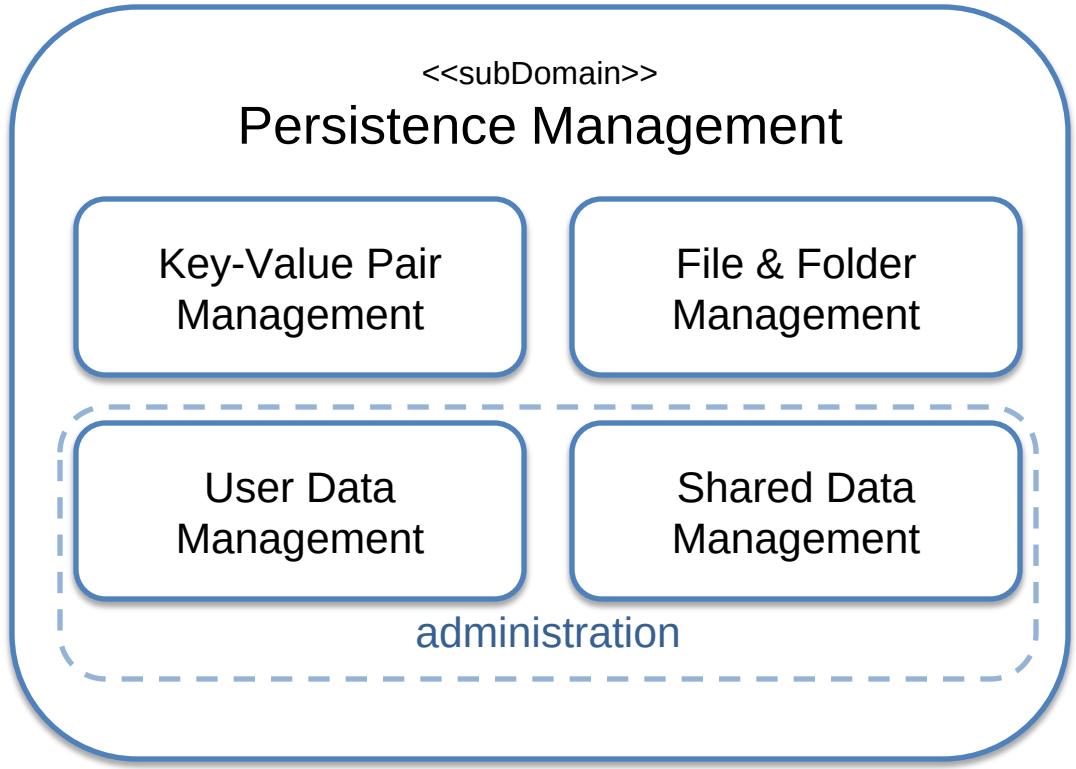
Persistence Scope



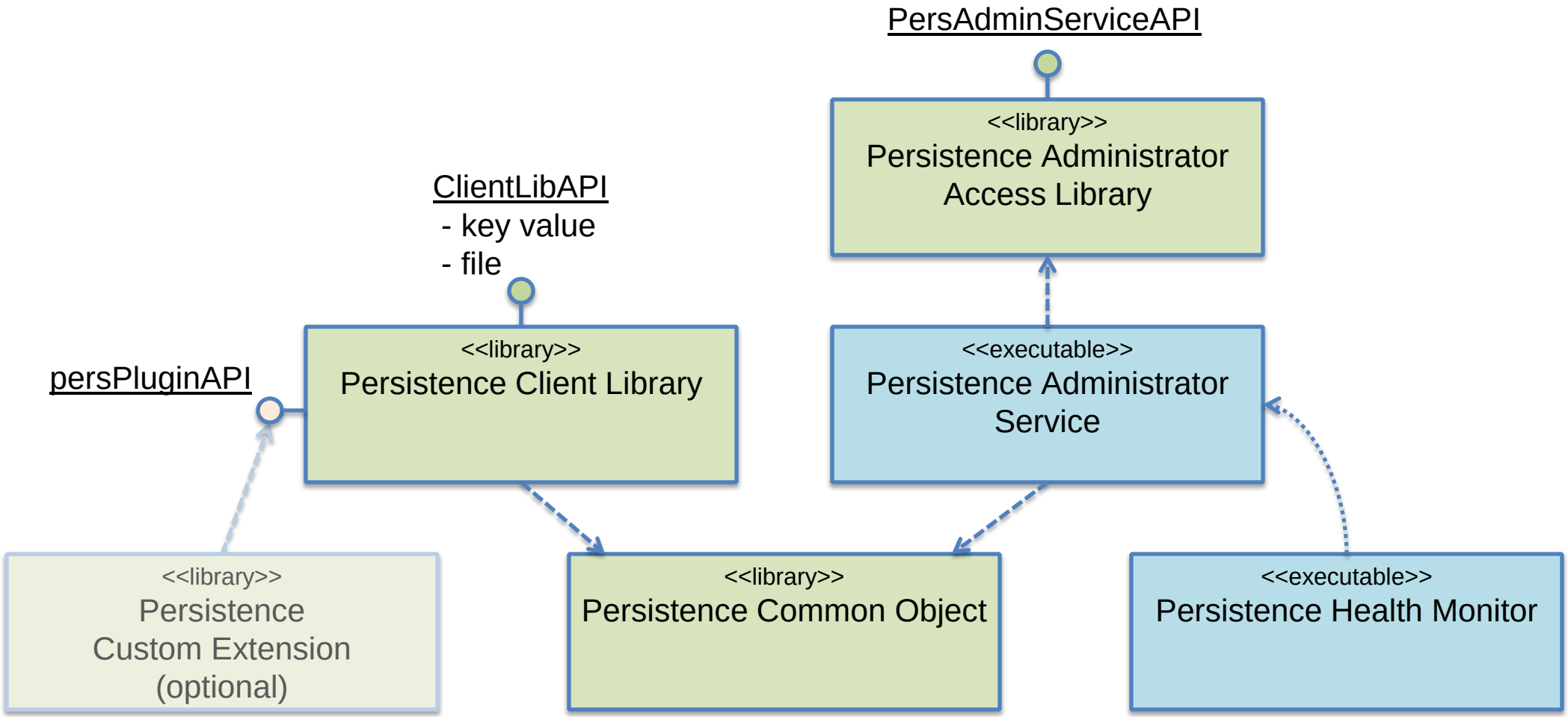
- File Systems and Storages
  - Flash storages (focus NAND)
  - Persistence plug-in for Lifecycle (Health monitoring and repair)
- Partition Management
  - Definition of the system partition
  - Partition Management
- Persistence Management
  - Key/Value Pair Management
  - File- and Folder Structure Management
  - User Data Management
  - Shared Data Management

Persistence Management

- Manages and distinguishes between:
- Shared and application data
- Node and user data
- Write-trough and cached data
- Availability and size
- Change notification of shared data



Components – Persistence Context

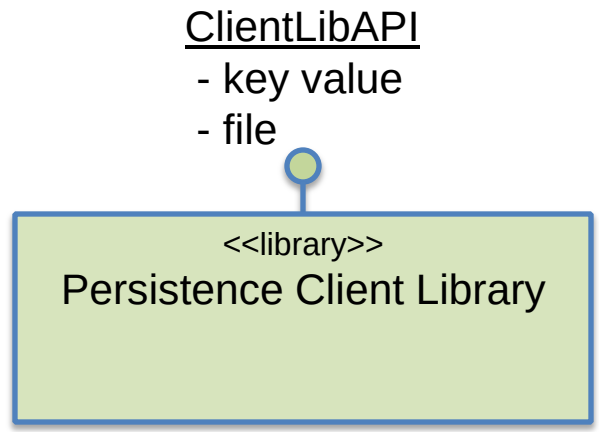


Components – System Context

- Software Management
  - Uses persistence administrator to setup the system based on dedicated resource installation files
- Housekeeping / Diagnostics
  - Uses persistence administration to backup or restore user data

Components – System Context

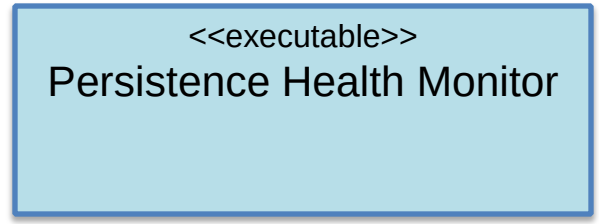
- Applications*
  - Use Persistence Client Library to access data. The access is controlled over the application context.
  - The data is based on
    - key-value (registry)
    - file



*e.g. HMI, Tuner, Navigation, Mediaplayer, Phone…

Components – System Context

- Persistence health monitor
  - Management is based on systemd monitoring and provides different level of escalations, e.g.
    - Application data restore to default
    - Application restart
    - System restart
    - …



Components – System Context

- Persistence Client Library and Persistence Administration Service
  - Use the persistence common object to get a unique management for
    - Database access
    - Synchronization implementation
    - Parsing of configuration

Persistence Common Object

- Key-value store backend is not finalized in the context of the Common Object which not under Compliance scope
- Already prepared scope of realization
  - Itzam/C / initial provided, not more maintained
  - Key-value-store, based on kissdb / recommended
  - RawDB
  - SQLite

Persistence Common Object

Key-value store backend

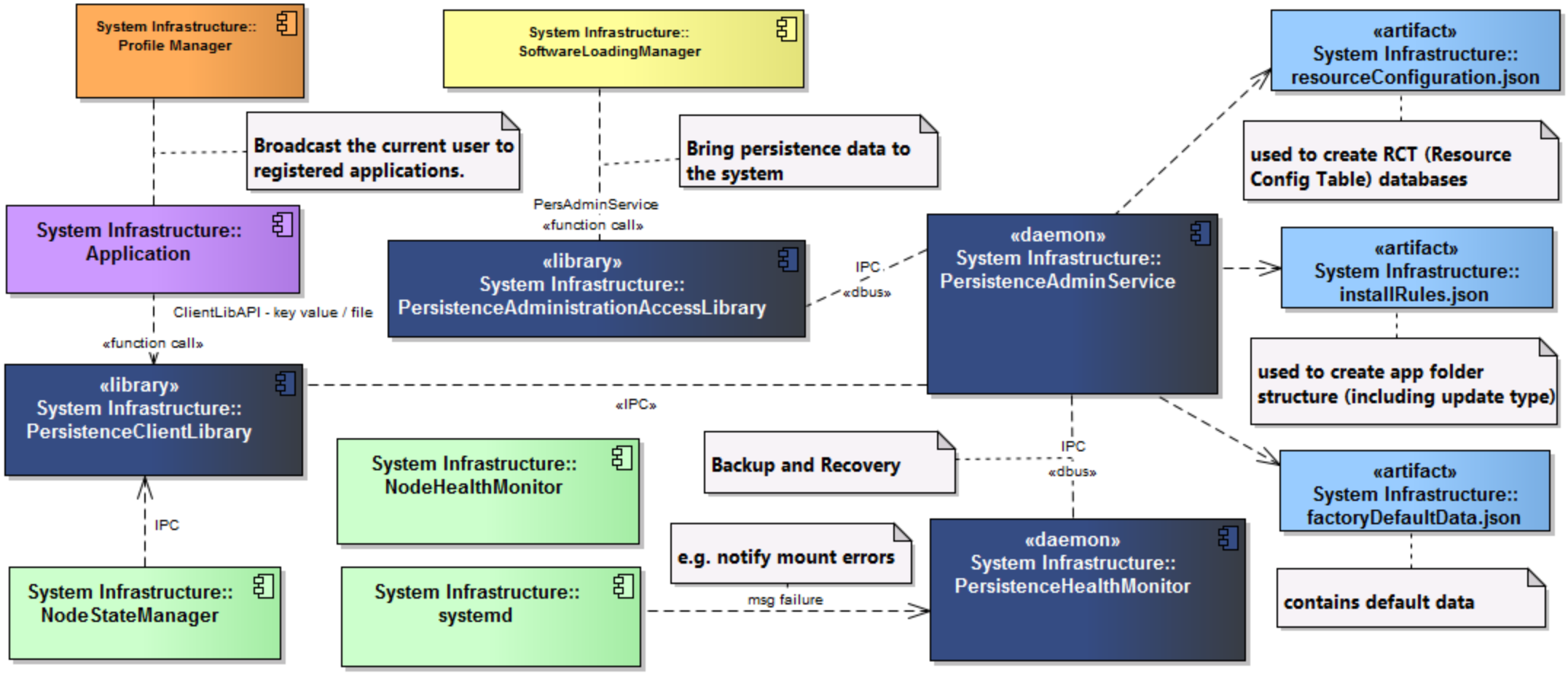
- Used open source database kissdb as basis
  - https://github.com/zerotier/kissdb
- Added features like
  - Data caching
  - Shared access
  - Backup and recovery mechanism
- By default Itzam/C backend is still configured
  - Use configure with „--with-database=key-value-store"

Components – System Context

- Persistence provides over the “Persistence Custom” extension the possibility to adapt the persistence to the given environment limitations.
- Some of the expected extensions are:
  - Factory configuration
  - System coding
  - Specific data encryption, validation

<<library>>
Persistence
Custom Extension
(optional)

Component – System Context



Components – Interface Extract

- Client: initialization access
  - API document
    - library initialization IF Version
    - shutdown notification type definitions
    - shutdown events definitions
    - functions for Library initialization
- Client Library: Generic errors
  - Error definition IF Version
- Client: File access
  - API document
    - file access IF Version
    - functions file access
- Client: Key-value access
  - API document
    - Key-Value access IF Version
    - Defines, Struct, Enum
    - functions Key-Value access

Extract for key-value access

Functions

int	**pclKeyDelete** (unsigned int ldbid, const char *resource_id, unsigned int user_no, unsigned int seat_no) delete persistent data
int	**pclKeyGetSize** (unsigned int ldbid, const char *resource_id, unsigned int user_no, unsigned int seat_no) gets the size of persistent data in bytes
int	**pclKeyReadData** (unsigned int ldbid, const char *resource_id, unsigned int user_no, unsigned int seat_no, unsigned char *buffer, int buffer_size) reads persistent data identified by ldbid and resource_id
int	**pclKeyRegisterNotifyOnChange** (unsigned int ldbid, const char *resource_id, unsigned int user_no, unsigned int seat_no, **pclChangeNotifyCallback_t** callback) register a change notification for persistent data
int	**pclKeyUnRegisterNotifyOnChange** (unsigned int ldbid, const char *resource_id, unsigned int user_no, unsigned int seat_no, **pclChangeNotifyCallback_t** callback) unregister a change notification for persistent data
int	**pclKeyWriteData** (unsigned int ldbid, const char *resource_id, unsigned int user_no, unsigned int seat_no, unsigned char *buffer, int buffer_size) writes persistent data identified by ldbid and resource_id

Components – Interface Extract

```
int pclKeyReadData ( unsigned int      ldbid,
                     const char *      resource_id,
                     unsigned int      user_no,
                     unsigned int      seat_no,
                     unsigned char *   buffer,
                     int               buffer_size
                   )
```

reads persistent data identified by ldbid and resource_id

Parameters:

ldbid	logical database ID
resource_id	the resource ID
user_no	the user ID; user_no=0 can not be used as user-ID beacause '0' is defined as System/node
seat_no	the seat number
buffer	the buffer to read the persistent data
buffer_size	size of buffer for reading

Returns:

positive value (0 or greater): the bytes read; On error a negative value will be returned with th following error codes:

- **EPERS_LOCKFS EPERS_NOT_INITIALIZED EPERS_BADPOL EPERS_NOPLUGINFUNCT**
- since V6.1.1: **EPERS_RES_NO_KEY** : the specified resource is not a key
- since V6.1.2: **EPERS_NOPRCTABLE**: the application (folder) is not or incorrectly installed, **EPERS_CREATE_NOT_ALLOWED**: resource creation not allowed

Components – Interface Extract

- Administration and service
  - API document
    - persAdmin Return Values
    - Configuration parameter

Extract for Administration access

Functions

long	**persAdminDataExport** (**PersASSelectionType_e** type, const char *output_pathname, const char *applicationID, unsigned int user_no, unsigned int seat_no) Allow creation of a data export on different level (application, user or complete)
long	**persAdminDataImport** (**PersASSelectionType_e** type, const char *input_pathname, const char *applicationID, unsigned int user_no, unsigned int seat_no) Allow import (from an export data file) on different level (application, user or complete)
long	**persAdminDataReset** (**PersASSelectionType_e** type, **PersASDefaultSource_e** defaultSource, const char *applicationID, unsigned int user_no, unsigned int seat_no) Allow restore of values from default on different level (application, user or complete)
long	**persAdminResourceConfigAdd** (const char *resource_file) Allow the configuration of persistence data based on the information contained in a Resource Installation File.
long	**persAdminUserDataCopy** (unsigned int src_user_no, unsigned int src_seat_no, unsigned int dest_user_no, unsigned int dest_seat_no) Allow the copy of user related data between different users.
long	**persAdminUserDataDelete** (unsigned int user_no, unsigned int seat_no) Delete the user related data from persistence containers.

Components – Interface Extract

long persAdminResourceConfigAdd (const char * resource_file)

Allow the configuration of persistence data based on the information contained in a Resource Installation File.

Parameters:

resource_file pathname of the Resource Installation File

Returns:

positive value for success (the actual value could offer an indication about the amount of configured data); negative value: error code (**persAdmin Return Values**)

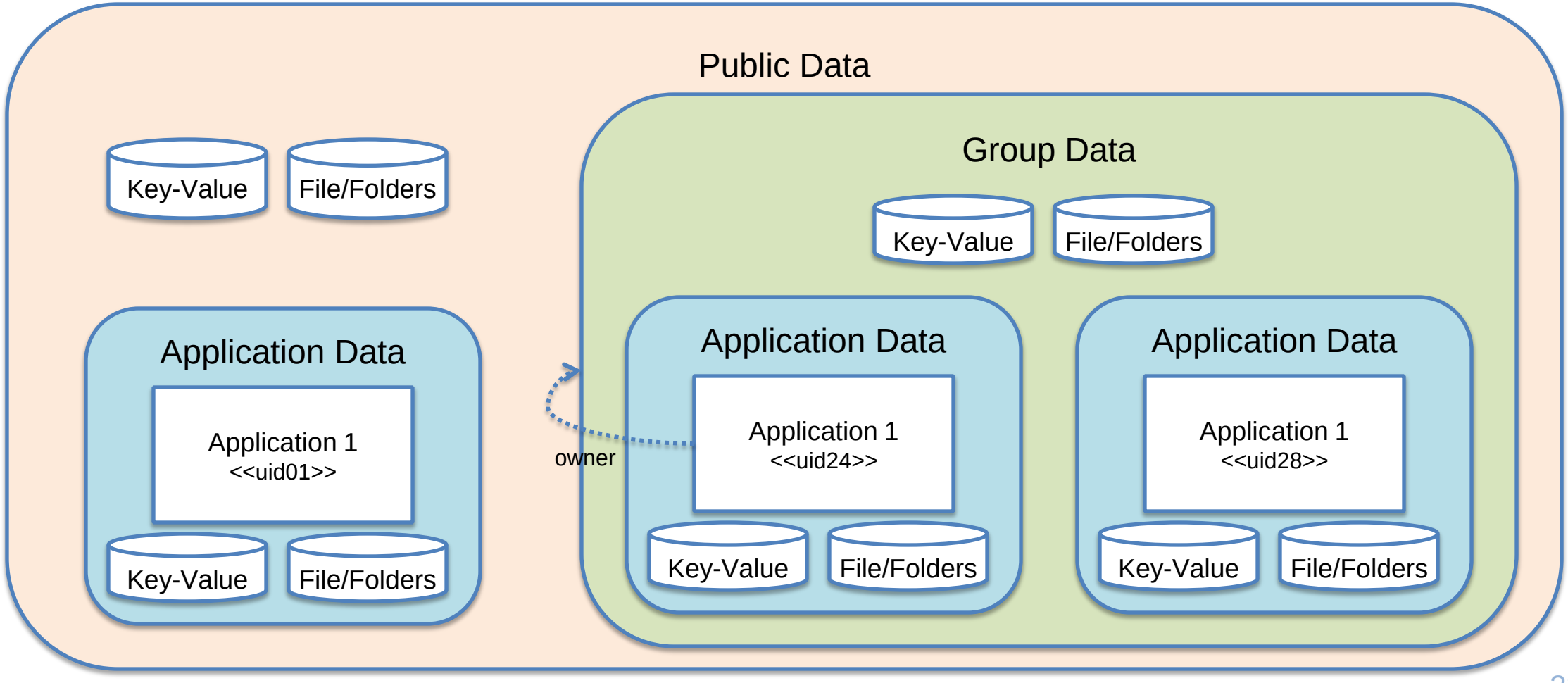
Note:

- For more information see:
- https://collab.genivi.org/wiki/download/attachments/55869520/GENIVI_PersAdmin_ResourceInstallation.pdf?version=1&modificationDate=1386073820142

Concept – Data Separation

- Local data
  - Access is limited to the application itself
- Group shared data
  - Shared by a group of applications
  - Group is owned by a single application
  - Only owner application is able to write data
- Public shared data
  - Shared by any application within the system
  - Only infrastructure data should be managed there
- 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

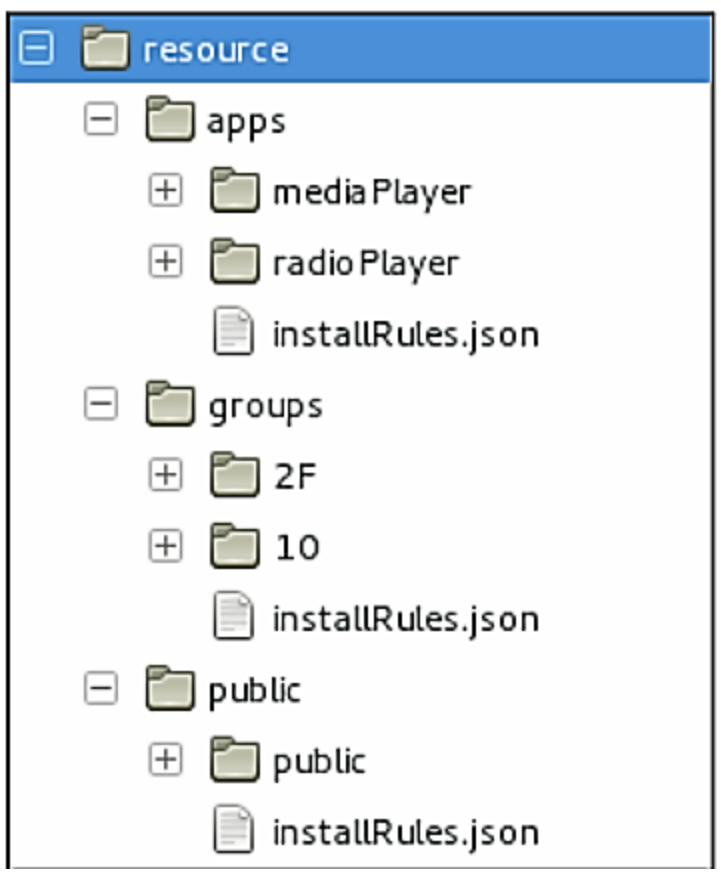
- Create default application folders including links to shared data
- Deploy the default content
- Create local database for each application
- Create shared databases
- Provide application specific links to shared databases
  - (group/ public)
- Setup of application file system access policies
- Delete,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

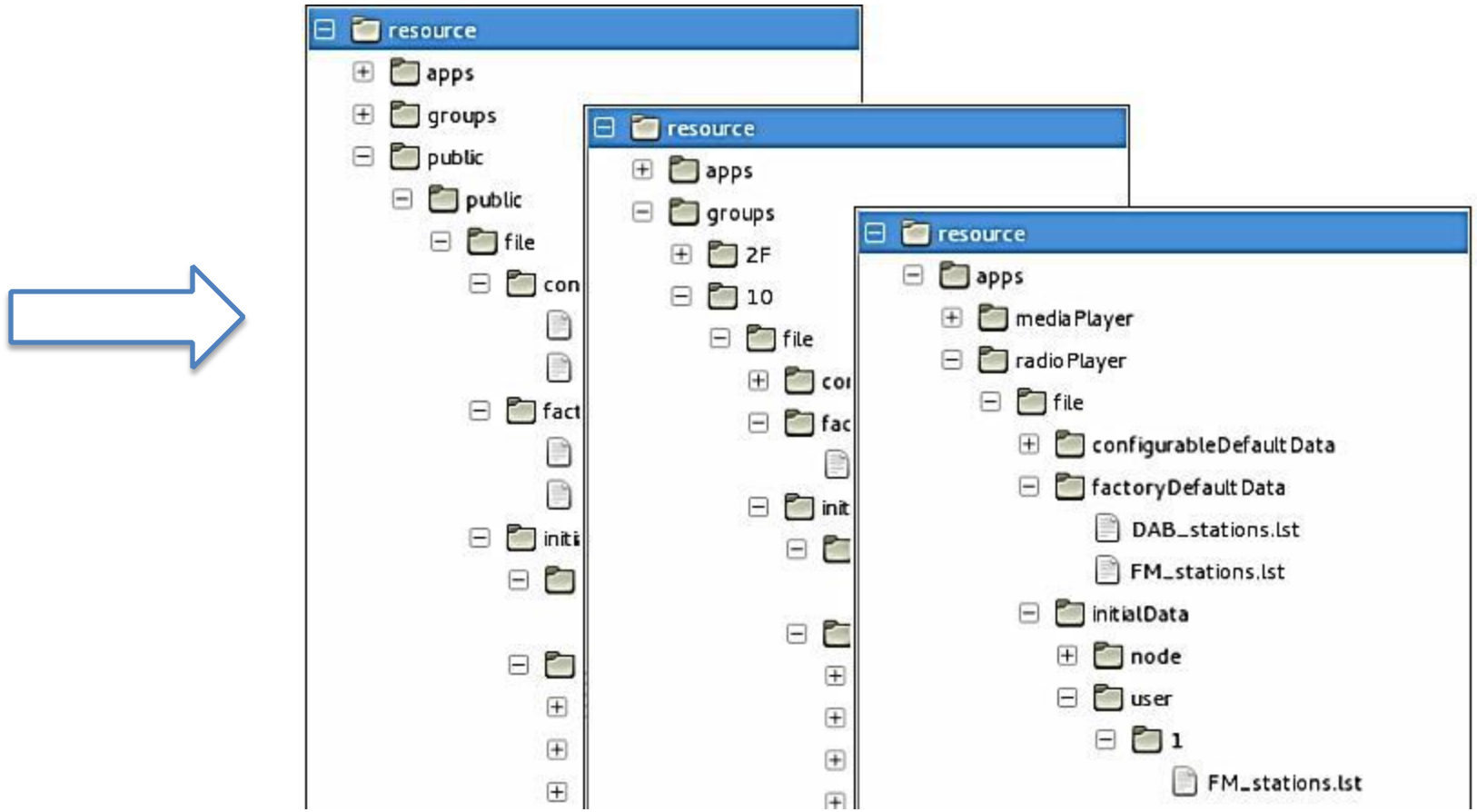
Setup Persistence Data - Scope

- 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)

Persistence Data - Organization

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





Setup Persistence – 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_NewI
      nstall",
"AppX":"PersAdminCfgInstallRules_Uninstall",
}
```

Setup persistence – 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 persistence – Resource configuration

Resource 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"                 → hash / custom identifier
 }
```

Setup persistence – Resource configuration

JSON rule data

- Factory default data
- Configurable default data

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

Setup persistence – 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
  - released as an open GENIVI project
    - Developed by Mentor Graphics (maintainer)

http://git.projects.genivi.org/?p=eclipse-json-gui.git

Setup persistence – Resource / perspective

- For the management of the Resource Installation File a central management can be developed to assume the access in a product context between the different shared application data.
- Based on the simple Eclipse Plug-in a merge tool can be implemented to allow merging of the resource installation file to avoid e.g. complex management during the factory setup phase for the product
  - Excurse: Continental develops a central web based management of the resources

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 integrated usage
  - Store data in the location persistence provides
  - Backup/recovery is in the responsibility of the application
- Free usage
  - Store data wherever 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 provides based on the data separation concept the base to allow dedicated integration of security requirements.
  - Limitation of the data access based on Linux users right
  - Additional Mandatory Access Control can be used
  - Definition of dedicated secure storage over Custom Plug-in
  - Reuse of the file system based capability e.g. efs
- Basically the final aspect of security in persistence context has to be part of the overall security concept.

Open Source Project Information

Persistence Project Page

- http://projects.genivi.org/persistence-management
  (Start moving to https://at.projects.genivi.org/wiki/display/PROJ/Persistence+Client+Library)

Bug tracker

- 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 (maintainer)
  - 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 (Maintainer)
  - 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 (Maintainer)
  - http://git.projects.genivi.org/?p=persistence/persistence-health-monitor.git

GENIVI Compliance Status

Persistence Common Object

- Not in compliance
- GENIVI Project
  - Developed by Continental AG (Maintainer)
  - Different storage backend available
    - Itzam/C database backend (initial provided, not used)
    - 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

Persistence Subsystem

Thank you for your attention!

addendum

- Lifecycle usage recommendation
  - In context of the implementation is clearly expected that with the current persistence client library the initialization of done by the application is done with notification against the NSM notification.
  - The application has the responsibility to call the corresponding pclLifecycleSet ( ) API for PCL in context of the shutdown call implemented by the application