Persistency in Production

Month 04, 2018 | Using Genivi persistency in production

KPIT Technologies : Vikrant Bhangay, Nisha Parrakat
GENIVI Alliance

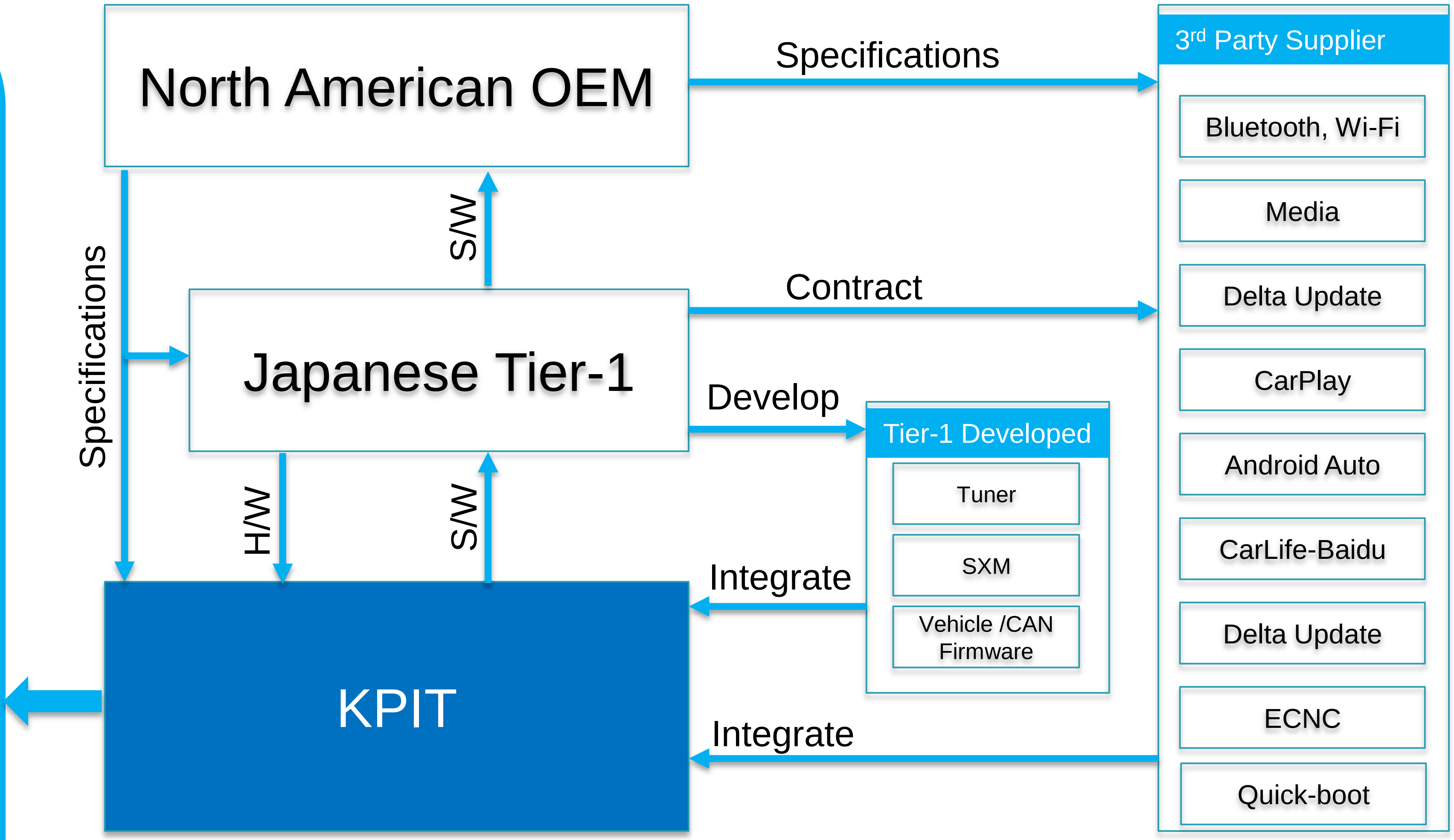
Content

- Production Program - KPIT Role
- Software Platform Architecture
- GENIVI Components Adaptation
- GENEVI Adaptation – Persistency
- Persistency Adaptation Challenges
- Persistency Adaptation Approach
- Custom Tools / Utilities
- Problems Experienced
- Next steps

Production Program - KPIT Role

KPIT as a Lead Software Integrator for MY18 & MY19 Production Program

- **Complete SW integration & release**
- HMI and application development
- KIVI middleware porting and adaptation
- System Infrastructure development / Adaptation
- Linux BSP porting, tuning and optimization
- 3rd party software integration
- Cybersecurity, Fast Boot, FOTA
- Ownership of system KPIs
- Build management
- Integration Testing



North American OEM

Specifications

Specifications

S/W

Japanese Tier-1

Contract

Develop

H/W

S/W

KPIT

Integrate

Integrate

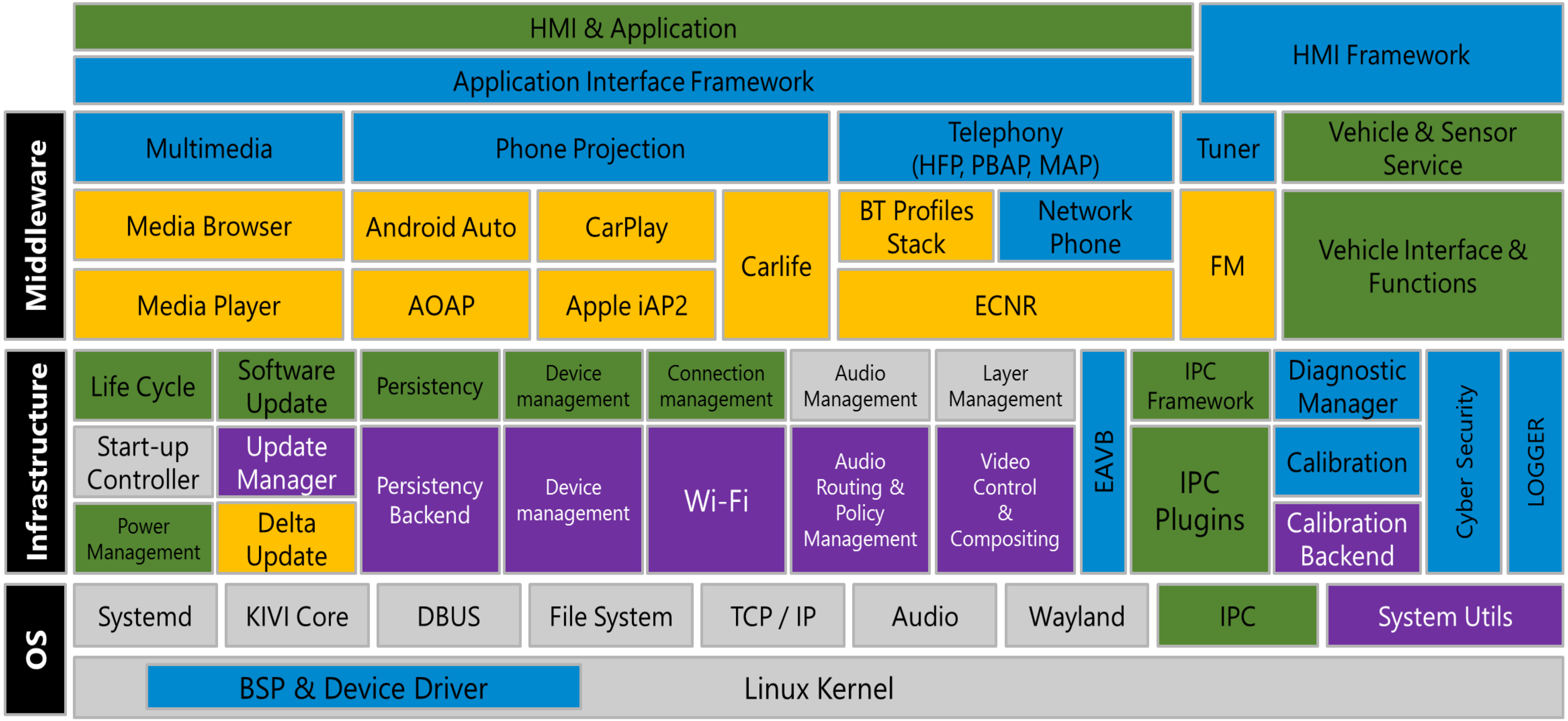
Tier-1 Developed

- Tuner
- SXM
- Vehicle /CAN Firmware

3rd Party Supplier

- Bluetooth, Wi-Fi
- Media
- Delta Update
- CarPlay
- Android Auto
- CarLife-Baidu
- Delta Update
- ECNC
- Quick-boot

KIVI - Software Platform Architecture



HMI & Application

Application Interface Framework

HMI Framework

Middleware

Multimedia

Phone Projection

Telephony (HFP, PBAP, MAP)

Tuner

Vehicle & Sensor Service

Media Browser

Android Auto

CarPlay

Carlife

BT Profiles Stack

Network Phone

FM

Vehicle Interface & Functions

Media Player

AOAP

Apple iAP2

ECNR

Infrastructure

Life Cycle

Software Update

Persistency

Device management

Connection management

Audio Management

Layer Management

EAVB

IPC Framework

Diagnostic Manager

Cyber Security

LOGGER

Start-up Controller

Update Manager

Persistency Backend

Device management

Wi-Fi

Audio Routing & Policy Management

Video Control & Compositing

IPC Plugins

Calibration

Power Management

Delta Update

Calibration Backend

OS

Systemd

KIVI Core

DBUS

File System

TCP / IP

Audio

Wayland

IPC

System Utils

BSP & Device Driver

Linux Kernel

🟩 New/Custom Development 🟦 KPIT Components 🟨 3rd Party Components ⬜ OSS Components 🟪 Adaptation/Customize of OSS

GENIVI Component Adaptation in KIVI

Life Cycle Management

GENIVI provided LCM components are used almost as it is with minor bug fixes

- Node State Manager (NSM)
- Node Startup Controller (NSC)

Persistency

GENIVI provided Persistency components are used almost as it is with minor bug fixes

- Persistence Client Library (PCL v1.0.0) (user 1, seat 0) , private data
- Persistence Administrator (PAS v 1.0.5)
- Persistence Common Object (PCO v1.0.3)

Audio Manager

GENIVI Audio Manager framework used. All plugins are developed by KPIT as per production program requirement

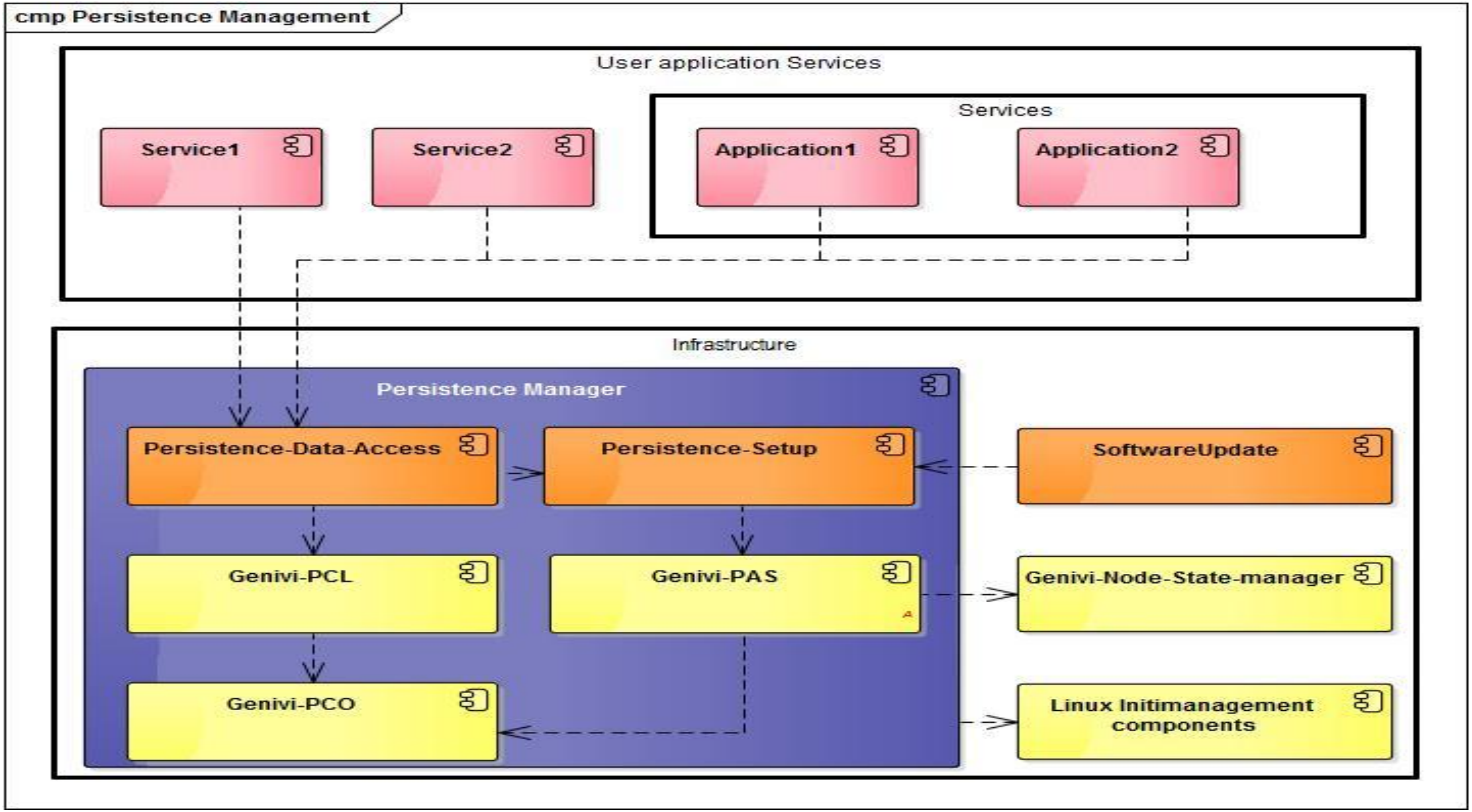
- Audio Manager Daemon (version 1.0)
- Audio Manager Controller Plugin (version 1.0)
- Audio Manager Routing Plugin (version 1.0)
- Audio Manager Command Plugin (version 1.0)

Assumption : Understanding of Genivi components

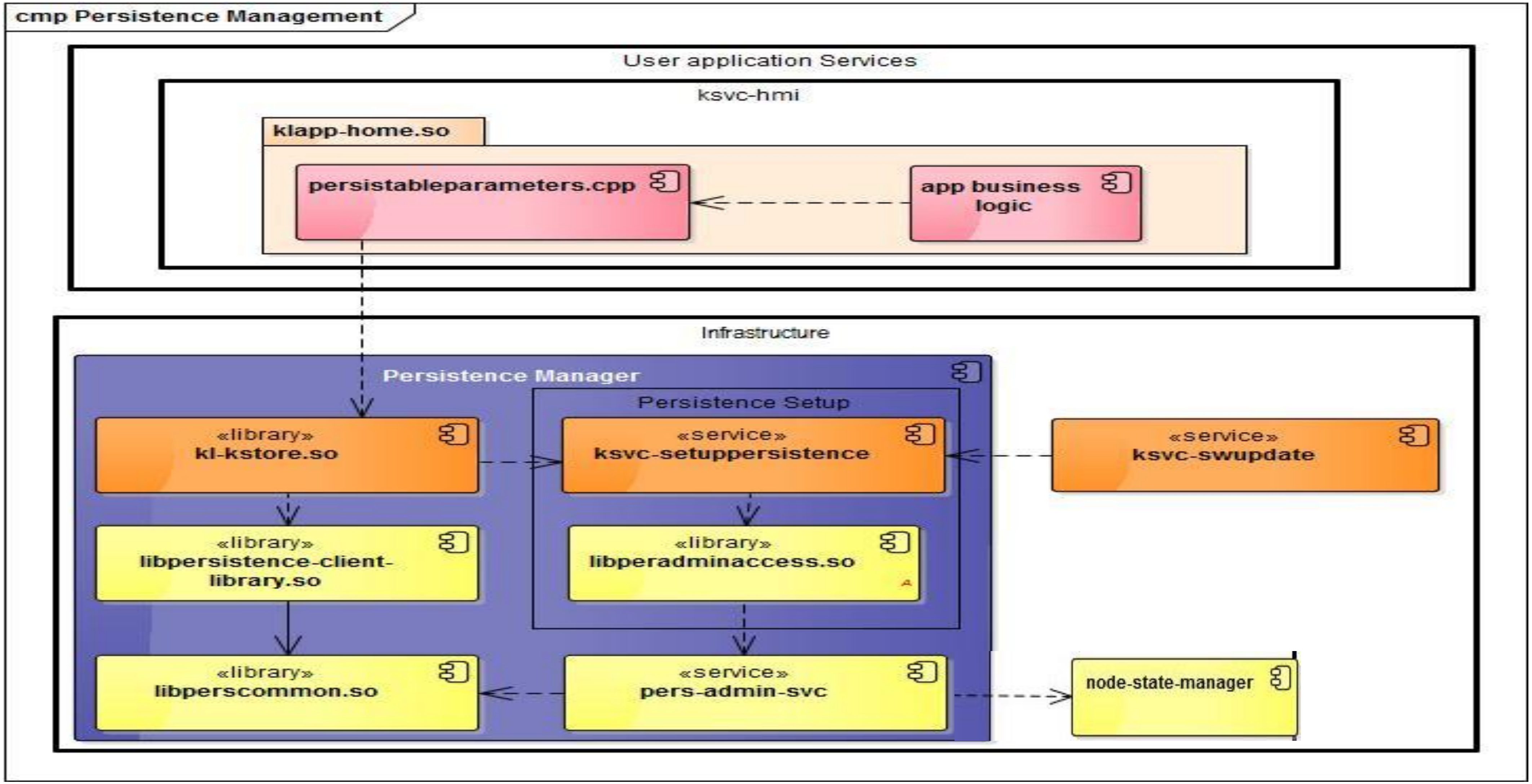
We know

- ➢ Persistence Client Library
- ➢ Persistence Administrator
- ➢ What are Cacheable items, Write-through items.
- ➢ Persistence setup Json tar files
- ➢ Persistence folder structure

GENEVI Adaptation – Persistency



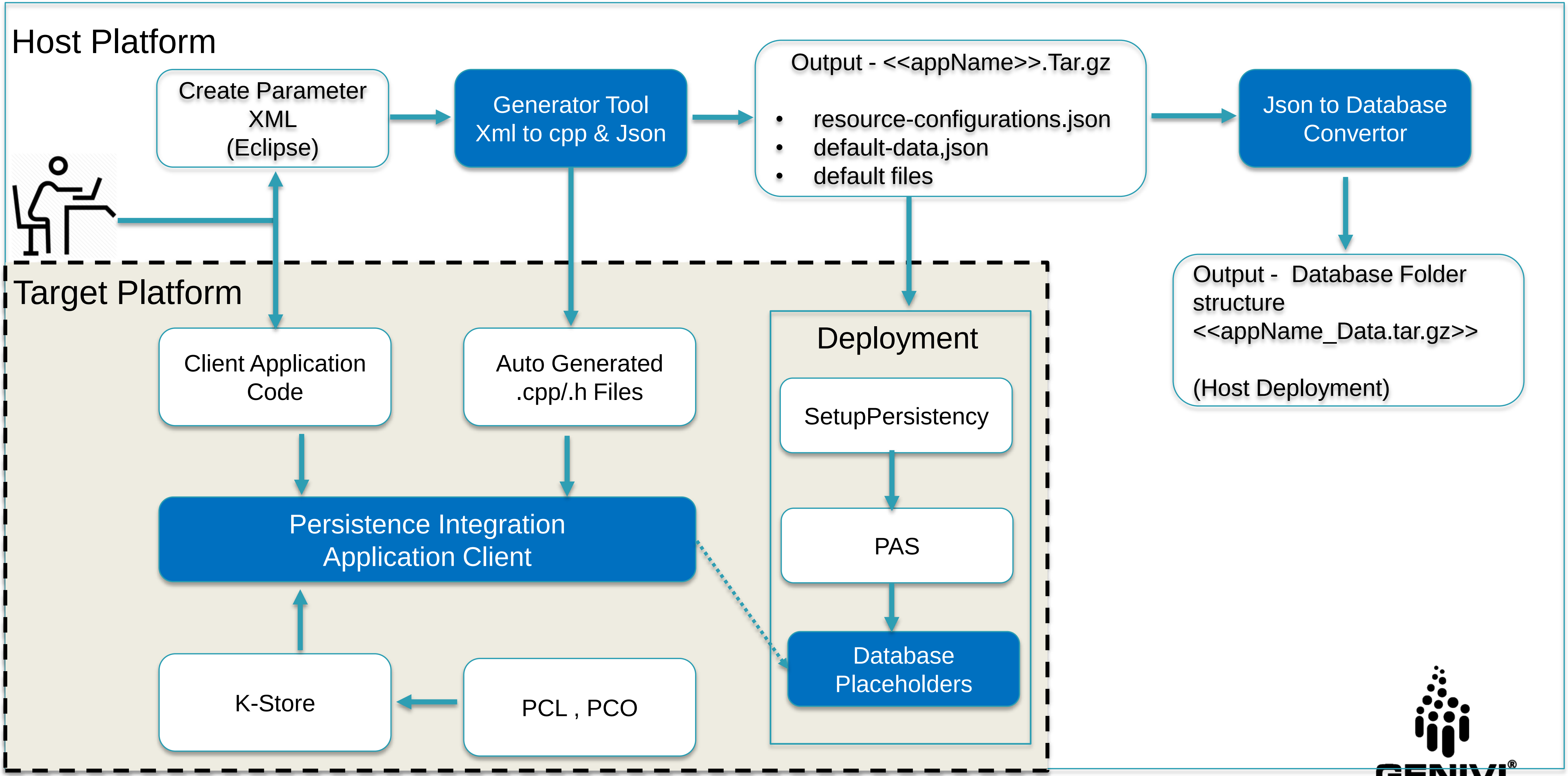
GENIVI Adaptation – Deployment



Persistency Adaption Challenges for Production

1. Database deployment on target
   - ➢ First time Deployment during EOL Testing
   - ➢ Deployment of newly added applications using software update
   - ➢ Deployment on host during development and testing
2. Serialization & De-serialization of key-value pair data
3. Use of same database for multiple applications as a part of single process
4. Enforcing cacheable and write-through behaviors.
5. Backup trigger management
6. Exception handling (File system corruption, mount failure,)
7. Custom tools requirement for development and testing

1.1 Approach – App deployment workflow

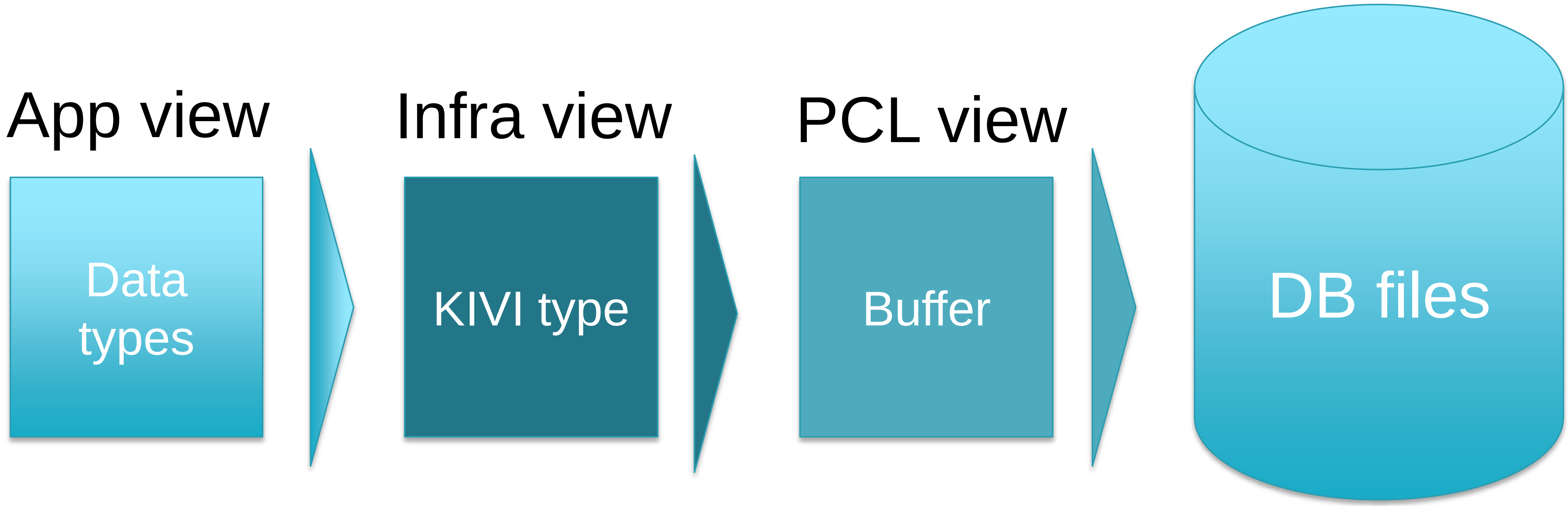

Host Platform
Create Parameter XML (Eclipse)
Generator Tool Xml to cpp & Json
Output - <<appName>>.Tar.gz
• resource-configurations.json
• default-data,json
• default files
Json to Database Convertor
Output - Database Folder structure <<appName_Data.tar.gz>>
(Host Deployment)
Target Platform
Client Application Code
Auto Generated .cpp/.h Files
Deployment
SetupPersistency
PAS
Database Placeholders
Persistence Integration Application Client
K-Store
PCL , PCO

1.2 Approach - Database Deployment / Installation On Target

First time installation and through S/W update

- Rootfs contains application JSON tar files
- During first boot before application is fully operational respective database folder structure is created
- New application database is installed using software update.

2.1 Approach – Abstract Serialization & de-serialization



App view

Data types

Infra view

KIVI type

PCL view

Buffer

DB files

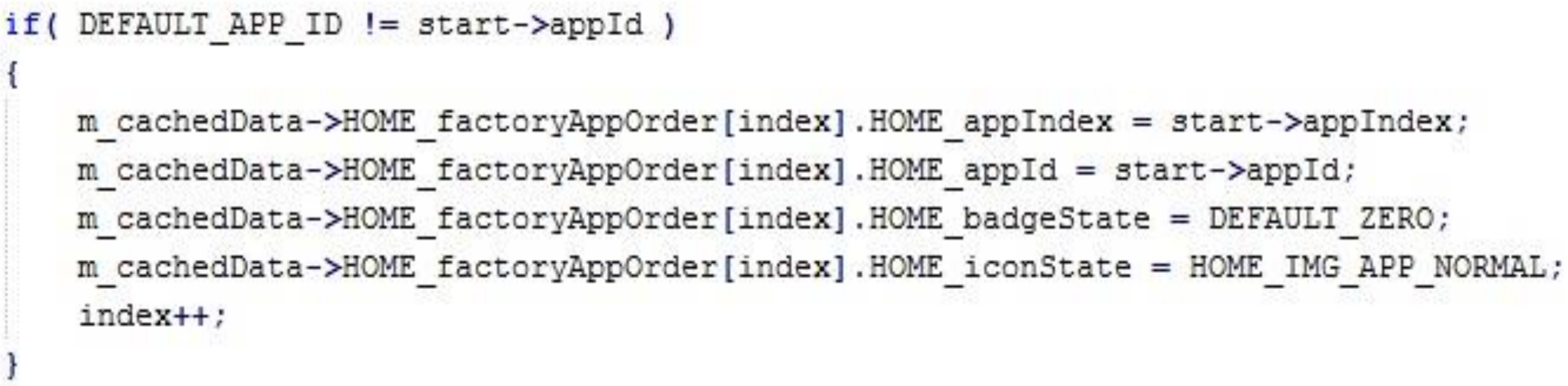
string listsoflinks;

kvariant_t(listoflinks);

unsigned Char* RawData;

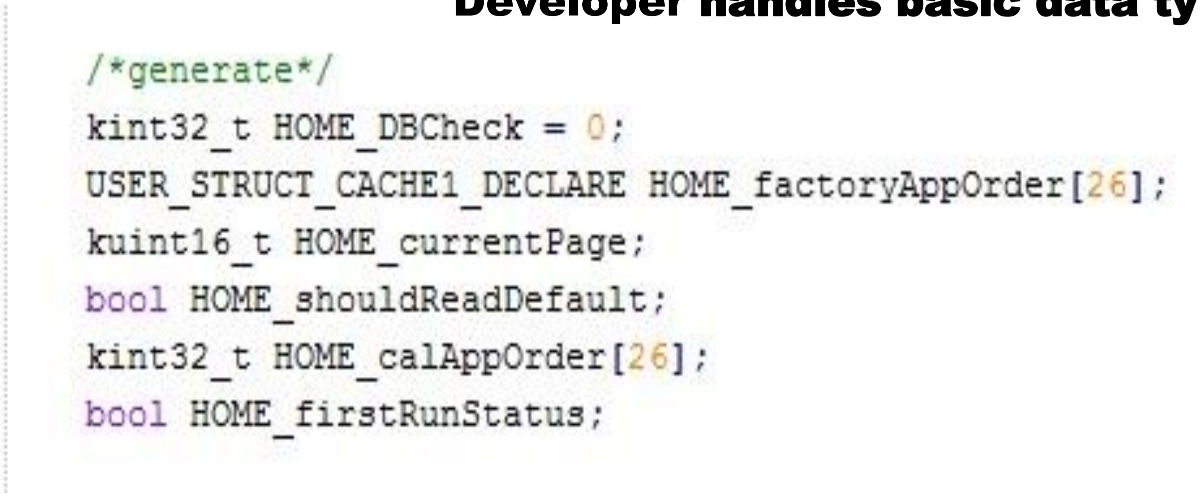
2.2 Approach – Application uses primary data types

Application Persistables as Simple variables



```
if( DEFAULT_APP_ID != start->appId )
{
    m_cachedData->HOME_factoryAppOrder[index].HOME_appIndex = start->appIndex;
    m_cachedData->HOME_factoryAppOrder[index].HOME_appId = start->appId;
    m_cachedData->HOME_factoryAppOrder[index].HOME_badgeState = DEFAULT_ZERO;
    m_cachedData->HOME_factoryAppOrder[index].HOME_iconState = HOME_IMG_APP_NORMAL;
    index++;
}
```

Developer handles basic data types



```
    /*generate*/
    kint32_t HOME_DBCheck = 0;
    USER_STRUCT_CACHE1_DECLARE HOME_factoryAppOrder[26];
    kuint16_t HOME_currentPage;
    bool HOME_shouldReadDefault;
    kint32_t HOME_calAppOrder[26];
    bool HOME_firstRunStatus;
```

2.3 Activity - Persistency integration

- Create Parameters.xml

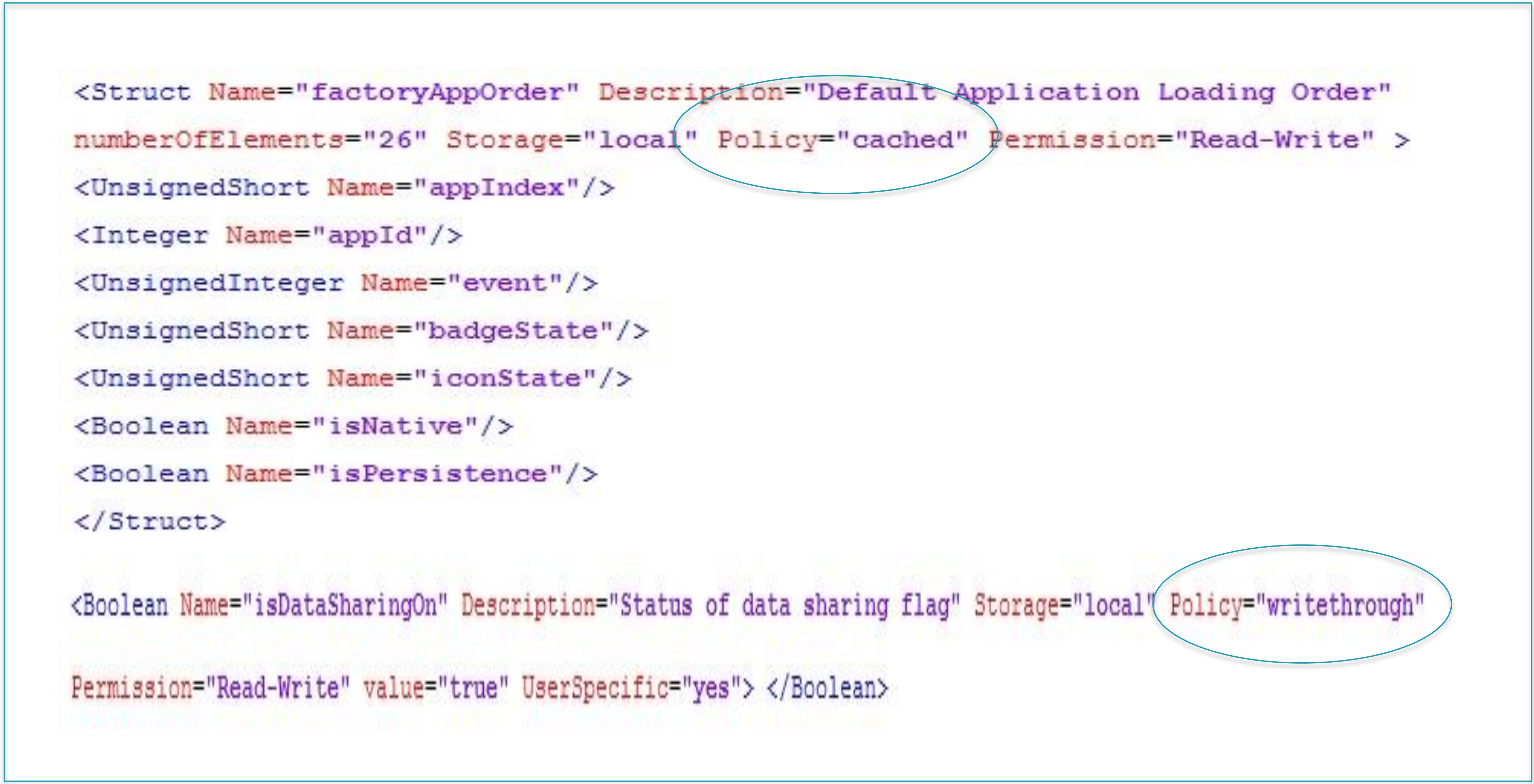
sample.xml.html

- Autogenerate persistableparameters

sample_PersistableParameters_h.html

- Use persistables in normal code

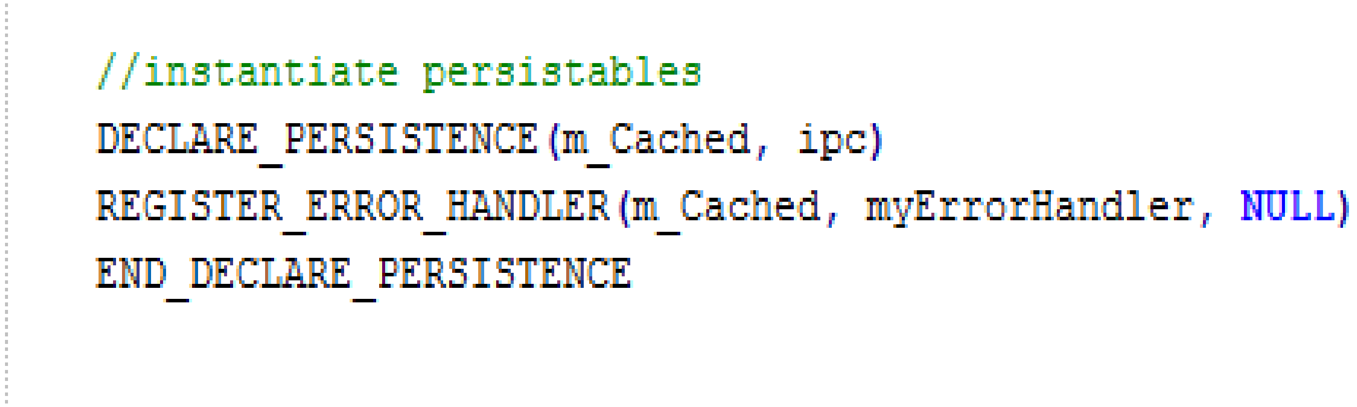
2.5 Snapshot – Cachables and Writethrough



```
<Struct Name="factoryAppOrder" Description="Default Application Loading Order"
numberOfElements="26" Storage="local" Policy="cached" Permission="Read-Write" >
<UnsignedShort Name="appIndex"/>
<Integer Name="appId"/>
<UnsignedInteger Name="event"/>
<UnsignedShort Name="badgeState"/>
<UnsignedShort Name="iconState"/>
<Boolean Name="isNative"/>
<Boolean Name="isPersistence"/>
</Struct>

<Boolean Name="isDataSharingOn" Description="Status of data sharing flag" Storage="local" Policy="writethrough"

Permission="Read-Write" value="true" UserSpecific="yes"> </Boolean>
```

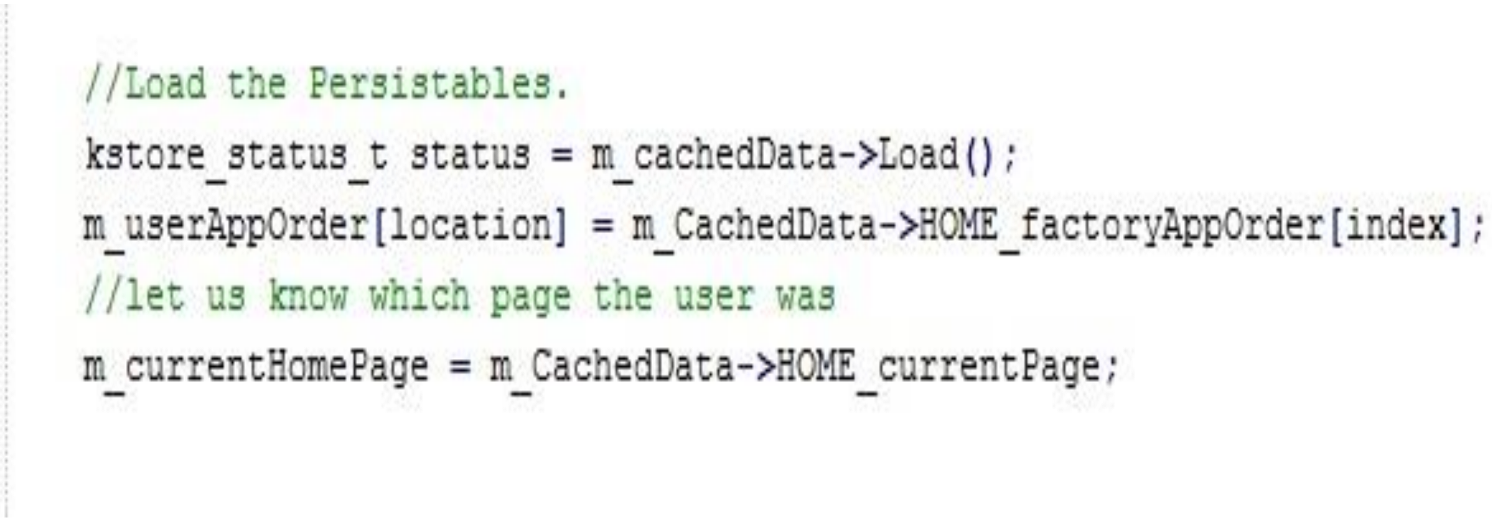
2.4 Snapshot - Persistables in normal code.

Instantiate the Persistence autogenerated classes.



```
//instantiate persistables
DECLARE_PERSISTENCE(m_Cached, ipc)
REGISTER_ERROR_HANDLER(m_Cached, myErrorHandler, NULL)
END_DECLARE_PERSISTENCE
```

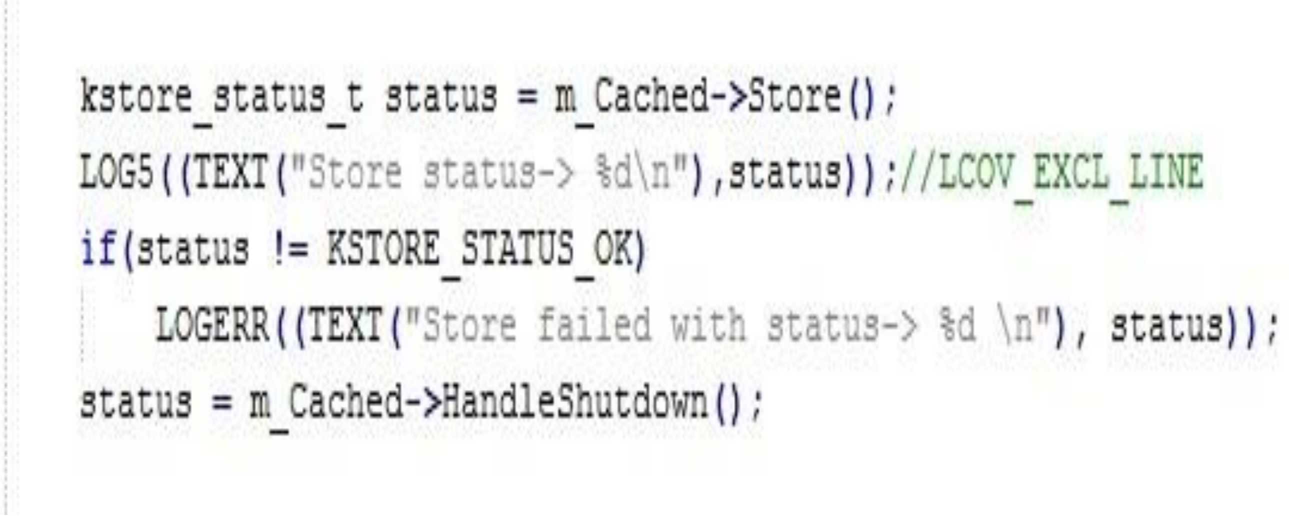
Load the Persistable data at startup



```
//Load the Persistables.
kstore_status_t status = m_cachedData->Load();
m_userAppOrder[location] = m_CachedData->HOME_factoryAppOrder[index];
//let us know which page the user was
m_currentHomePage = m_CachedData->HOME_currentPage;
```

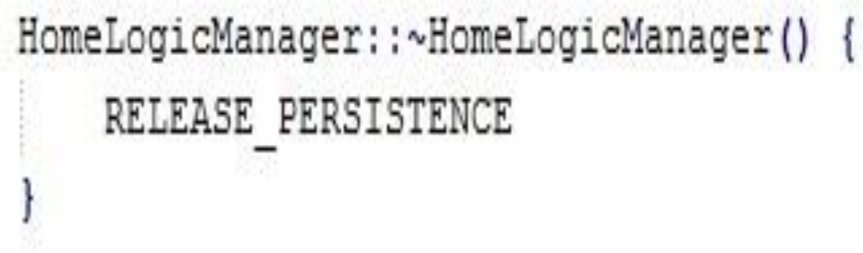
2.4 Snapshot - Persistables in normal code.

Store the Persistable data at shutdown.



```
kstore_status_t status = m_Cached->Store();
LOG5((TEXT("Store status-> %d\n"),status));//LCOV_EXCL_LINE
if(status != KSTORE_STATUS_OK)
    LOGERR((TEXT("Store failed with status-> %d \n"), status));
status = m_Cached->HandleShutdown();
```

Destroy the Persistables



```
HomeLogicManager::~HomeLogicManager() {
    RELEASE_PERSISTENCE
}
```

Sample code HMI

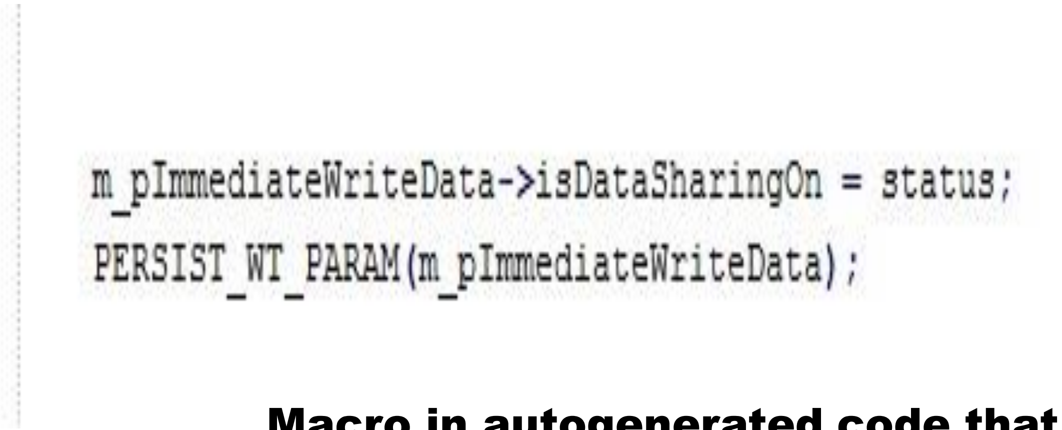
Sample_Src_HomeLogicManager_cpp.html

Sample_Media_file_usage_cpp.html

SampleFavoriteManager_FilePathType.cpp.html

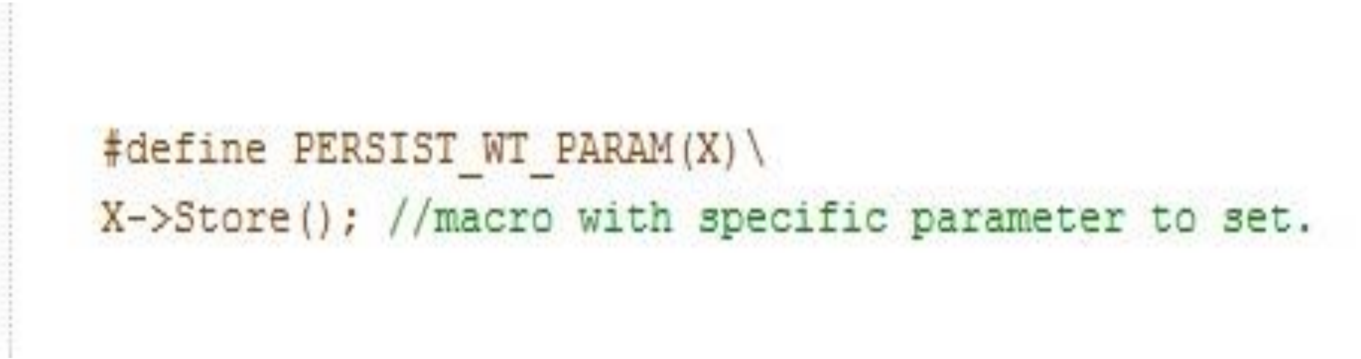
2.5 Snapshot - Writethrough Persistables

Assign value and call macro to store



```
m_pImmediateWriteData->isDataSharingOn = status;
PERSIST_WT_PARAM(m_pImmediateWriteData);
```

Macro in autogenerated code that calls explicit set.



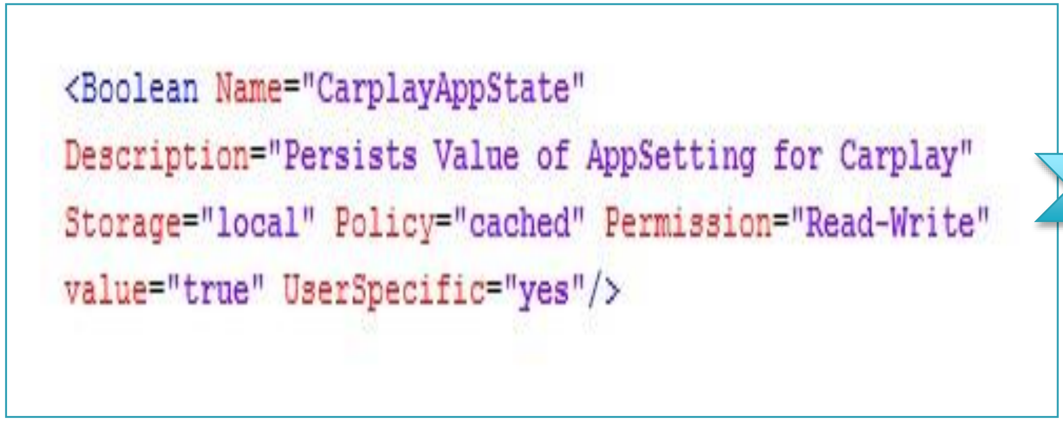
```
#define PERSIST_WT_PARAM(X)\
X->Store(); //macro with specific parameter to set.
```

2.6 Approach – Advantages

- Abstraction of persistence APIs.
- Abstraction of storage policy of persistency infrastructure.
- Automatic coupling to lifecycle.
- Json Tars with the app developer.
- Ease of trying out and testing.
- Less number of PCL Key-value pairs so low memory utilization
- Multiple applications within single process using same database
- Multiple applications under multiple team working for same process
- Conditional backup during shutdown

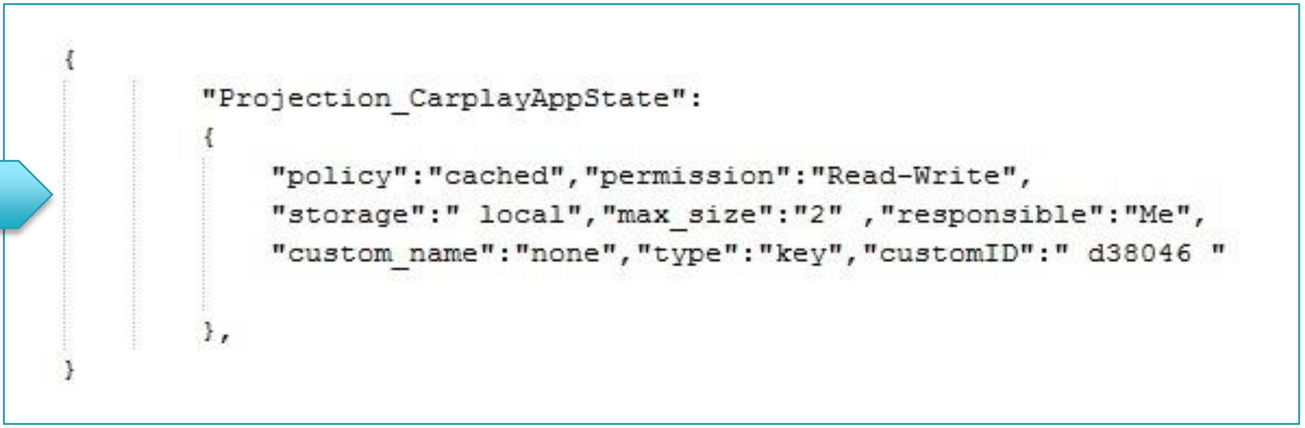
3.1 Approach – multiple xmls generate multiple json

App_projections.xml



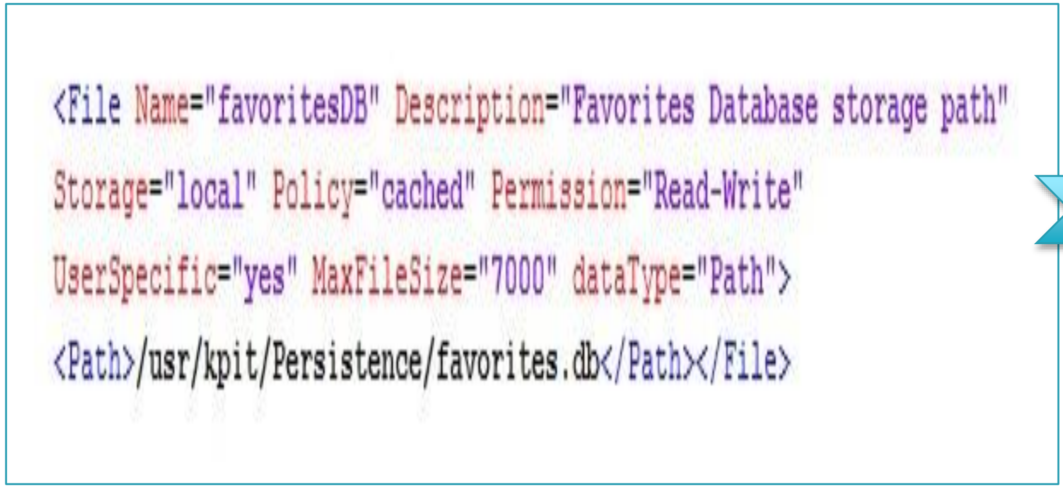
```xml
<Boolean Name="CarplayAppState"
Description="Persists Value of AppSetting for Carplay"
Storage="local" Policy="cached" Permission="Read-Write"
value="true" UserSpecific="yes"/>
```

Resource-configuration.json



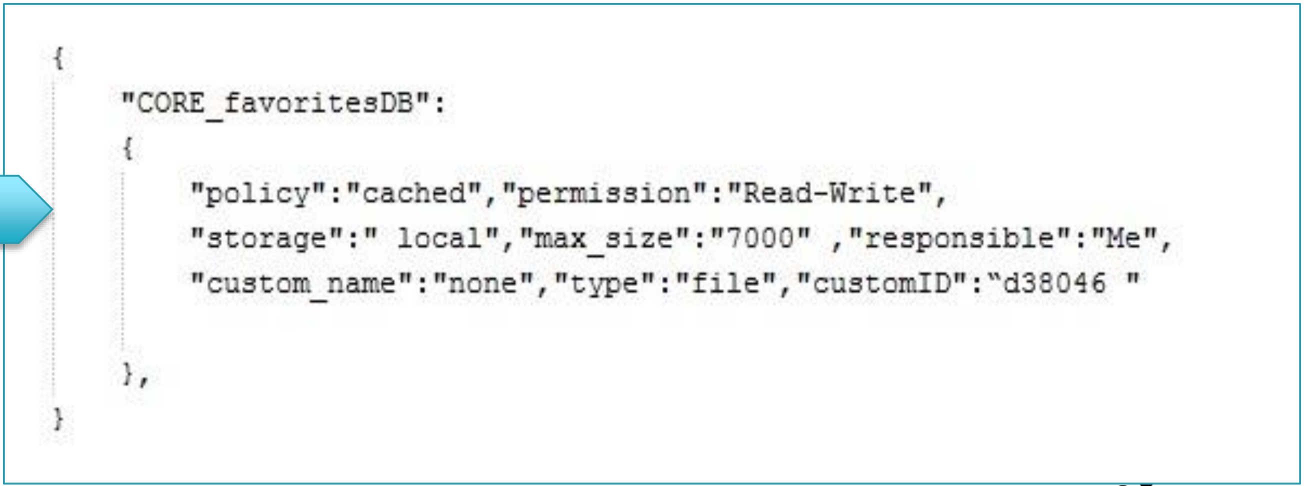
```json
{
        "Projection_CarplayAppState":
        {
            "policy":"cached","permission":"Read-Write",
            "storage":" local","max_size":"2" ,"responsible":"Me",
            "custom_name":"none","type":"key","customID":" d38046 "

        },
}
```

App_hmi_core.xml

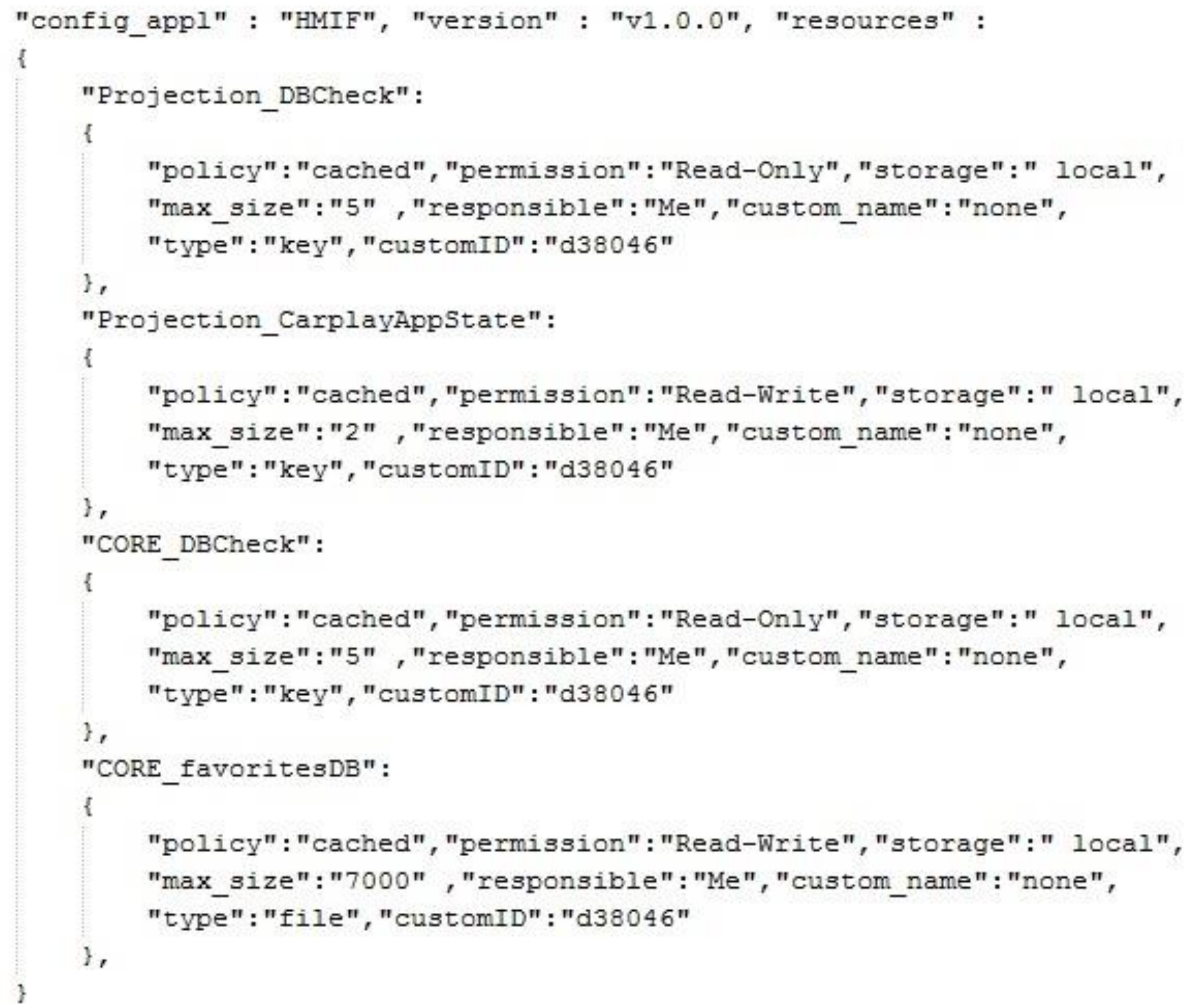


```xml
<File Name="favoritesDB" Description="Favorites Database storage path"
Storage="local" Policy="cached" Permission="Read-Write"
UserSpecific="yes" MaxFileSize="7000" dataType="Path">
<Path>/usr/kpit/Persistence/favorites.db</Path></File>
```

Resource-configuration.json

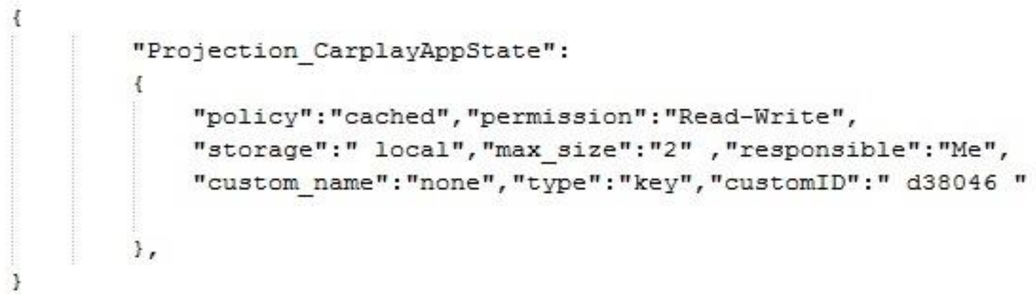


```json
{
    "CORE_favoritesDB":
    {
        "policy":"cached","permission":"Read-Write",
        "storage":" local","max_size":"7000" ,"responsible":"Me",
        "custom_name":"none","type":"file","customID":"d38046 "

    },
}
```

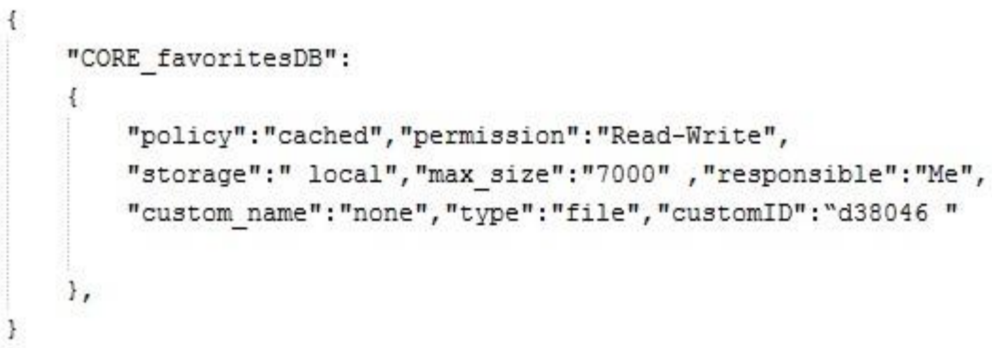


```
"config_appl" : "HMIF", "version" : "v1.0.0", "resources" :
{
    "Projection_DBCheck":
    {
        "policy":"cached","permission":"Read-Only","storage":" local",
        "max_size":"5" ,"responsible":"Me","custom_name":"none",
        "type":"key","customID":"d38046"
    },
    "Projection_CarplayAppState":
    {
        "policy":"cached","permission":"Read-Write","storage":" local",
        "max_size":"2" ,"responsible":"Me","custom_name":"none",
        "type":"key","customID":"d38046"
    },
    "CORE_DBCheck":
    {
        "policy":"cached","permission":"Read-Only","storage":" local",
        "max_size":"5" ,"responsible":"Me","custom_name":"none",
        "type":"key","customID":"d38046"
    },
    "CORE_favoritesDB":
    {
        "policy":"cached","permission":"Read-Write","storage":" local",
        "max_size":"7000" ,"responsible":"Me","custom_name":"none",
        "type":"file","customID":"d38046"
    },
}
```

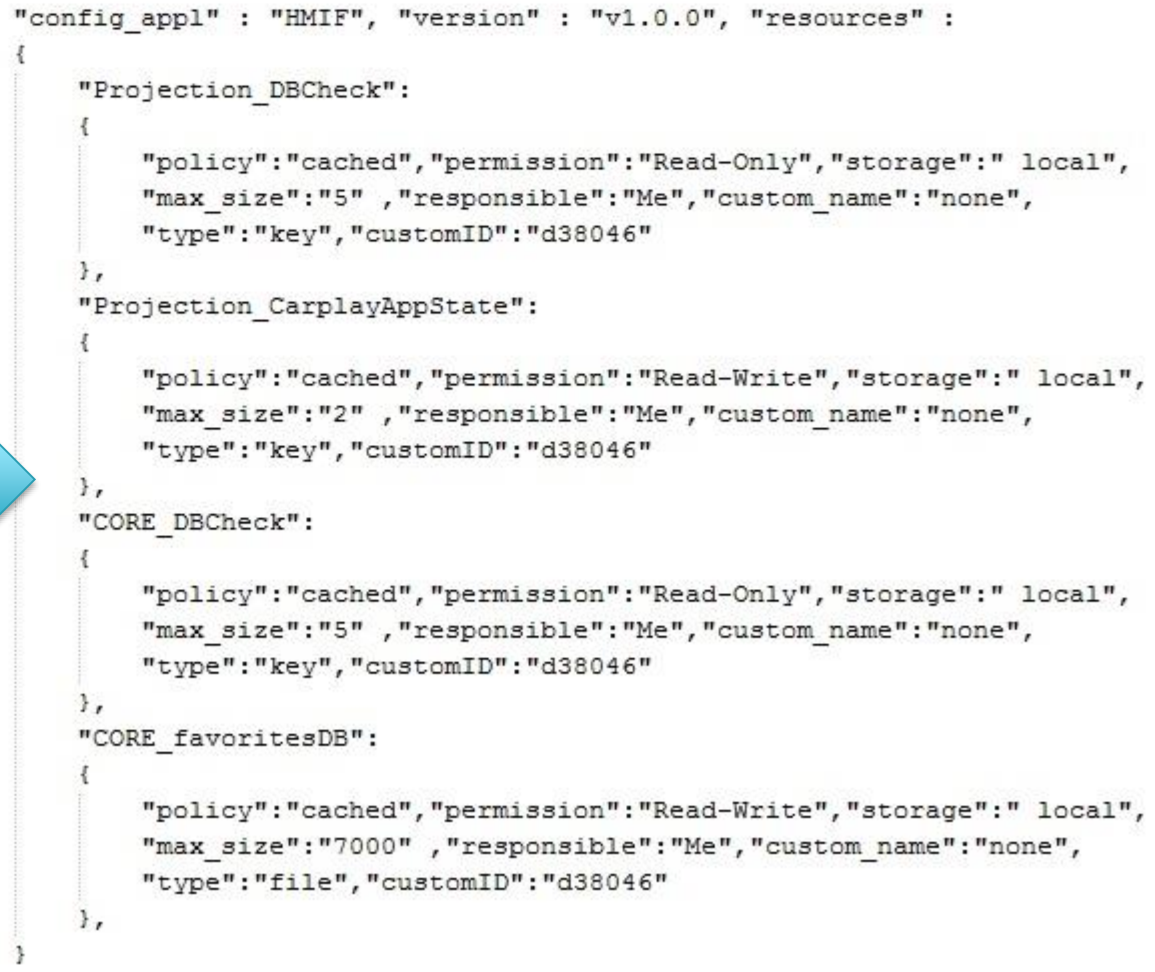
3.2 Approach – Post build utility merges jsons



```
{
        "Projection_CarplayAppState":
        {
            "policy":"cached","permission":"Read-Write",
            "storage":" local","max_size":"2" ,"responsible":"Me",
            "custom_name":"none","type":"key","customID":" d38046 "

        },
}
```

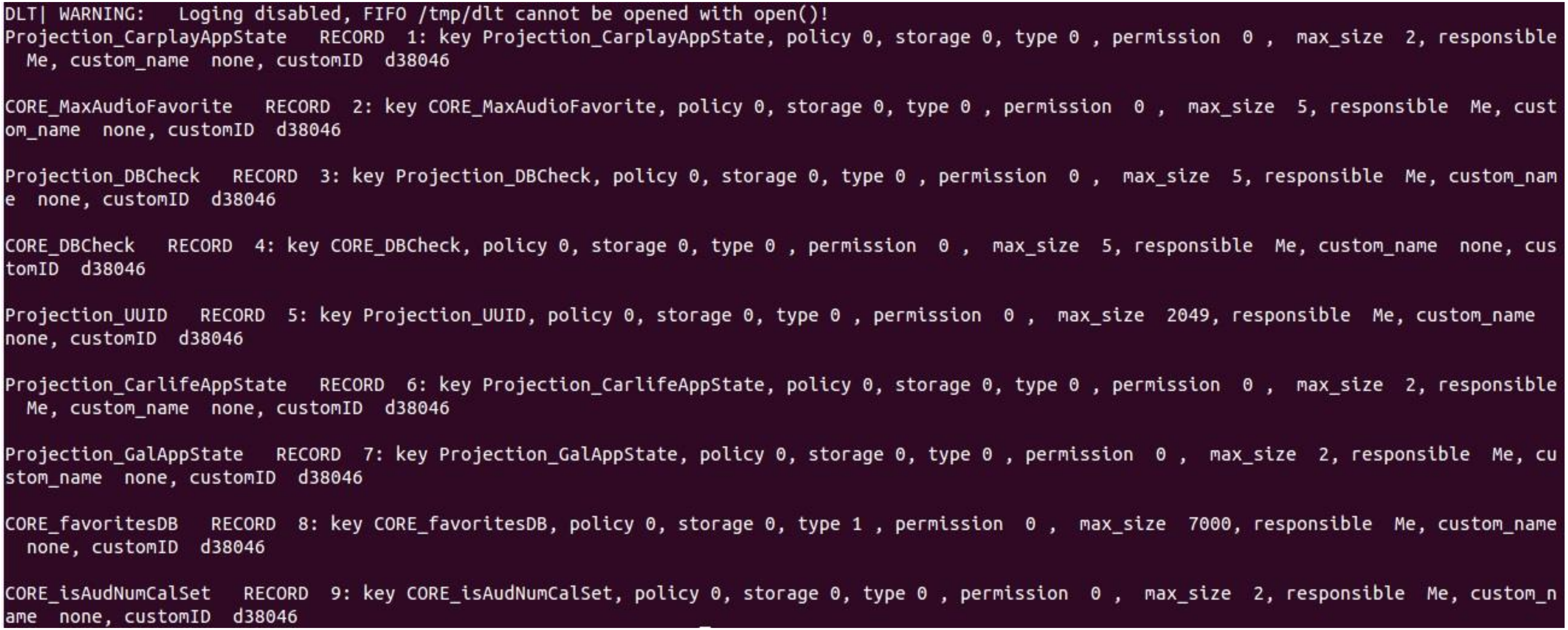


```
{
    "CORE_favoritesDB":
    {
        "policy":"cached","permission":"Read-Write",
        "storage":" local","max_size":"7000" ,"responsible":"Me",
        "custom_name":"none","type":"file","customID":“d38046 "

    },
}
```



```
"config_appl" : "HMIF", "version" : "v1.0.0", "resources" :
{
    "Projection_DBCheck":
    {
        "policy":"cached","permission":"Read-Only","storage":" local",
        "max_size":"5" ,"responsible":"Me","custom_name":"none",
        "type":"key","customID":"d38046"
    },
    "Projection_CarplayAppState":
    {
        "policy":"cached","permission":"Read-Write","storage":" local",
        "max_size":"2" ,"responsible":"Me","custom_name":"none",
        "type":"key","customID":"d38046"
    },
    "CORE_DBCheck":
    {
        "policy":"cached","permission":"Read-Only","storage":" local",
        "max_size":"5" ,"responsible":"Me","custom_name":"none",
        "type":"key","customID":"d38046"
    },
    "CORE_favoritesDB":
    {
        "policy":"cached","permission":"Read-Write","storage":" local",
        "max_size":"7000" ,"responsible":"Me","custom_name":"none",
        "type":"file","customID":"d38046"
    },
}
```

3.3 Approach – Multiple app keys under one process



```
DLT| WARNING:   Loging disabled, FIFO /tmp/dlt cannot be opened with open()!
Projection_CarplayAppState   RECORD  1: key Projection_CarplayAppState, policy 0, storage 0, type 0 , permission  0 ,  max_size  2, responsible
  Me, custom_name  none, customID  d38046

CORE_MaxAudioFavorite   RECORD  2: key CORE_MaxAudioFavorite, policy 0, storage 0, type 0 , permission  0 ,  max_size  5, responsible  Me, cust
om_name  none, customID  d38046

Projection_DBCheck   RECORD  3: key Projection_DBCheck, policy 0, storage 0, type 0 , permission  0 ,  max_size  5, responsible  Me, custom_nam
e  none, customID  d38046

CORE_DBCheck   RECORD  4: key CORE_DBCheck, policy 0, storage 0, type 0 , permission  0 ,  max_size  5, responsible  Me, custom_name  none, cus
tomID  d38046

Projection_UUID   RECORD  5: key Projection_UUID, policy 0, storage 0, type 0 , permission  0 ,  max_size  2049, responsible  Me, custom_name
none, customID  d38046

Projection_CarlifeAppState   RECORD  6: key Projection_CarlifeAppState, policy 0, storage 0, type 0 , permission  0 ,  max_size  2, responsible
  Me, custom_name  none, customID  d38046

Projection_GalAppState   RECORD  7: key Projection_GalAppState, policy 0, storage 0, type 0 , permission  0 ,  max_size  2, responsible  Me, cu
stom_name  none, customID  d38046

CORE_favoritesDB   RECORD  8: key CORE_favoritesDB, policy 0, storage 0, type 1 , permission  0 ,  max_size  7000, responsible  Me, custom_name
  none, customID  d38046

CORE_isAudNumCalSet   RECORD  9: key CORE_isAudNumCalSet, policy 0, storage 0, type 0 , permission  0 ,  max_size  2, responsible  Me, custom_n
ame  none, customID  d38046
```

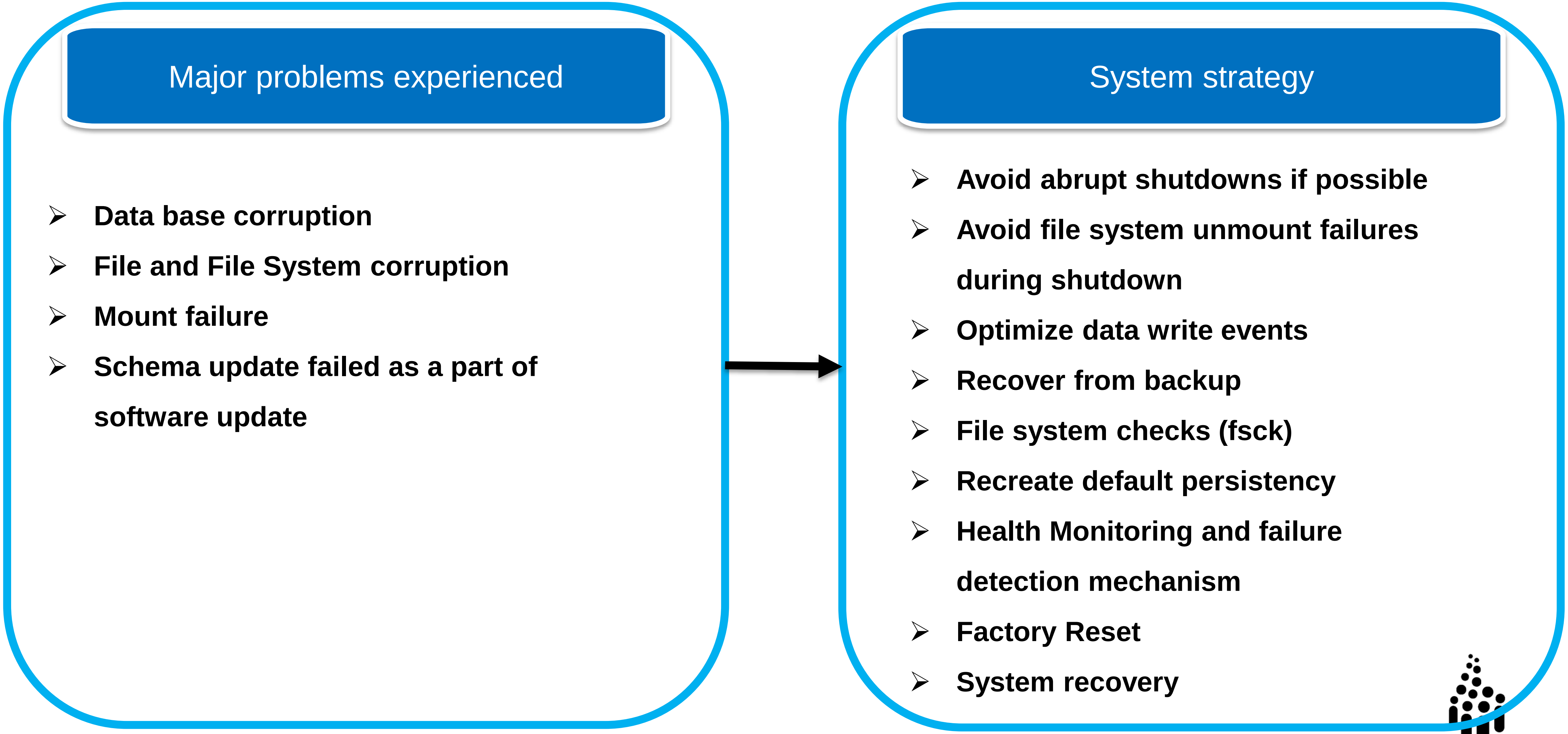
4 Approach :- Backup trigger management

- Change in write through persistable is backed up soon after.
- Change of cacheable data registers application for backup.
- At shutdown setup persistence service backups up all the applications registered for backup.

Custom Tools / Utilities

Sr.No.	Tools	Purpose
1	Packpersistencesetup	• Merging separate json tars of multiple applications into one under a process.
2	Persistencexml2cpp	• Autogenerating Persistable classes • Creating Json tars
3	JsonToDatabase	• To generate tar files that can be used for host testing.

Problem Experienced



Major problems experienced

- Data base corruption
- File and File System corruption
- Mount failure
- Schema update failed as a part of software update

System strategy

- Avoid abrupt shutdowns if possible
- Avoid file system unmount failures during shutdown
- Optimize data write events
- Recover from backup
- File system checks (fsck)
- Recreate default persistency
- Health Monitoring and failure detection mechanism
- Factory Reset
- System recovery

Next steps

- Enhancements
  - Serialization of Structure inside structures
  - Serialization of Structures strings as members.
  - Serialization of classes.
- Provision to store file buffer inside key-value pair to store file inside database
- Auto generation tool in Python instead of Java xtend
- Using persistence frame work on Android platform

Thank you!

Visit GENIVI at http://www.genivi.org or http://projects.genivi.org

Contact us: help@genivi.org