CCSP Provisioning and Management Overview

☒ **Draft**

☐ Baselined for Estimates

Author	
Authorized by	
Version	0.1
Date	May 16, 2014

Table of Contents

List of Figures

1 P&M Overview

P&M is a CCSP component that implements provisioning and management functionality of the device, supporting TR-181 (TR-157, TR-143) along with CCSP specific extensions.

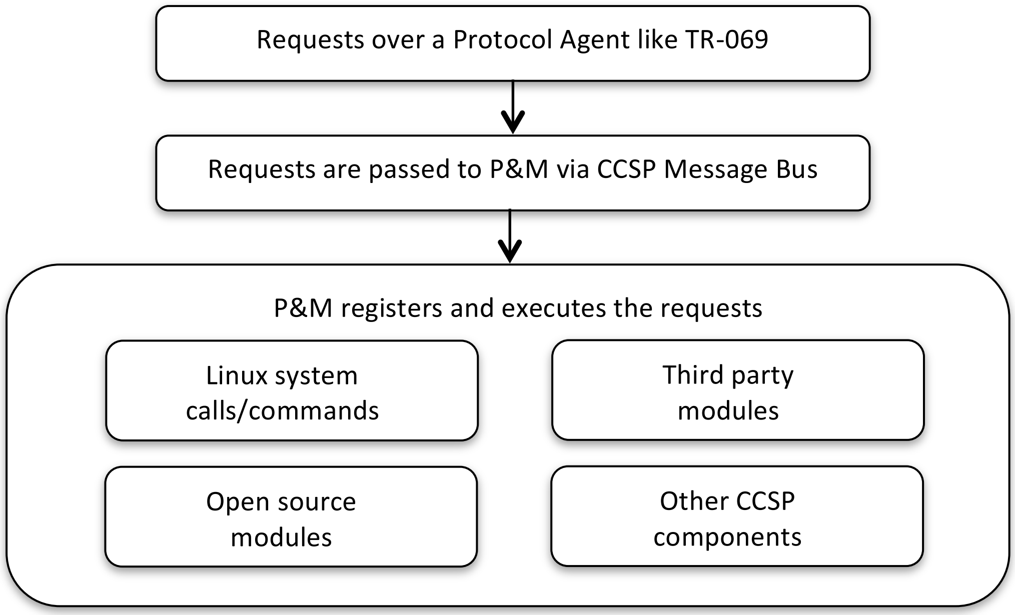
Requests over a Protocol Agent like TR-069
Requests are passed to P&M via CCSP Message Bus
P&M registers and executes the requests
Linux system calls/commands
Third party modules
Open source modules
Other CCSP components

Figure 1 P&M Overview

P&M module tracks any changes in the system for the attributes configured by service provider and sends upstream notifications. Any changes to these attributes will be reported to service provider using PA.

2 P&M Component Definitions

2.1 P&M Architecture

This is the architecture of P&M component:

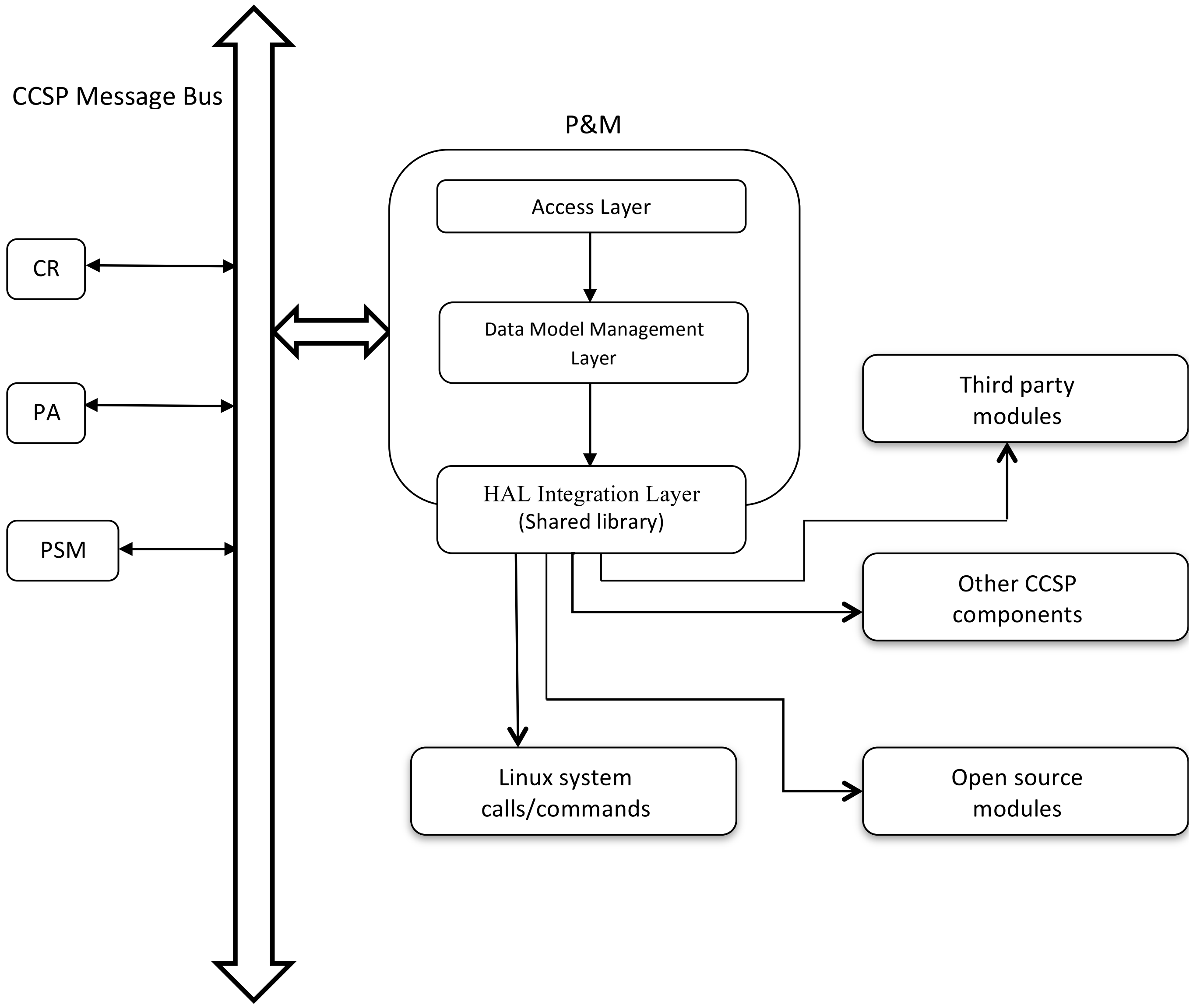

Figure 2 P&M Architecture Overview

P&M consist three layers:

1) Access Layer

Access Layer interfaces with the CCSP message bus and receives any set or get calls and passes them onto the DML layer to manage the data which is in request.

2) Data Model Management Layer

- Goal is to support a large number of data model objects for good scalability
- Data Model Management Layer loads all data model access APIs through a pre-defined XML file which contains description of data model objects and parameters.
- The data model implementation in shared library interfaces HAL Integration (backend) Layer by calling backend integration APIs.

3) P&M HAL Integration (backend) Layer, a.k.a, component specific HAL

This is the layer making calls to underlying Linux system calls/commands, third party modules, open source modules and other CCSP components to execute the requests. This layer will be more component specific and will be providing APIs to CCSP so as to manage a particular hardware module of the system.

The implementation of APIs is responsible to convert the user space calls into Device IOCTL (kernel space) accordingly.

3 Functional Structure

3.1 CCSP Message Bus APIs

P&M DBUS object path is “/com/cisco/spvtg/ccsp/pam”.

P&M supports following CCSP Message Bus APIs:

1) initialize
2) finalize
3) getParameterNames
4) getParameterValues
5) setParameterValues
6) setCommit
7) setParameterAttributes
8) getParameterAttributes
9) AddTblRow
10) DeleteTblRow
11) busCheck

3.2 CCSP Message Bus APIs required from other components

P&M requires following CCSP Message Bus APIs.

From CR:

1) registerCapabilities
2) unregistername_space
3) unregisterComponent
4) discComponentSupportingNamespace
5) checkNamespaceDataType
6) SendDeviceProfileChangeSignal

From PA:

7) SendParameterValueChangeSignal

From PSM:

8) getParameterValues
9) setParameterValues
10) getParameterNames

4 P&M System Flow

4.1 Parameter set/get flow in P&M

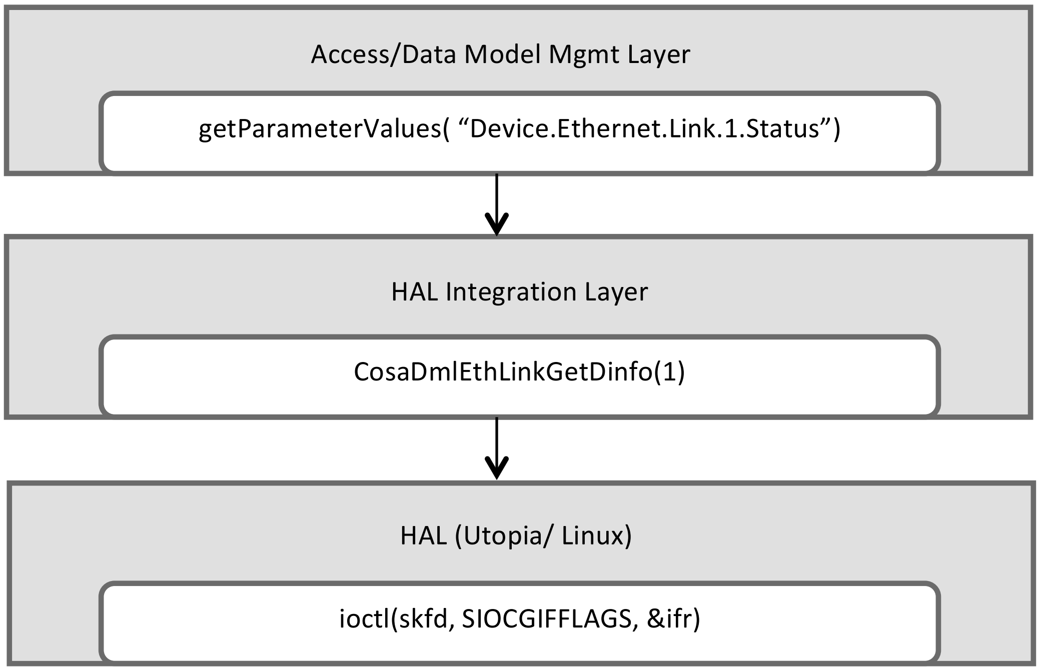
Access/Data Model Mgmt Layer
getParameterValues(“Device.Ethernet.Link.1.Status”)
HAL Integration Layer
CosaDmlEthLinkGetDinfo(1)
HAL (Utopia/ Linux)
ioctl(skfd, SIOCGIFFLAGS, &ifr)

Figure 3 Parameter Get Flow

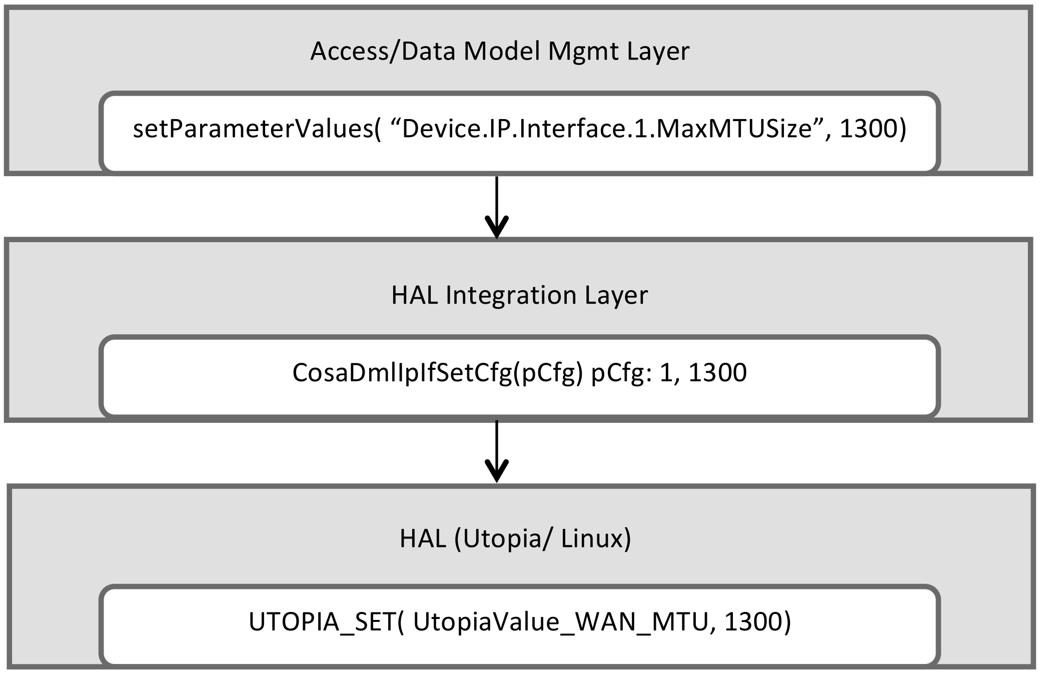
Access/Data Model Mgmt Layer
setParameterValues(“Device.IP.Interface.1.MaxMTUSize”, 1300)
HAL Integration Layer
CosaDmlIpIfSetCfg(pCfg) pCfg: 1, 1300
HAL (Utopia/ Linux)
UTOPIA_SET(UtopiaValue_WAN_MTU, 1300)

Figure 4 Parameter Set Flow

4.2 P&M boot-up flow

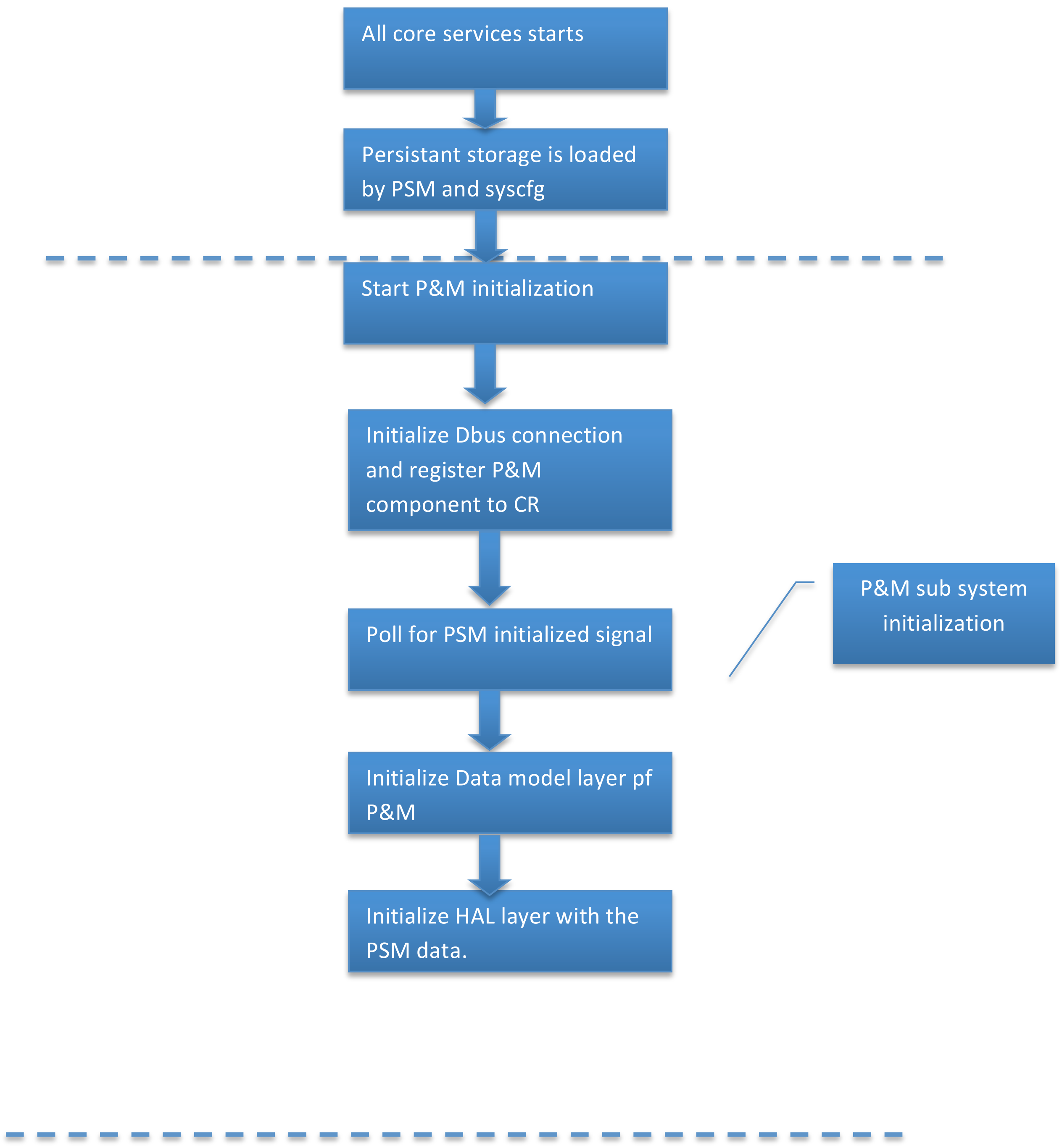
All core services starts
Persistant storage is loaded by PSM and syscfg
Start P&M initialization
Initialize Dbus connection and register P&M component to CR
Poll for PSM initialized signal
Initialize Data model layer pf P&M
Initialize HAL layer with the PSM data.
P&M sub system initialization

Figure 5 P&M boot-up flow

4.3 Instance Number Operation Flow

Object instance number is used to uniquely identify the object, even across reboots. It is a special value P&M has to maintain, though it is not shown as a parameter. Instance number is generated when a new object is added, or during initialization, if no existing instance number is found. Generated instance number is saved in persistent configuration. Once the object is removed, the instance number is usually not reused.

Instance Number Operation Flow, when system boots up:

- Data Model Access Layer Starts and Loads DM adapter. DM adapter initialize its backend API
- DM adapter calls CosaDmlXyzGetEntry() to retrieve each object
- Backend reads saved configuration from persistent storage
- DM adapter checks, if Instance Number/Alias exists
    - If exists then get saved configuration from persistent storage
    - If not fund
        - DM adapter generates Instance Number/Alias
        - DM adapter calls CosaDmlXyzSetValues()
        - Backend saves Instance Number/Alias into persistent storage

When an object is added during runtime:

- DMM receives ACS RPC to add an object. DM adapter creates an object internally and set the parameters with default values. DM adapter picks a unique instance number (and alias).
- DMM returns the instance number. DMM receives Set with filled parameters.
- DM adapter calls backend API to Add Entry. Instance Number and Alias (if any) are filled in.
- Backend saves the configuration, including Instance Number and Alias, and take respective actions.

5 Glossary

The following list describes acronyms and definitions for terms used throughout this document:

- **CCSP**: **Common Component Software Platform**
- **P&M**: **Provisioning and Management**
- **PA**: **Protocol Agent**
- **CR: Component Registrar**
- **PSM: Persistent Storage Manager**
- **TDM: Test and Diagnostics Manager**