CcspPandM

Overview

PandM is a CCSP component that implements core provisioning and management functionality of the device. Its primary role is to respond to commands from other CCSP components and protocol agents that need to set or query variables pertaining to provisioning and management. Its interface to other components uses the CCSP Message Bus interface and is based around a data model derived from TR-181 (TR-157, TR-143) along with CCSP specific extensions. PandM is a Key module which holds parameters related to many key services like: dhcpv6, LAN, DeviceInfo etc. It maintains a TR181 data model XML file with dbus object path as /com/cisco/spvtg/ccsp/pam . Has a layered architecture which makes it less complex for GET/SET implementation.

P&M interfaces with Protocol Agent, Cr, PSM and other common components via the Message Bus Interface.Initialize On the lower side, P&M interfaces with the HAL layer modules, Ethernet, WiFi, MoCA, IP, DHCP client/server, etc.

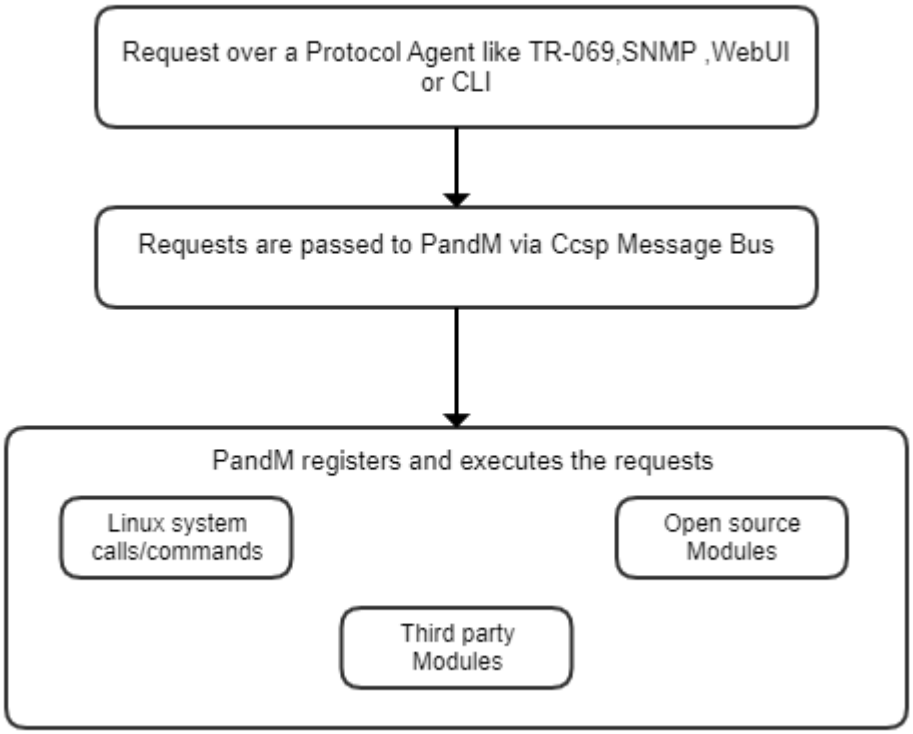


Figure 1 PandM Overview

system calls - Example

```
sysevent set ipv4-tsip_IPAddress %s
sysevent set ipv4-tsip_Subnet %s
sysevent set ipv4-tsip_Gateway %s
sysevent set ipv4-resync_tsip %d
system("sysevent set wan_staticip-status stopped");
system("sysevent set wan_staticip-status started");
```

Architecture

This is the architecture of PandM component:

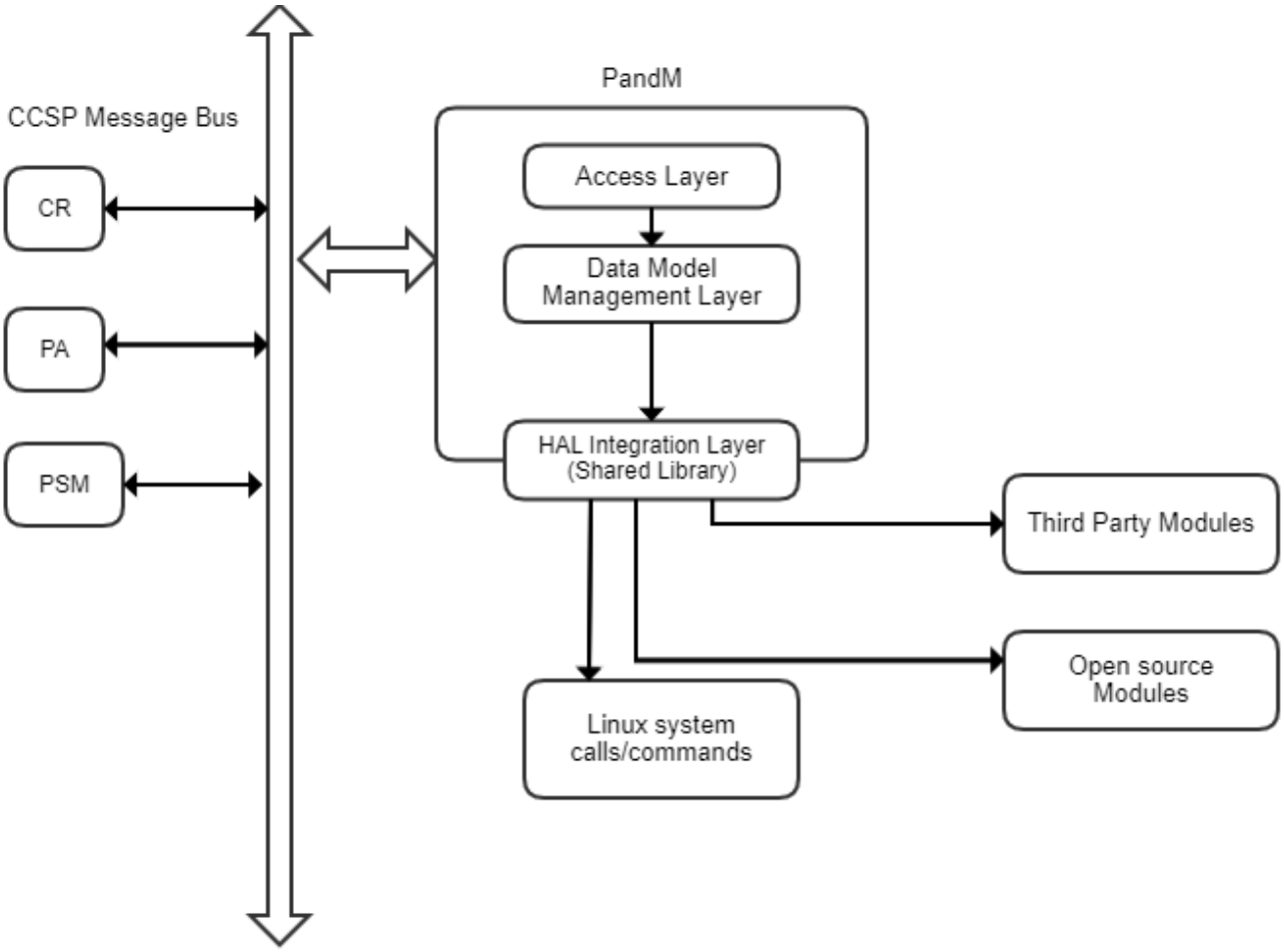


Figure 2 PandM Architecture Overview

PandM consist of 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

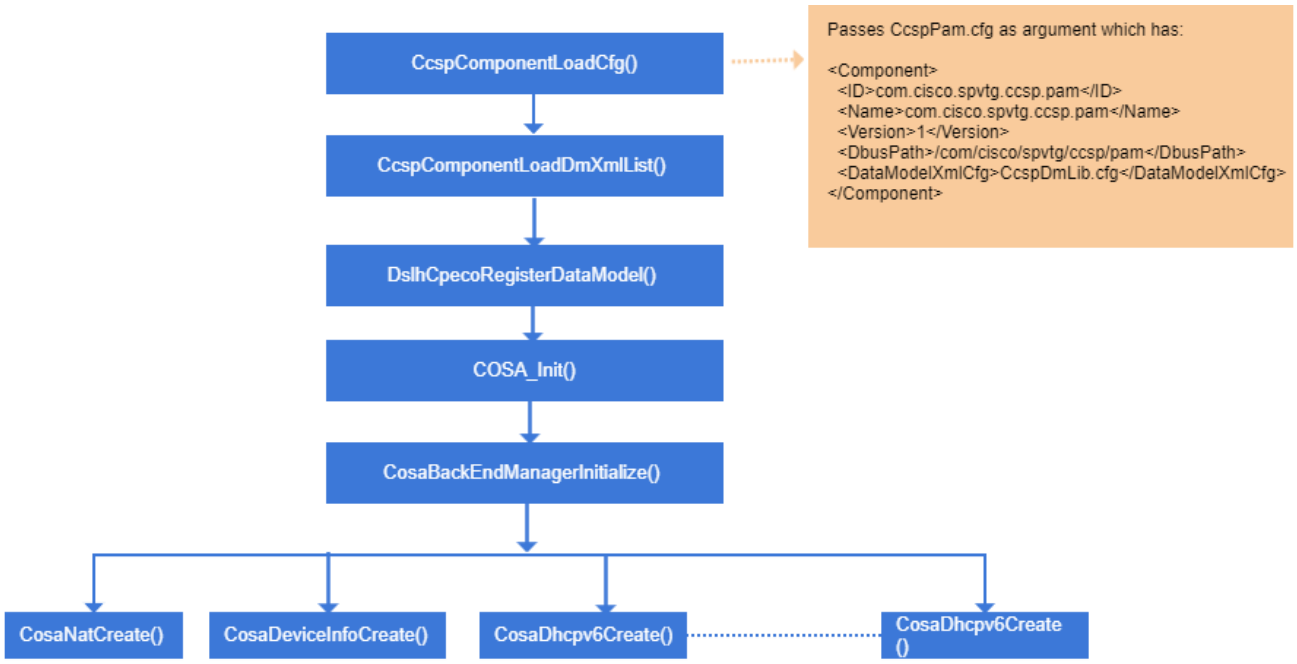
- Goal is to support a large number of data model objects for good scalability
- Data Model Management Layer loads all data model access APIs through a pre-defined XML file which contains description of data model objects and parameters.
- The data model implementation in shared library interfaces HAL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

3) PandM 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.

Code Flow



PandM DM Objects on Initialization

CosaNatCreate()
CosaProcStatusCreate()
CosaDeviceInfoCreate()
CosaUserinterfaceCreate()
CosaEthernetCreate()
CosaUsersCreate()
CosaDdnsCreate()
CosaFirewallCreate()
CosaSecurityCreate()
CosaIPCreate()
CosaDhcpv4Create()
CosaHostsCreate()
CosaDNSCreate()
CosaRoutingCreate()
CosaBridgingCreate()
CosaIFStackCreate()
CosaPPPCreate()
CosaDhcpv6Create()
CosaDeviceControlCreate()
CosaIPv6rdCreate()
CosaRACreate()
CosaNeighdiscCreate()
CosaMldCreate()

CosaDiagnosticsCreate()
CosaTimeCreate()
CosaUpnpCreate()
TR181_ParentalControlCreate()
CosaRLogCreate()
CosaGreCreate()
CosaGreTunnelCreate()
CosaCGreCreate()
CosaHotspotCreate()
CosaFileTransferCreate()
CosaTSIPCreate()
CosaDeviceFingerprintCreate()

Object/Parameter supported by P&M

Device.DeviceInfo.
Device.GatewayInfo.
Device.InterfaceStack.{i}.
Device.Ethernet.
Device.MoCA.Interface.
Device.Bridging.Bridge.{i}.
Device.PPP.Interface.{i}.
Device.IP.Interface.{i}.
Device.Routing.Router.{i}.
Device.NAT.
Device.DHCPv4.
Device.DHCPv6.
Device.Users.

Functional Structure

CCSP Message Bus APIs

PandM DBUS object path is “/com/cisco/spvtg/ccsp/pam”.

PandM supports following CCSP Message Bus APIs:

initialize	Initializes NAT, DeviceInfo, Firewall, Ethernet, IP, Hosts, MoCAand Bridging.
finalize	finalize
getParameterNames	API returns parameter names associated with the supported objects.
getParameterValues	Returns values of the parameters queried for.
setParameterValues	API to the parameters with the values provided
setCommit	Mainly used in bulk/atomic set operations when more than one parameter is involved.

setParameterAttributes	API to set notifications status for the parameters.
getParameterAttributes	API to get notifications status for the parameters.
AddTblRow	API to add rows to the objects of table type.
DeleteTblRow	API to delete rows to the objects of table type.
busCheck	API is used in diagnostic mode.

CCSP Message Bus APIs required from other components

PandM 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

PandM System Flow

Parameter set/get flow in PandM

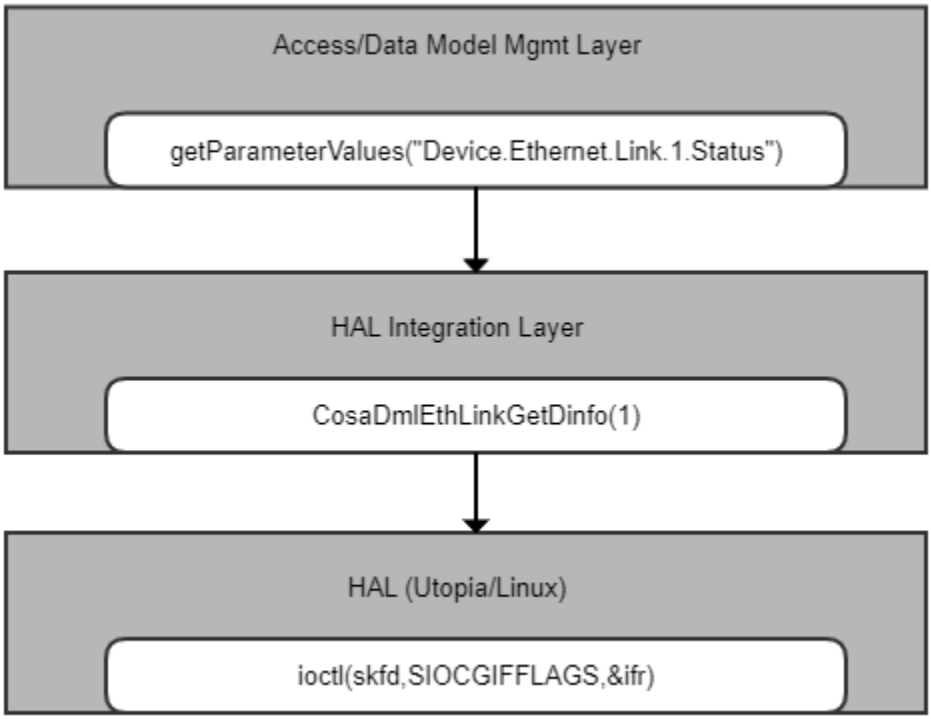


Figure 3 Parameter Get Flow

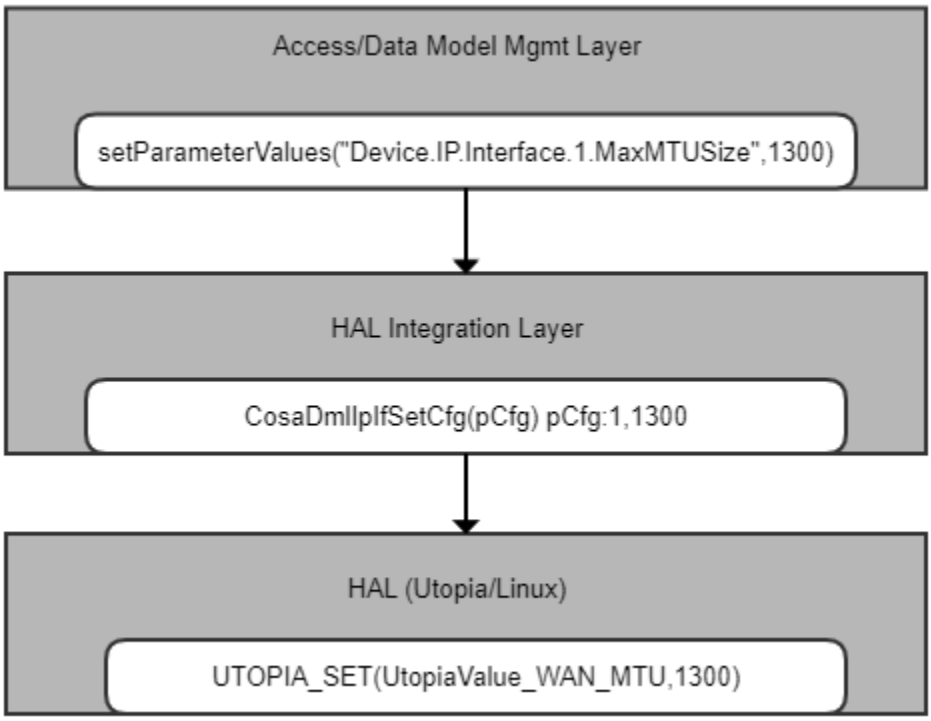


Figure 4 Parameter Set Flow

PandM boot-up flow

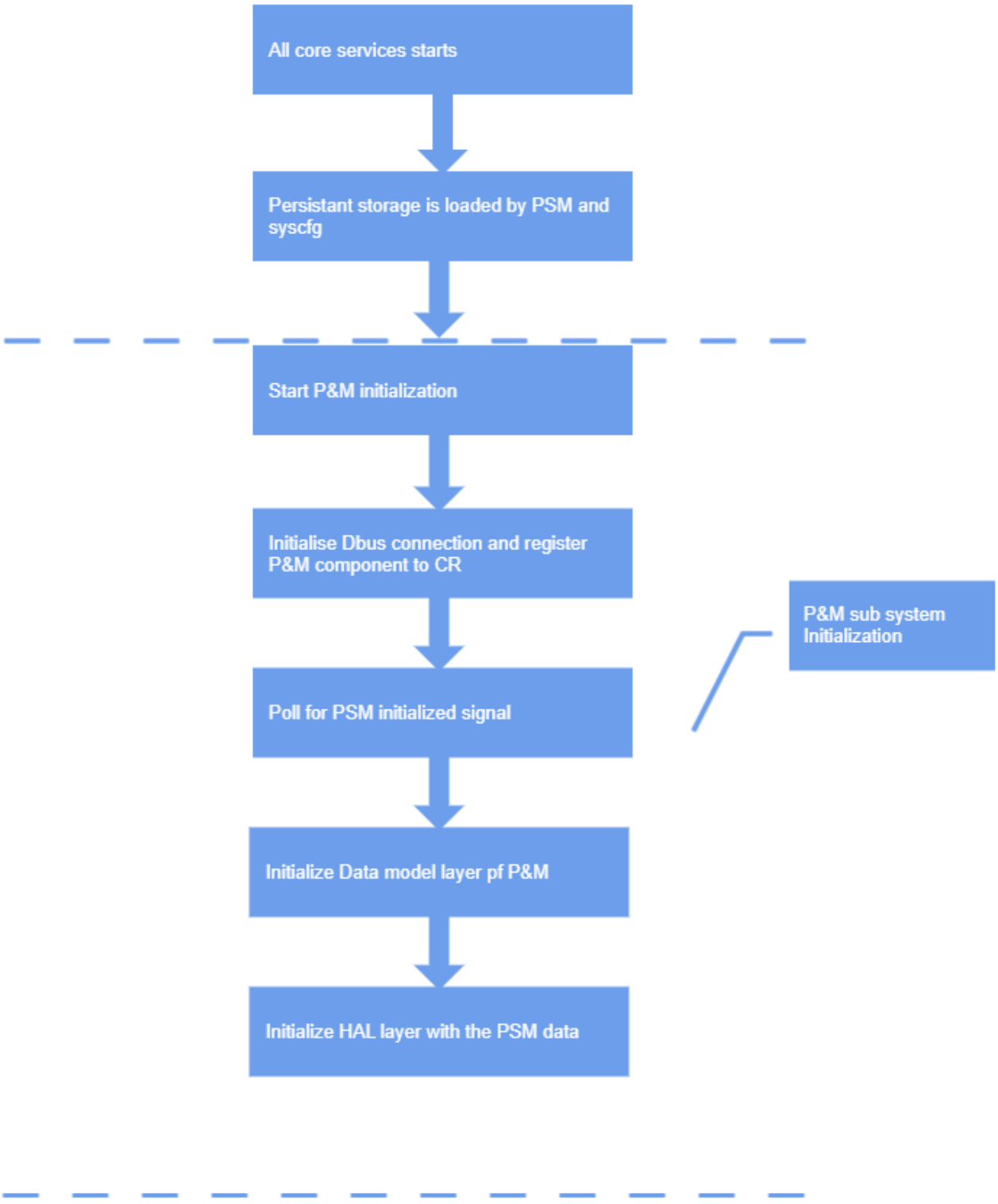


Figure 5 PandM boot-up flow

Instance Number Operation Flow

Object instance number is used to uniquely identify the object, even across reboots. It is a special value PandM 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 response 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.