Dialogic Open System Release

LINUX OAM

High Level Architecture Document version 1.0

TABLE OF CONTENTS

Dialogic Open System Release 1
LINUX OAM 1
High Level Architecture Document version 1.0 1
1. Introduction 5
1.1. Scope 5
1.2. Glossary 5
2. Revision History 5
3. The OAM Binaries and Libraries 6
3.1. Configuration 7
3.1.1. ConfigGen project 7
3.1.2. boardinfo project 7
3.1.3. configurator project 7
3.1.4. confslot project 7
3.1.5. mkcfg project 7
3.2. Initialization, detection, and download 8
3.2.1. DeviceNameDoler project 8
3.2.2. dlgpmacdetector project 8
3.2.3. dlgpmacdetectorexe project 8
3.2.4. dlgpmacinitializerexe project 8
3.2.5. dlgpmacinitializer project 8
3.2.6. dm3post project 8
3.2.7. dm3start project 8
3.2.8. downloader project 8
3.2.9. genload project 8
3.2.10. libdm3be project 9
3.2.11. msilineactivate project 9
3.2.12. regvox project 9
3.2.13. scripts project 9
3.2.14. sctsassign project 9
3.2.15. sctscfg project 9
3.2.16. sctsdtdxag project 9
3.2.17. shutboard project 9
3.3. Event management, fault detection and management, and diagnostics 10
3.3.1. dlgdm3autodump project 10
3.3.2. dlgdm3diagagent project 10
3.3.3. dlgerrors project 10
3.3.4. dlgeventservice project 10
3.3.5. dlgfaultd project 10
3.3.6. dlgosloggerextension project 10
3.3.7. dlgpmacbsp project 10
3.3.8. dlgpmacfdsp project 11
3.3.9. dlgsnapshot project 11
3.3.10. dlgspwrfaultdetector project 11

3.3.11. dlgsysmonitor project 11
3.3.12. dlgsyslogger project 11
3.3.13. evtgen project 11
3.3.14. dm3clump project 11
3.3.15. udd project 11
3.4. Clocking and timeslot management 12
3.4.1. cdaemon_ctl project 12
3.4.2. clockdaemon project 12
3.4.3. clockingserver project 12
3.4.4. clockstatus project 12
3.4.5. dlgdm3clockingagent project 12
3.4.6. dlgdasilite project 12
3.4.7. dlgpmactimeslotassigner project 12
3.4.8. dlgpmactimeslotassignerexe project 12
3.4.9. dlgspwrclockingagent project 12
3.4.10. dlgtimeslot project 13
3.4.11. tbmodify project 13
3.4.12. tbparms project 13
3.5. SNMP 14
3.5.1. adl project 14
3.5.2. aldoamipc project 14
3.5.3. boardserver_oamipc project 14
3.5.4. libagentscommon_oamipc project 14
3.5.5. netsnmp_agents_oamipc project 14
3.6. Miscellaneous infrastructure and device management copmonents 15
3.6.1. RegisterEntity project 15
3.6.2. device_mapper project 15
3.6.3. devmaptools project 15
3.6.4. dlgoslayer project 15
3.6.5. dlgpmacadmininterface project 15
3.6.6. dlgpmacparser project 15
3.6.7. dlgpmacutilities project 16
3.6.8. dlgproductagentmgr project 16
3.6.9. libasi project 16
3.6.10. libdm3pplow project 16
3.6.11. libdm3r4cleo project 16
3.6.12. libsysteminfo project 16
3.6.13. pblocator project 16
3.6.14. teelogger project 16
3.6.15. usbu project 17
3.6.16. OAMOSSL project 17
3.6.17. OAMNaming project 17
3.6.18. OAMMarshall project 17
3.6.19. OAMTransport project 17
3.6.20. OAMEventService project 17
4. Architectural Diagrams 18

Table of Figures

Figure 1 Overall OAM architecture for DM3 and IPT (PMAC) 18
Figure 2 Overall architecture for clocking 19
Figure 3 Overall architecture for HotSwap functionality 20
Figure 4 Overview of data stored and managed by Device Mapper 21
Figure 5 Overall architecture for fault detection and handling 22
Figure 6 Component interactions between components in OAMIPC architecture 23
Figure 7 Data flow from OAMIPC clients and servers 24
Figure 8 Inputs to Genload and components 25

1. Introduction

This document was written to provide the users of the Dialogic Open System Release OAM source code with detailed understanding of the various libraries and executables available for download.

1.1. Scope

This document provides a high level overview of the OAM Libraries supported under the Dialogic Open System Release project. Each library and executable is outlined with a general architecture overview followed by descriptions of each source file's content.

1.2. Glossary

Term or Acronym	Description
OAM (aka OA&M)	Operations, Administration, and Management
DM3	Dialogic 3rd generation media processing cards
PMAC	The internal name for the IPT line or products.
OAMIPC	OAM Inter Process Communication

2. Revision History

Revision	Reason for Changes
1.0	**Initial Release**

3. The OAM Binaries and Libraries

Listed in this section are a variety of software projects that make up the OAM subsystem. These projects contain the source code (mostly C and C++) for the OAM software components. These components take the form of either libraries (.so) or binary executables.

In the tarball, each project is contained in a subdirectory below /cm/vobs/oam/components. For instance, the files for the ConfigGen project can be found in /cm/vobs/oam/components/ConfigGen.

Below each project directory, there is a subdirectory called src. This directory contains the source code for the project. For instance, the source code for the ConfigGen project can be found in /cm/vobs/oam/components/ConfigGen/src.

The output of building a project is put in either /cm/vobs/binaries/external/unix/ia32/gcc3.4.4/pci/release/bin for binaries or /cm/vobs/binaries/external/unix/ia32/gcc3.4.4/pci/release/lib for libraries. Note that this example assumes the use of the 3.4.4 version of GCC.

It should be noted that there are OAM components that are part of another tarball called oam_common. The points made above apply to oam_common as well, however the directories are under /cm/vobs/oam_common.

To aid readability, the component projects are grouped together in areas of functionality.

3.1. Configuration

This section contains the projects related to system and board configuration.

3.1.1. ConfigGen project

This is an executable that is used for trunk configuration on boards such as the DMVB and DTI16.

3.1.2. boardinfo project

This is a library that manages a database of information regarding boards used in conjunction with System Release software. It contains information such as the hardware platform, product ID, board family, and board model. This library is currently used by the usbu (unsupported board utility).

3.1.3. configurator project

This is an executable that is used to configure boards installed in the system. Using this utility, Springware, DM3, and PMAC boards may be configured.

3.1.4. confslot project

This is an executable that is run during system initialization. It is used to configure and initialize any conferencing components that may be present on any of the installed hardware.

3.1.5. mkcfg project

This is an executable that can be used to configure the configuration file for Springware boards. The configurator is the preferred method for configuring all boards, including Springware boards.

3.2. Initialization, detection, and download

This section contains the projects related to system initialization, board detection and board download.

3.2.1. DeviceNameDoler project

This is a library that is used to allocate device names such as dxxxB1C1 or dtiB1T1.

3.2.2. dlgpmacdetector project

This is a library that is used to detect IPT boards that are physically present in a system.

3.2.3. dlgpmacdetectorexe project

This is an executable that makes use of the dlgpmacdetector library. It invokes the library calls necessary to detect IPT boards.

3.2.4. dlgpmacinitializerexe project

This is an executable that makes use of the dlgpmacinitializer library.

3.2.5. dlgpmacinitializer project

This is a library that provides functions for initializing IPT boards. Among the functionality provided, it registers IPT devices with device mapper.

3.2.6. dm3post project

This is an executable that is use as part of the fault detection framework for DM3 hardware. It is used to invoke a POST (Power On Self Test) of the board and report the results.

3.2.7. dm3start project

This is an executable that is used to start a DM3 based board after the firmware has been downloaded and configuring it.

3.2.8. downloader project

This is an executable that is responsible for downloading DM3 based boards.

3.2.9. genload project

This is an executable that is responsible for downloading Springware based boards.

3.2.10. libdm3be project

This is a library that is responsible for enumerating all of the DM3 boards on a system.

3.2.11. msilineactivate project

This is an executable that is run during system startup. It configures and initializes any station interface components that may be present on any installed hardware.

3.2.12. regvox project

This is an executable that is used in conjunction with genload to start and configure Springware boards.

3.2.13. scripts project

This project contains the shell scripts that are used to start and stop the overall system.

3.2.14. sctsassign project

This is an executable that is used to assign SC Bus timeslots to Springware based boards.

3.2.15. sctscfg project

This is an executable that is used to configure SC Bus timeslots for Springware based boards.

3.2.16. sctsdtdxag project

This is an executable that is used to assign SC bus timeslots to the front end of Springware based boards.

3.2.17. shutboard project

This is an executable that is used to shutdown DM3 boards. This is done as part of stopping a system that has DM3 based boards installed.

3.3. Event management, fault detection and management, and diagnostics

This section contains the projects related to event management, fault detection and management, and diagnostics.

3.3.1. dlgdm3autodump project

This is a library that is used as part of the fault management architecture. Any errors that are generated due to a DM3 board crash are reported though this component.

3.3.2. dlgdm3diagagent project

This is a library that is used as part of the fault management architecture. This agent is responsible for starting, stopping, and querying the state of the DM3 diagnostic software. This includes downloading diagnostic firmware if necessary.

3.3.3. dlgerrors project

This is a library that is used by some modules to manage error messages and the text associated with them.

3.3.4. dlgeventservice project

This is a set of libraries that are used in conjunction with the Event Service. The two libraries are for consuming events and for generating (supplying) events. Modules that supply or consume events use these libraries.

3.3.5. dlgfaultd project

This is an executable that runs as a daemon. It manages the execution of fault detectors on the system for all installed boards.

3.3.6. dlgosloggerextension project

This is a library that provides logging capabilities that are used by some of the modules in this subsystem.

3.3.7. dlgpmacbsp project

This is a library that acts as a clocking agent for Dialogic IPT boards. It is through this clocking agent that clocking is managed for these boards.

3.3.8. dlgpmacfdsp project

This is a library that implements the fault detector for IPT boards.

3.3.9. dlgsnapshot project

This is an executable which performs fault detection and will capture debug information from the hardware. This information can be used to debug a board crash.

3.3.10. dlgspwrfaultdetector project

This is a library that provides the fault detection agent for Springware based boards.

3.3.11. dlgsysmonitor project

This is a daemon process which runs all of the diagnostic agents and monitors for errors. If an error occurs, things like dlgsnapshot are run to capture debug information.

3.3.12. dlgsyslogger project

This is a library that provides logging capabilities for all components in this Subsystem (OAM). It interacts with RTF (Runtime Trace Facility) which is in the diagnostic subsystem.

3.3.13. evtgen project

This is an executable that can be used to test the event service functionality. It allows events of any type to be generated and sent through the event delivery mechanism.

3.3.14. dm3clump project

This is a library that provides the fault detection agent for DM3 based boards.

3.3.15. udd project

This is diagnostic utility for Springware based hardware. It is made up of a library and a set of java classes in a JAR file. This is commonly referred to as the "Universal Dialogic Diagnostic".

3.4. Clocking and timeslot management

This section contains the projects related to clocking and timeslot management.

3.4.1. cdaemon_ctl project

This is an executable which administers the clocking daemon.

3.4.2. clockdaemon project

This is a daemon process which manages the clocking configuration of any installed boards. It performs fallback operations.

3.4.3. clockingserver project

This is an executable which loads all clocking agents and runs them.

3.4.4. clockstatus project

This is an executable which is used to query status of the clocking subsystem.

3.4.5. dlgdm3clockingagent project

This is a library that is used as part of the clocking management architecture. It manages the clocking for DM3 based boards.

3.4.6. dlgdasilite project

This is a library which provides an alternative to the clocking API.

3.4.7. dlgpmactimeslotassigner project

This is a library that provides functions for assigning timeslots from the CTBus to IPT boards.

3.4.8. dlgpmactimeslotassignerexe project

This is an executable that makes use of the dlgpmactimeslotassigner library to assign timeslots from the CTBus to IPT boards.

3.4.9. dlgspwrclockingagent project

This is a library that provides the clocking agent for Springware based boards.

3.4.10. dlgtimeslot project

This is a client server pair used to manage timeslots on the system. It is through this daemon that timeslots are assigned and unassigned from the CTBus to installed boards.

3.4.11. tbmodify project

This is a set of executables called tbmodify and tbassign. These are used to manage the timeslots assigned for DM3 based boards.

3.4.12. tbparms project

This is an executable that retrieves timeslot information from the DM3 configuration files and puts it into the database managed by device mapper.

3.5. SNMP

This section contains the projects related to SNMP.

3.5.1. adl project

These are libraries used by the SNMP components to query technology specific information to be reported via SNMP. This includes information about channel status, link status, and SRL statistics.

3.5.2. aldoamipc project

This is a server which is implemented using the OAMIPC infrastructure. Components in the adl project make use of this server.

3.5.3. boardserver_oamipc project

This is a daemon process that acts as the SNMP server.

3.5.4. libagentscommon_oamipc project

This is a set of libraries that make up the SNMP agents. These agents are based on the OAMIPC architecture.

3.5.5. netsnmp_agents_oamipc project

This provides components to support the SNMP functionality.

3.6. Miscellaneous infrastructure and device management copmonents

This section contains the basic infrastructure and device management components.

3.6.1. RegisterEntity project

This is a library used by modules which allocate resources on a system such as files. This information could be used during an uninstallation procedure to clean up log and configuration files.

3.6.2. device_mapper project

This is a library that manages devices and their attributes, such as device name, and associates them with a physical device. At runtime, device mapper is invoked as part of opening a device.

3.6.3. devmaptools project

This is a set of executables that allow attributes to be set and retrieved via the shell rather than invoking library functions from a C or C++ application.

3.6.4. dlgoslayer project

This is a library that contains OS abstractions for functions that are OS specific. Among all of the functions in this library are functions to sleep, load dynamically loaded libraries, and query for process ID.

3.6.5. dlgpmacadmininterface project

This is a library that is used by the utilities that are used to administer IPT boards. This library interfaces
to lower level functions.

3.6.6. dlgpmacparser project

This is a library that provides functions to parse the configuration files associated with IPT boards. It is used by other executables that administer IPT boards.

3.6.7. dlgpmacutilities project

This is a library that provides utility functions for use by other libraries and executables used to administer IPT boards. Among the functions are those to translate between board ID to PCI bus and slot and to determine the directory specified to store configuration information.

3.6.8. dlgproductagentmgr project

This is a library that provides the functionality to manage all agents on the system. These agents are responsible for fault detection and clocking management on a variety of technologies (DM3, Springware, etc.).

3.6.9. libasi project

This is a library which provides some wrapper functions for some OS routines, such as shared memory management and semaphore management.

3.6.10. libdm3pplow project

This is a library that provides functions and classes to send messages to the firmware on DM3 boards.

3.6.11. libdm3r4cleo project

This is a library that provides capabilities to enumerate the devices on DM3 boards and present them as "R4" devices. This allows both Springware and DM3 devices to be handled the same way using the R4 API.

3.6.12. libsysteminfo project

This is a library that provides a mechanism to query information about boards installed on a system. Springware, DM3, and IPT boards are handled by this library.

3.6.13. pblocator project

This is an executable that is used to enumerate the boards installed on a system and report their current status. It will list all boards on the system regardless of technology (DM3, Springware, etc.) and depending on the options passed may not require the boards to be downloaded and started. This is commonly referred to as the "Physical Board Locator". It also contains the functionality of a utility that was once called "listboards".

3.6.14. teelogger project

This is a utility used by some of the initialization scripts to route output to both the screen and to a log file.

3.6.15. usbu project

This is an executable that is called by the initialization script. If any boards are installed that are not supported by the System Release, it alerts the user via screen output. This is commonly referred to as the "unsupported board utility".

3.6.16. OAMOSSL project

This is the "OAM Operating System Layer" component. This component is implemented as a library (libOAMOSSL.so) and has multiple purposes:

- Provide OS independent interfaces for OS specific operations, such as threading, synchronization, and inter-process communication (IPC).
- Provide higher level socket operations for use by other OAM_COMMON components. These operations take care of establishing socket connections and managing data transfers that require multiple calls to send() or recv().

3.6.17. OAMNaming project

This component is implemented as a library (libOAMNaming.so) and provides an interface to the configuration file (dlgOamNaming.cfg) which specifies the ports used by OAMIPC based servers.

3.6.18. OAMMarshall project

This component is implemented as an executable (OAMMarshall) and is used as part of the build process. It takes as input interface definitions in the form of XML code. The output are header files to be used by the client and server implementations. Its function is much like an IDL (Interface Definition Language) compiler.

3.6.19. OAMTransport project

This component is implemented as a library (libOAMTransport.so) and provides the transport mechanism used by OAMIPC based clients and servers. It is used by code generated by OAMMarshall. It makes use of OAMOSSL for its socket communication, threading, and synchronization.

3.6.20. OAMEventService project

This is the OAMIPC based event service. It is a server (OAMEventService) which allows events to be broadcast to one or more clients. Event suppliers and consumers are implemented as clients of this server. The supplier and consumer components are part of the OAM subsystem.

4. Architectural Diagrams

This section presents a number of diagrams to put the components described above into the context of a working system. These diagrams may not be complete, but do provide an overview of how the OAM system works.

Figure 1 Overall OAM architecture for DM3 and IPT (PMAC)

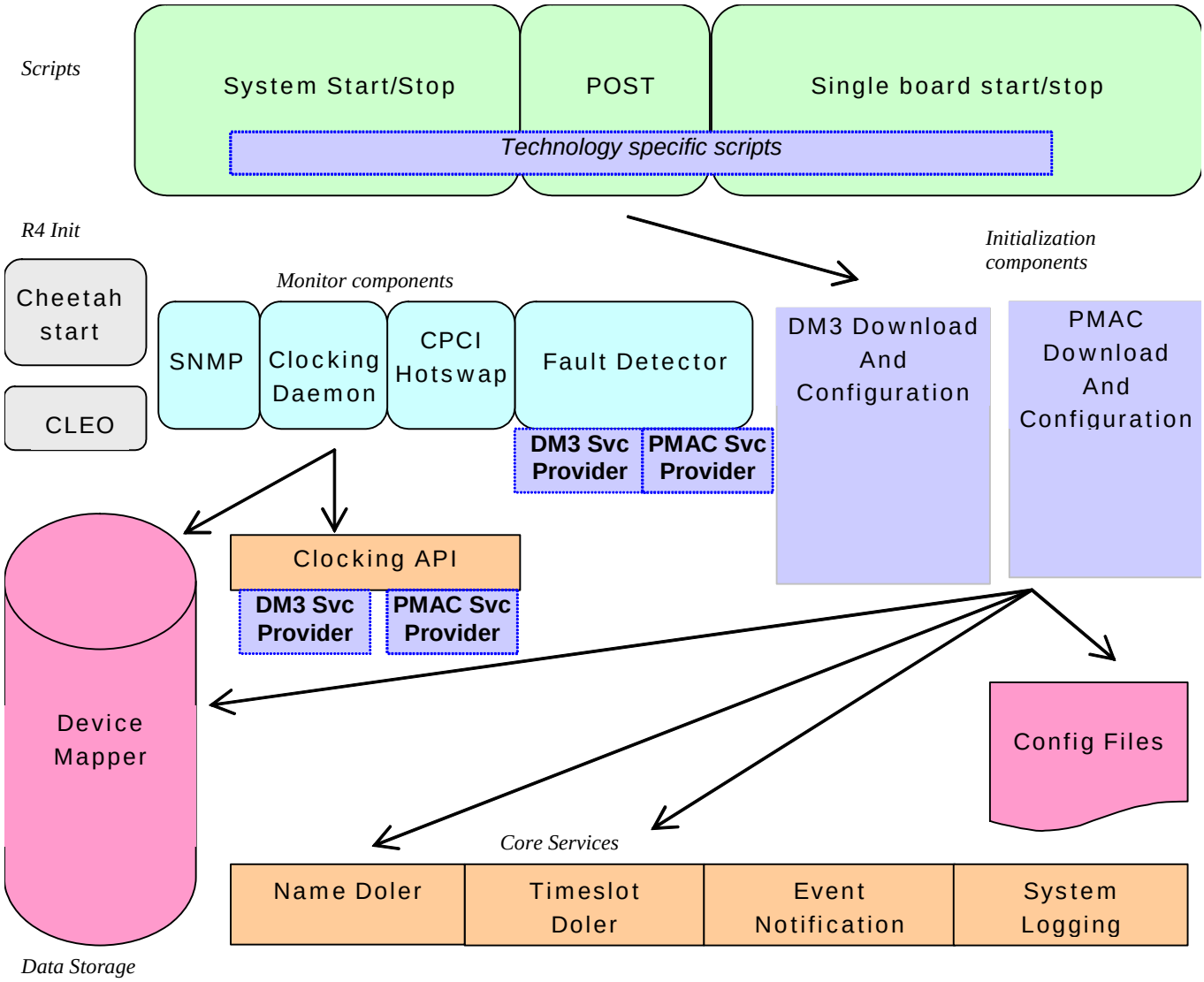

Figure 2 Overall architecture for clocking

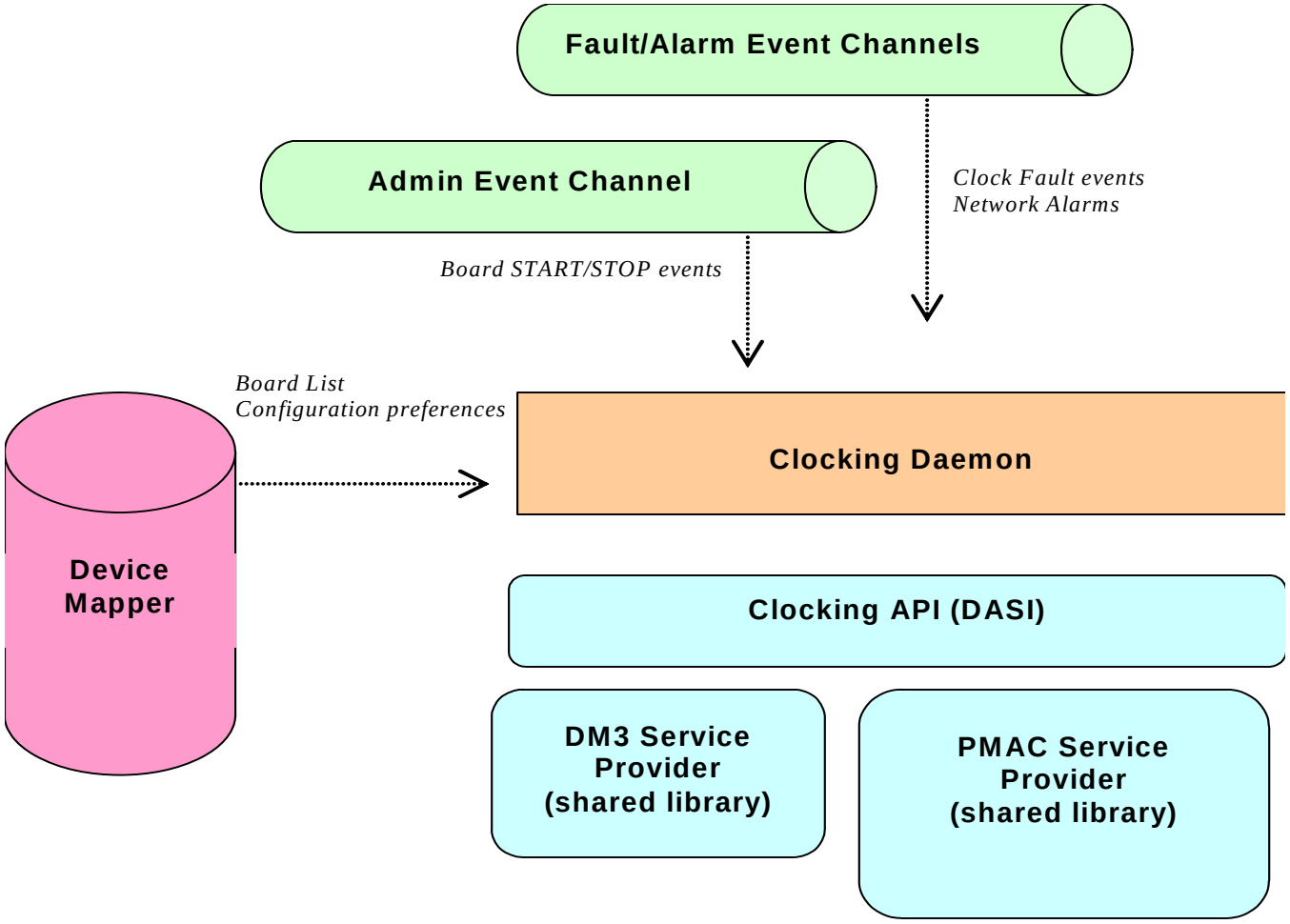

Figure 3 Overall architecture for HotSwap functionality

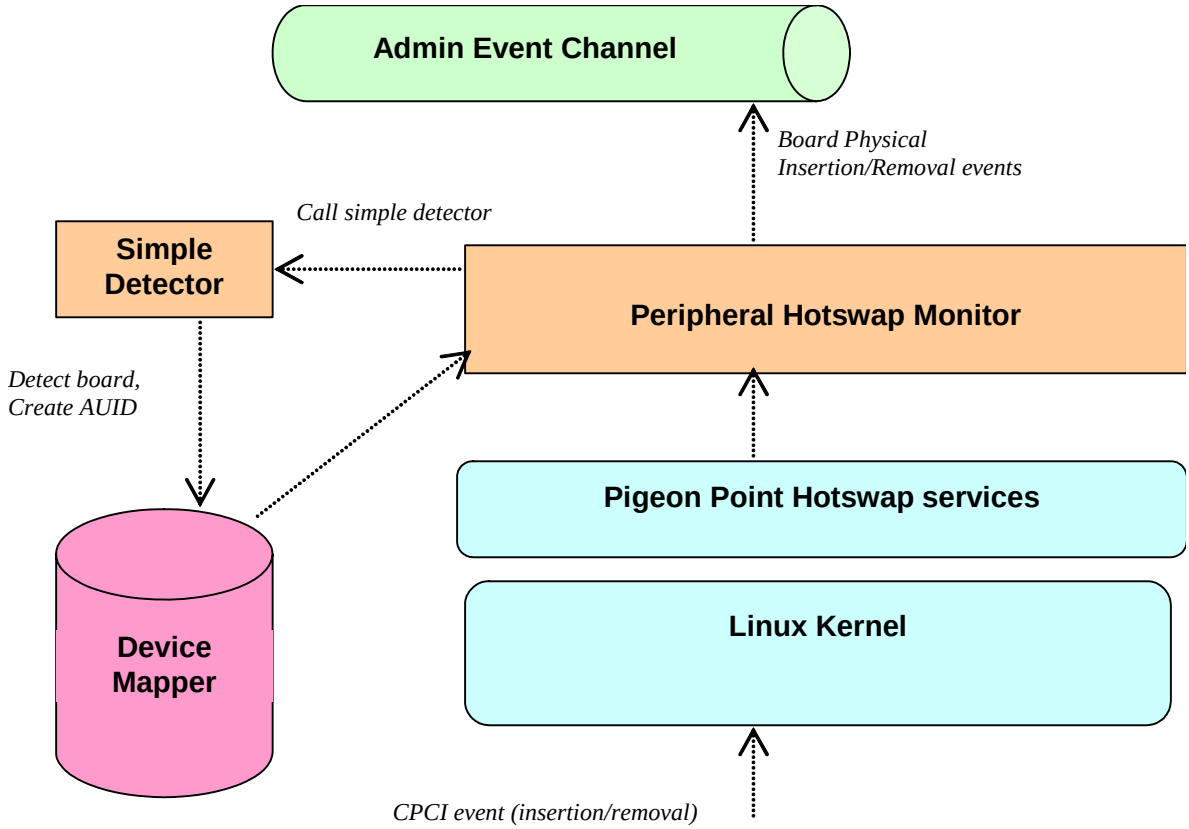

Figure 4 Overview of data stored and managed by Device Mapper

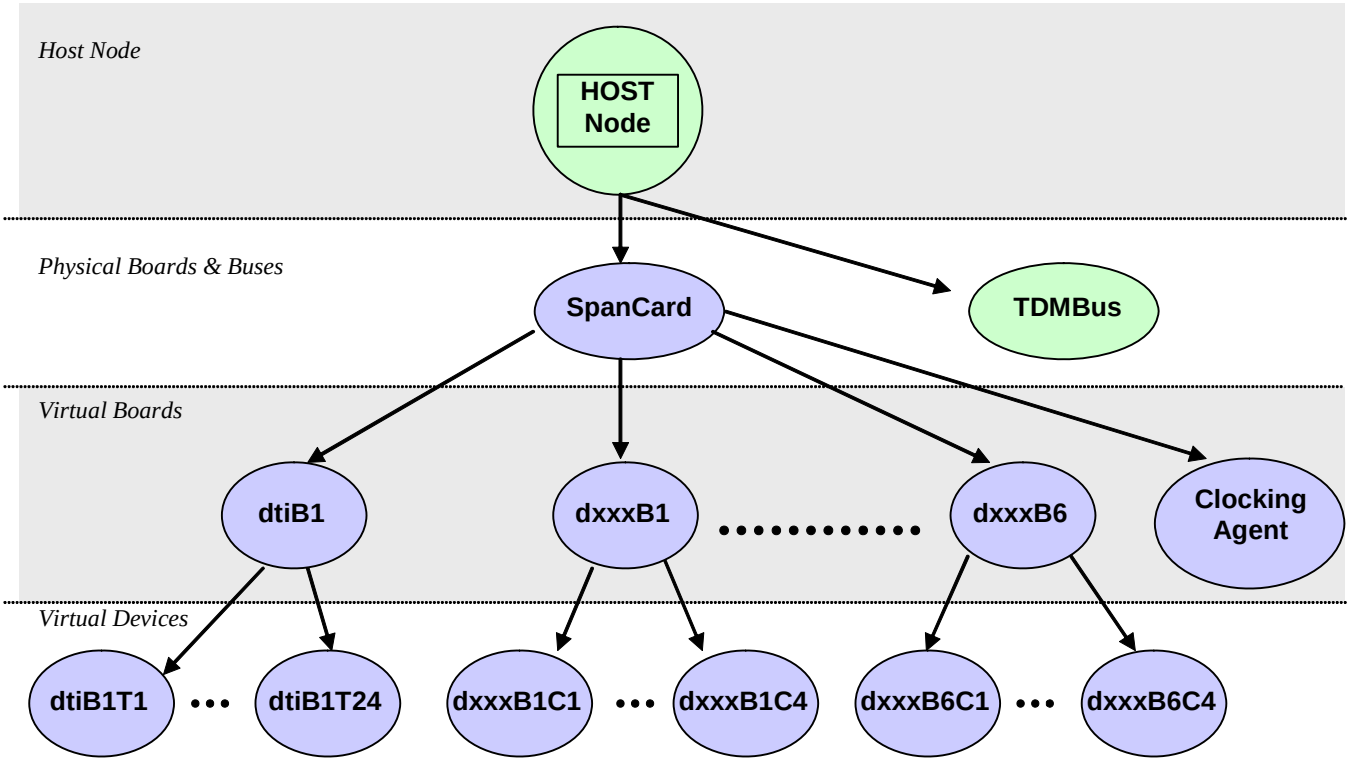

Figure 5 Overall architecture for fault detection and handling

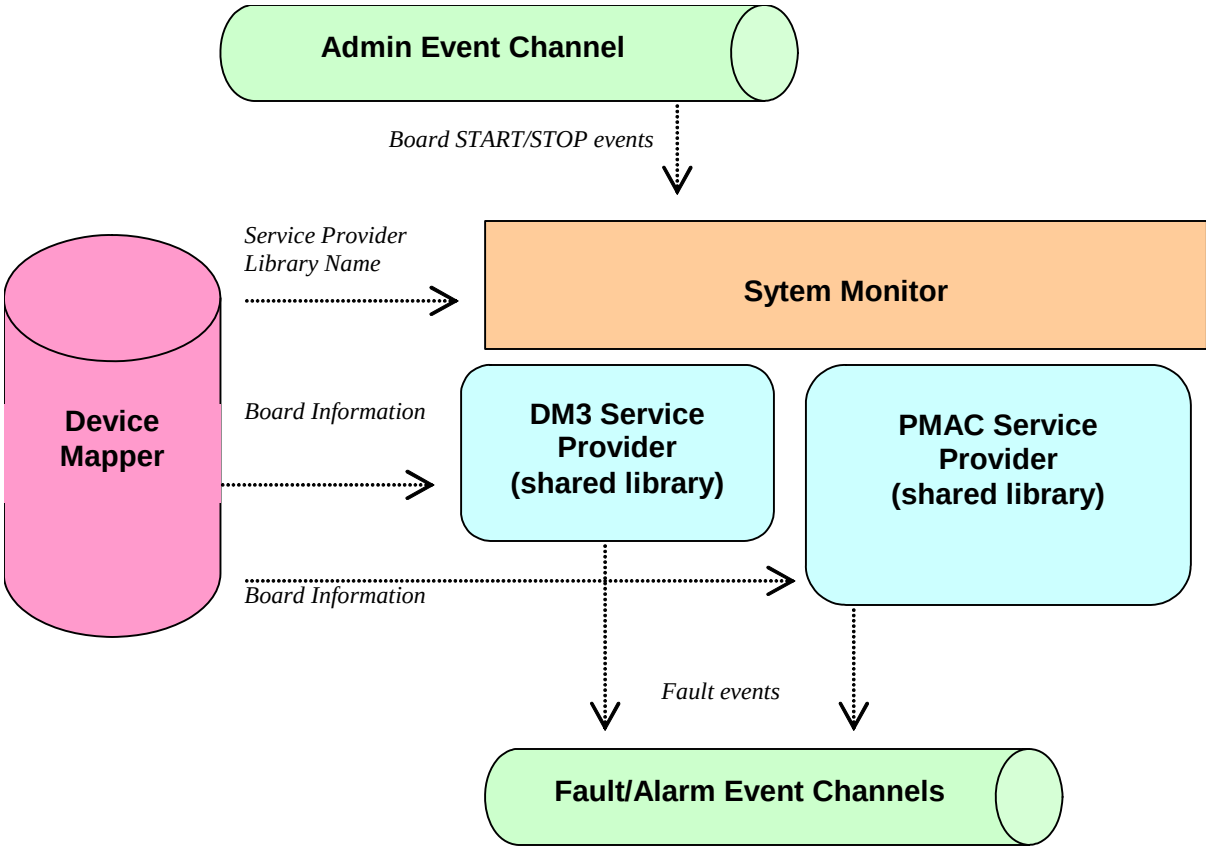

Figure 6 Component interactions between components in OAMIPC architecture

As an example, device mapper uses this IPC mechanism. The device mapper client communicates to the device mapper server through the OAM transport. The initial connection between the client and server is done by looking up information from the naming service. The service is in actuality a configuration file called dlgOamNaming.cfg.

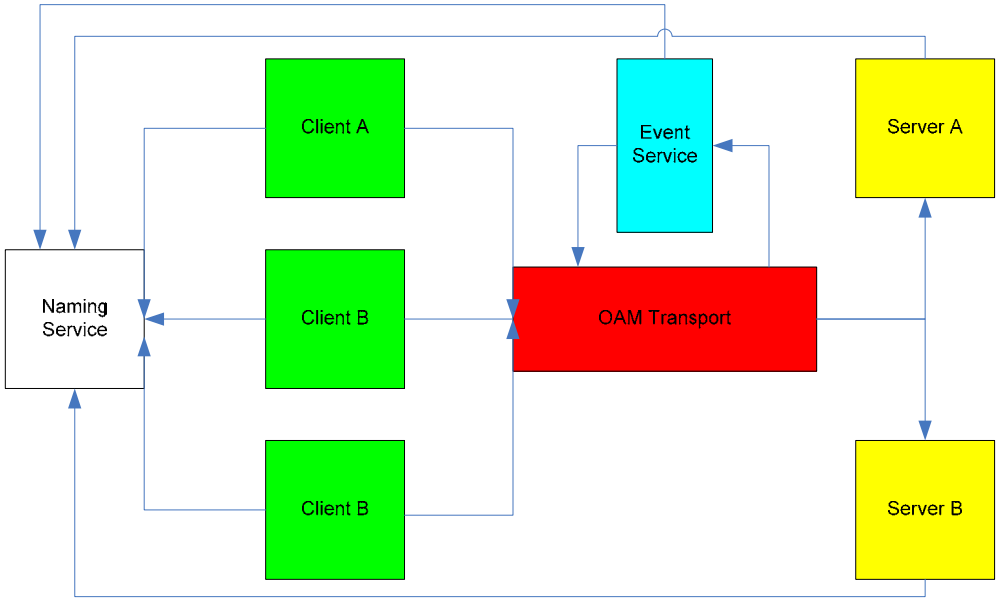

Figure 7 Data flow from OAMIPC clients and servers

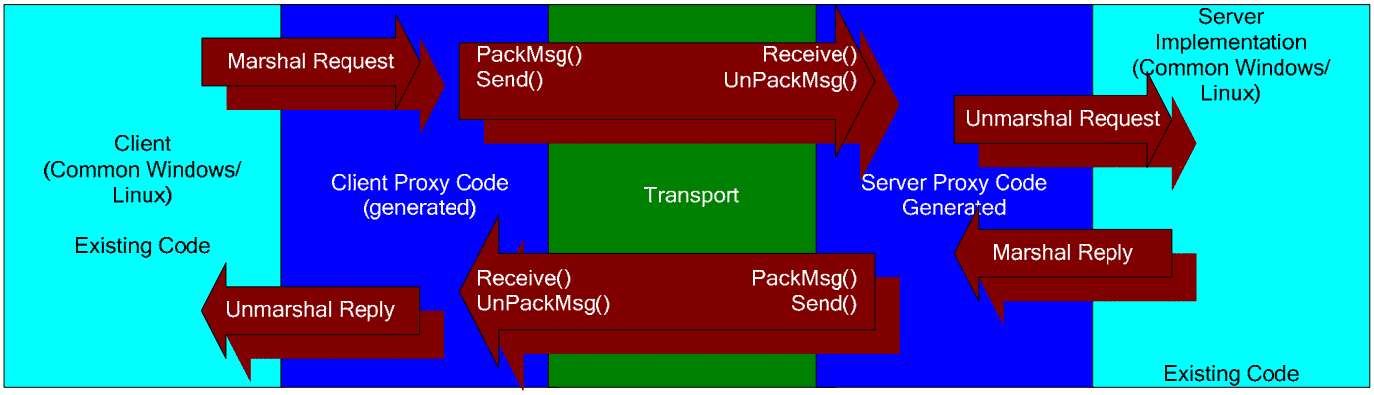

Figure 8 Inputs to Genload and components