QLA Architecture

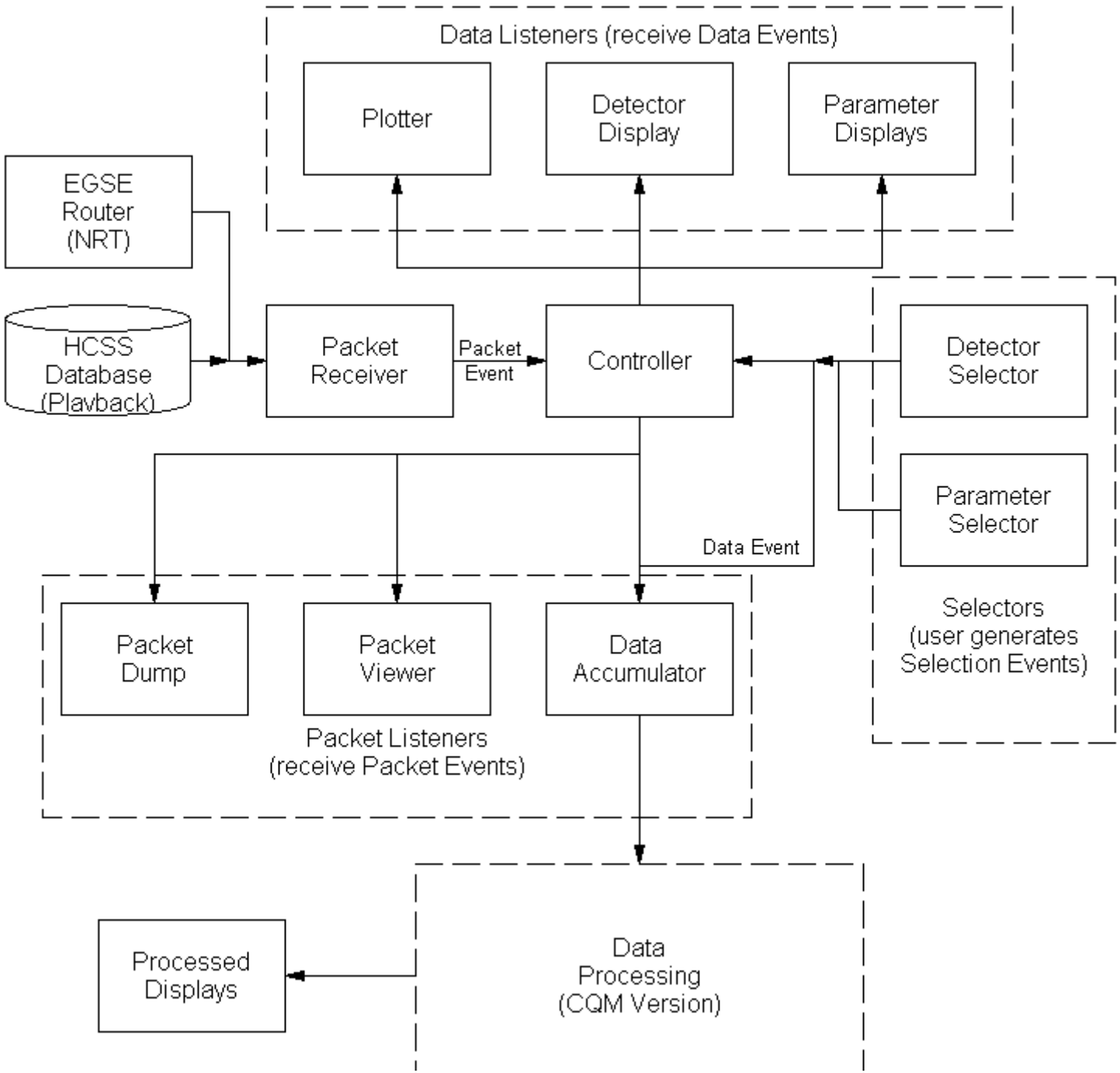
SPIRE-RAL-DOC-002269

Version 2.2

22nd December 2004

This document describes the architecture of the SPIRE Quick Look Analysis System, in order to satisfy the Use Cases UCF-747 and UC-QLA101 as specified in the SPIRE ICC Use-Case Definitions (SPIRE-SAP-DOC-001241)

The SPIRE QLA has a Model-View-Controller (MVC) architecture driven by events and using the Listener pattern to determine where the events are sent. This mechanism is widely used in Java's Swing package. The following diagram shows the flow of events through the system:


Data Listeners (receive Data Events)
Plotter
Detector Display
Parameter Displays
EGSE Router (NRT)
HCSS Database (Playback)
Packet Receiver
Packet Event
Controller
Detector Selector
Parameter Selector
Selectors (user generates Selection Events)
Data Event
Packet Dump
Packet Viewer
Data Accumulator
Packet Listeners (receive Packet Events)
Processed Displays
Data Processing (CQM Version)

PacketReceiver handles the reception of packets, both in NRT from the EGSE router, and in playback mode from the HCSS database (playback can also be performed remotely over the public Internet). This then sends a packet event to the Controller, which distributes this event to all components that have registered themselves as listening for packets. One of these packet listeners is the DataAccumulator which extracts all the parameters that have been selected (by selection events), and assigns a time to each data point (as TAI epoch 1958). These parameters are stored as a time-series. When an updated set of data is available it sends a data event to the Controller, which in turn sends it to all components that have registered themselves as listening for data. The rate that these events are sent is configurable with a default of once per housekeeping packet. It is not sent for every science packet in order to avoid overwhelming displayers with update requests. Note that no data is missed whatever the rate is set to: it is all available.

The Controller can also be asked to provide notification when a set of parameters reach a predetermined set of values. This notification is provided in the way of a callback to a Jython function. A Java API will also be provided if there is a demand for it.

The same data is only stored once, even if multiple listeners are monitoring it. Similarly, data that have the same timeline (i.e. from the same type of packet) share the same time data. Data is *only* stored if there is at least one listener for it.

At the bottom of the diagram can be seen the data processing section. This section will be part of the CQM version, which is still under development. The processing in this section is based on the HCSS IA framework. The data accumulator can be used as the starting point of an IA processing network and contains a method to get a *process* that will create data events as *products*.

Last revised 22nd December 2004 by S.Guest.