GLAST LAT SYSTEM SPECIFICATION	Document # **LAT-????**	Date Effective 03/19/03
	Author(s) Daniel Flath, Alex Schlessinger	Supersedes None
	Subsystem/Office Science Analysis Software	
Document Title **Level 1 Pipeline Implementation**		

Proposal

Level 1 Pipeline Implementation

CHANGE HISTORY LOG

Revision	Effective Date	Description of Changes

CONTENTS

1. PURPOSE

This document proposes a prototype design of the Level 1 Data Processing Facility. The prototype will serve as a toy implementation for processing of raw data from the GLAST Engineering Module (Summer, 2003). Depending on its success in this capacity it will be revised and/or extended to handle data processing for the LAT during flight.

2. DEFINITIONS

DBD – Data Base Driver
Perl libraries implementing implementation-dependant DB access routines. Used by DBI but hidden from developers.
e.g. DBD::Oracle

DPF – Data Processing Facility
Automated tool that archives, catalogs and processes data.
a.k.a. The Pipeline, The Robot.

DPF-FRD – Functional Requirements Document
Pipeline developers' known universe.

DBI – Data Base Interface
Perl library providing abstract (implementation independent) DB access. Developers' DB access point.

GLEAM – GLast Event Analysis Machine
All-purpose simulation tool capable of digitization, monte carlo, reconstruction and upholstery cleaning*.
(* additional fee)

RDBMS – Relational Data Base Management System
e.g. Oracle

3. DESIGN PHILOSOPHY

DPF design is guided by requirements outlined in the DPF FRD (LAT-SSTD-0002000xxx-D41.) A summary of design criteria derived from that document is presented in this section.

3.1 Considerations

- **Proprietary Software** – Although the DPF is slated for implementation at SLAC several 'hot' backups (or mirrors) may be implemented at other institutions. SLAC computing Services (SCS) maintain several proprietary software packages that will likely be used in the SLAC DPF implementation. These include LSF Batch and Oracle RDBMS which are likely unavailable at GSFC or elsewhere due to the high cost of license fees and maintenance.

- **Extensibility** – Due to the nature of prototyping the proposed implementation will certainly evolve. Capabilities will be extended and/or revised as the need arises.

3.2 Design Goals

- **Modularity** – In support of an upgrade path code must be modular – divided into distinct modules – with high-level interfaces. Such forethought of design will greatly reduce the overhead of replacing existing and introducing new components. In particular, introduction of abstraction layers between proprietary software and core DPF routines will allow alternate configurations to be implemented simply by replacing a component wrapper with custom code.

- **Flexibility** – In order to support the anticipated upgrade path the design must be flexible. The implementation of individual DPF components will be motivated by this. The processing database has been implemented to allow the introduction of arbitrary processing tasks, information tags, and data-types. The DPF itself must provide a comprehensive and simple method of supporting new applications and data that will be introduced in the future. Obviously it falls upon the User Interface component to provide this but every wheel and gear behind it must be implemented to support such functionality.

- **Reliability, Ease of Use & The Bottom Line** – Availability of data to collaboration and public scientists depends upon reliable DPF operation. Redundancy is, therefore, critical. At the very least, it must be possible for a DPF mirror to be activated quickly in the event of a failure of the SLAC implementation. Operations such as installing and activating a mirror, restarting or reinstalling the SLAC implementation, and adding custom processing tasks should be straight-forward and require no intimate knowledge of the inner machinery. Automated installation and configuration utilities will facilitate this.

4. IMPLEMENTATION OVERVIEW

4.1 Components in General

There are three varieties of components used in the DPF: 'custom' components proposed for development, 2nd Party components developed and maintained by members of the GLAST software group, and 3rd Party components (both public domain and proprietary) provided by independent vendors.

4.2 Custom Components

The custom components written in Perl script by the DPF team will serve as the glue that holds together the most important DPF components – The data, the applications that process it, and the Data Base catalog of all data and processing. Most of the custom code will be written in Perl script which has standard implementations for all operating systems supported by the Ground Software Working Group. The User Interface components will be written in a mix of Perl, Mod-Perl, and HTML.

Custom components are of two classes: Low-Level routines that provide an abstract interface to a second or third party component, and High-Level routines that do all of the important work of cataloging data, and scheduling and monitoring data processing. By clearly separating the code into these two groups it is possible to replace a 3rd Party component or change the interface to a 2nd Party component almost seamlessly. One need only replace the library of Low-Level routines with another implementing the same abstract interface linking the High-Level routines to the new component.

4.3 Second Party Components

These consist of data processing applications provided by the GLAST Software Working Groups. Interface wrappers written for each application will hide details of program operation (inputs and outputs) from the High-Level DPF routines. Configuration scripts will be written when particularly complex routines are required to supply input parameters to applications. GLEAM is one such example and will be discussed in more detail in the next section.

4.4 Third Party Components

These include the particular Data Base implementation used as well as Perl libraries and Apache web server software including Mod Perl. Of these, the only proprietary software is the Data Base. Unfortunately not all Data Bases do things the same way. An implementation in Oracle may necessarily be different from an implementation in PostgresSQL.

5. IMPLEMENTATION DETAIL

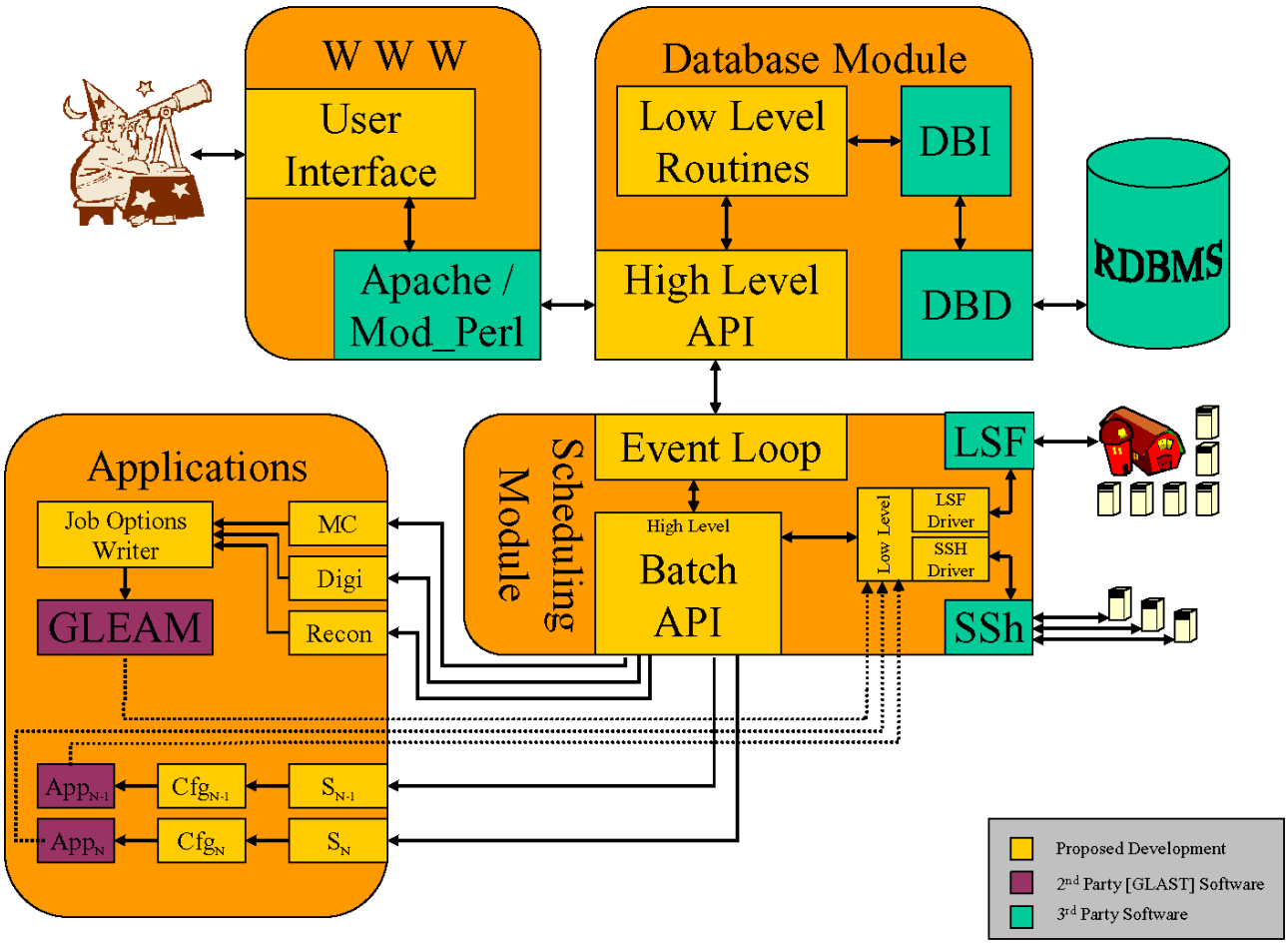

Figure 1 – DPF Modules & Communication Schematic

Notes:
(1) 'App' denotes a data processing application.
(2) 'Cfg' denotes a configuration script used to supply input parameters to a data processing application.
(3) 'S' denotes an application abstraction script used to hide, from the Scheduling Module, the details of application initialization and handling of output dataset(s).

6. DEVELOPMENT SCHEDULE

The following table gives the proposed schedule of development and testing. Major milestones include:

- A stress test of Data Base capabilities [May] will be carried out on the SLAC Oracle server to determine the response time for queries and inserts using mock data of roughly ten times the volume expected for the five-year GLAST mission.
- Linking to the Engineering Module (EM) [June] to test data retrieval, archiving, and processing capabilities developed to that point. The EM will then serve to verify capabilities introduced during further development.
- A document summarizing results of current development and EM testing [August] will be drafted and submitted for informal peer review. Pending the outcome of this review a schedule of further development will be undertaken.
- A final draft of the summary document including all development and test results [October] will be written in anticipation of the November LAT Ground Systems Peer Review.
- Presentation of development and test results during LAT Ground Systems Peer Review. [November]

Month	Dates	Dan	Alex
March	1..31	DB Access Test-Data Entry	Main Loop Skeleton
April	1..14	Testing	
	15..22	DB Access Debugging from test results	LSF Layer
	23..30	DB Access added to main loop	LSF Layer added to main loop
May	1..14	DB Stress Test Job Options File Writer	Application Abstraction Layer MC/Recon Abstraction Scripts
		Integration of JOF Writer / App Scripts	
	15..30	Web Interface to DB	File Transfer Layer [FTP] for EM data
		Integration & Testing using MC/DIGI/Recon	
June	1..	Link to EM [TKR only?] if available	
August	1..	Summary & Writeup of development & results of EM test for review Further development pending review	
October	1..	Summary & Writeup of development & results of EM test for presentation to collaboration	

Table 1 – Prototype DPF Development Schedule