DESIGN SPECIFICATION

THE LHC LOGGING

Abstract
The LHC Logging project is a subproject of the LHC Control Project. The overall functionality of the LHC Logging System, based on the user requirements has been described previously [Ref. 1]. As a next step in the development process, this document describes the architectural design of the LHC Logging System.

Prepared by:
Ronny BILLEN SL/MR
Ronny.Billen@cern.ch
Maciej PERYT LHC/IAS
Maciej.Peryt@cern.ch
Maciek Marczukajtis LHC/IAS
Maciek.Marczukajtis@cern.ch

Checked by:
John POOLE SL/MR
Claude-Henri SICARD LHC/IAS

Approved by:
Robin LAUCKNER SL/DI

With input from:
<table>
<thead>
<tr>
<th>Rev. No.</th>
<th>Date</th>
<th>Pages</th>
<th>Description of Changes</th>
</tr>
</thead>
</table>

Table of Contents

INTRODUCTION.................................................................................................................. 7
1.1 PURPOSE OF THIS DOCUMENT.................................................................................. 7
1.2 SCOPE OF THE SOFTWARE............................................................................................ 7
1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS...................................................... 7
1.4 REFERENCES.................................................................................................................. 8
1.5 OVERVIEW OF THE DOCUMENT................................................................................. 8
1.6 DESIGN METHOD......................................................................................................... 9

2. SYSTEM CONTEXT AND BASIC ARCHITECTURE............................................................. 10
2.1 BACKGROUND ............................................................................................................. 10
2.2 SYSTEM CONTEXT...................................................................................................... 10
2.2.1 INTERFACE TO LHC SYSTEMS................................................................................ 11
2.2.2 INTERFACE TO LHC ALARMS SYSTEM................................................................. 11
2.2.3 INTERFACE TO LHC POST-MORTEM SYSTEM....................................................... 11
2.2.4 INTERFACE TO USERS............................................................................................ 12
2.2.5 HANDLING LONG-TERM STORAGE OF DATA......................................................... 12
2.3 SYSTEM BASIC ARCHITECTURE................................................................................ 12
2.3.1 BASE TOOLS.......................................................................................................... 15
2.3.2 CORE COMPONENTS............................................................................................... 16
2.3.3 APPLICATIONS AND PUBLIC API’S........................................................................ 17

3. COMPONENT DESCRIPTION........................................................................................... 18
3.1 ORACLE 9i DBMS........................................................................................................ 19
3.1.1 TYPE....................................................................................................................... 19
3.1.2 PURPOSE.................................................................................................................. 19
3.1.3 REMARKS................................................................................................................. 19
3.2 ORACLE OLAP............................................................................................................. 20
3.2.1 TYPE....................................................................................................................... 20
3.2.2 PURPOSE.................................................................................................................. 20
3.2.3 FUNCTION............................................................................................................... 20
3.2.4 REMARKS................................................................................................................. 20
3.3 JDBC............................................................................................................................. 21
3.3.1 TYPE....................................................................................................................... 21
3.3.2 PURPOSE.................................................................................................................. 21
3.3.3 INTERNALS............................................................................................................. 21
3.4 JAVA XML PARSER..................................................................................................... 22
3.4.1 TYPE....................................................................................................................... 22
3.4.2 PURPOSE.................................................................................................................. 22
3.4.3 FUNCTION............................................................................................................... 22
3.4.4 DEPENDENCIES..................................................................................................... 22
3.4.5 REMARKS................................................................................................................. 22
3.5 SERVLET CONTAINER............................................................................................... 23
3.5.1 TYPE....................................................................................................................... 23
3.5.2 PURPOSE.................................................................................................................. 23
3.5.3 FUNCTION............................................................................................................... 23
3.5.4 DEPENDENCIES..................................................................................................... 23
3.6 JAVA 2 PLATFORM, STANDARD EDITION .................................................. 24
  3.6.1 TYPE ................................................................................. 24
  3.6.2 PURPOSE ........................................................................... 24
  3.6.3 DEPENDENCIES ................................................................. 24
3.7 CHARTING API ............................................................................. 25
  3.7.1 TYPE ................................................................................. 25
  3.7.2 PURPOSE ........................................................................... 25
  3.7.3 DEPENDENCIES ................................................................. 25
  3.7.4 FUNCTION ........................................................................... 25
  3.7.5 REMARKS ........................................................................... 25
3.8 WEB APP DEVELOPMENT FRAMEWORK ............................................ 26
  3.8.1 TYPE ................................................................................. 26
  3.8.2 PURPOSE ........................................................................... 26
  3.8.3 FUNCTION ........................................................................... 26
  3.8.4 DEPENDENCIES ................................................................. 26
3.9 TIME-SERIES MANIPULATION ......................................................... 27
  3.9.1 TYPE ................................................................................. 27
  3.9.2 PURPOSE ........................................................................... 27
  3.9.3 DEPENDENCIES ................................................................. 27
  3.9.4 FUNCTION ........................................................................... 27
3.10 TIME-SERIES QUERY ...................................................................... 29
  3.10.1 TYPE ................................................................................. 29
  3.10.2 PURPOSE ........................................................................... 29
  3.10.3 DEPENDENCIES ................................................................. 29
  3.10.4 FUNCTION ........................................................................... 29
3.11 METADATA MANIPULATION .............................................................. 30
  3.11.1 TYPE ................................................................................. 30
  3.11.2 PURPOSE ........................................................................... 30
  3.11.3 DEPENDENCIES ................................................................. 30
  3.11.4 FUNCTION ........................................................................... 30
3.12 METADATA QUERY ........................................................................ 31
  3.12.1 TYPE ................................................................................. 31
  3.12.2 PURPOSE ........................................................................... 31
  3.12.3 DEPENDENCIES ................................................................. 31
  3.12.4 FUNCTION ........................................................................... 31
3.13 USER PROFILE DATA MANIPULATION .............................................. 32
  3.13.1 TYPE ................................................................................. 32
  3.13.2 PURPOSE ........................................................................... 32
  3.13.3 DEPENDENCIES ................................................................. 32
  3.13.4 FUNCTION ........................................................................... 32
3.14 USER PROFILE DATA QUERY .............................................................. 33
  3.14.1 PURPOSE ........................................................................... 33
  3.14.2 DEPENDENCIES ................................................................. 33
  3.14.3 FUNCTION ........................................................................... 33
3.15 LHC DATE/TIME API ........................................................................ 34
  3.15.1 TYPE ................................................................................. 34
  3.15.2 PURPOSE ........................................................................... 34
  3.15.3 FUNCTION ........................................................................... 34
  3.15.4 DEPENDENCIES ................................................................... 34
INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT

This document describes the architectural design of the software for the LHC Logging System, taking as applicable the requirements specified in the Functional Specification [Ref 1]. The architecture maps these requirements into specific architectural decisions. The document takes into account the experience learned from previous logging systems, respectively for LEP, LHC String 1 and LHC String 2.

This document provides a complete picture of the desired LHC Logging functionality for the entire development phase, but individual features will be developed incrementally according to their priorities. These priorities will be discussed with the community of users. Functionality may also be subject to changes driven by the acceptance of particular features by the users and by their requests. These features will be reflected in future releases of the document.

This architectural design document (ADD) is to be used by the developers; it contains a high level view of the model of the software, where all coherent functionalities are put into separate software components that work together to provide the needed functionality.

The ADD provides a framework for the software development, which allows independent work on the lower level components for the detailed design and implementation phases.

1.2 SCOPE OF THE SOFTWARE

The name of the software to be produced is LHC Logging.

The software will provide a set of tools, templates and other software components to ensure the functionality of a long-term data logging facility for the LHC, covering any LHC subsystem requiring data logging.

The main objectives of the software are:

- To be able to receive pertinent data from LHC systems and store this data centrally.
- To be able to choose and retrieve data of interest from the central repository, pre-process it if needed and present the data by means of the user interface or transfer the data to external software tools.
- To make easy comparisons between different parameters, and thereby detect a given correlation.
- To have an effective tool for displaying data versus time.

The main motivations of the LHC Logging are:

- To manage information used to improve the LHC operational performance.
- To meet INB related requirements with respect to beam history.
- To make long-term statistics available for CERN management.
- To avoid duplicate logging efforts.

1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Architectural Design Document</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
</tbody>
</table>
1.4 REFERENCES


1.5 OVERVIEW OF THE DOCUMENT

The rest of the document is divided into the following chapters: **System Context and Basic Architecture, Component Description** and Error! Reference source not found..

The chapter **System Context and Basic Architecture** describes the system context in details, outlines the system architecture and discusses major architectural choices.
and possible alternatives. It is an overview aimed at providing a global vision of the design.

The chapter **Component Description** gives a more detailed description of the respective software modules. However, the description is not detailed down to a procedural level.

The chapter Error! Reference source not found. discusses special issues which impact the design or implementation of the software, such as security, reliability, performance, availability and testing issues.

### 1.6 DESIGN METHOD

For the modeling of the software components, the Unified Modeling Language (UML) has been used, as this notational language is becoming more and more popular with training organizations and tool vendors. UML is being developed under the auspices of the Object Management Group (OMG).
2. SYSTEM CONTEXT AND BASIC ARCHITECTURE

2.1 BACKGROUND

The first logging project at CERN based on a unique and central data repository was the **LEP Logging Project** launched in January 1992 [Ref 2]. Without previous experience, the system capabilities and limitations were related to the technology at that time. The LEP Logging system was operational for all LEP physics periods as from 1992 up to 2000 inclusive. The LEP Logging database is still accessible in read-only mode with a large amount of the historical data still on-line.

More recently, logging systems have been implemented for the LHC test beds **String1** and **String2**. The architecture and technology used in the framework of the String2 Logging Project — launched in the fourth quarter of 2000 and fully operational since the end of 2001 — is more in line with the actual state of the art.

Given the positive outcome of the previous logging projects in the accelerator sector, it was decided to combine the experience and lessons learned from these, and design the architecture accordingly.

Consequently, it is not envisaged for the time being to purchase any dedicated application or software package that might provide the required logging functionalities.

2.2 SYSTEM CONTEXT

The system context diagram as shown below, was drawn up in the Functional Specification Document. The LHC Logging System being the parts colored in magenta, data providers on the left hand side and data users on the right hand side. Note that the Logging System internally can be split up in three logical domains: data input, data storage and data output.

To clarify the conceptual entities that interact with the LHC Logging, following diagram can be drawn.
2.2.1 INTERFACE TO LHC SYSTEMS

The interface to the LHC equipment is a standardized data transfer mechanism, whereby the pertinent data for logging is transferred from the LHC equipment front end to the LHC Logging Database.

The formats of the data are XML files that contain time-series and the metadata specific to the LHC equipment concerned.

2.2.2 INTERFACE TO LHC ALARMS SYSTEM

Two possible alternatives are still open, depending on the architectural design of the LHC Alarms System. The architectural difference comes from the choice of an integrated database in the Alarms System – as it is the case for the current CERN Alarms System – or whether the Alarms are "logged" in the LHC Logging System.

2.2.2.1 ALARMS SYSTEM WITHOUT INTEGRATED DATABASE

The interface to the LHC Alarms System is a standardized data transfer mechanism, whereby the LHC Alarms are transferred from the LHC Alarms System to the LHC Logging Database, and stored therein.

The formats of the data are XML files that contain the dynamic data of the Alarms concerned. In this scenario the interface is very similar to the one to LHC equipment.

2.2.2.2 ALARMS SYSTEM WITH INTEGRATED DATABASE

The interface to the LHC Alarms System is a standardized data transfer mechanism, whereby the LHC Alarms are transferred from the LHC Alarms System to the LHC Logging data extraction tool.

The formats of the data are XML or plain text files that contain the appropriate data of the Alarms concerned.

2.2.3 INTERFACE TO LHC POST-MORTEM SYSTEM

The interface to the LHC Post-Mortem System is a standardized data transfer mechanism, whereby chosen sets of Post-Mortem data are transferred from the LHC Post-Mortem System to the LHC Logging data extraction tool.

The formats of the data are XML or plain text files that contain the appropriate data of the Post-Mortem concerned.

Eventually the data transfer system should also work in the opposite direction, whereby logging data is pulled into the analysis tools of the Post-Mortem system.
2.2.4 INTERFACE TO USERS

The interface to the User provides the tools to extract the logging data and manage user preferences, if applicable. This functionality comes in three flavors:

- The fundamental way to extract data is through a lightweight web application running in a web browser and generated on the server-side.
- The programmatic way is through an API that provides a read-only access to the Logging System from the third-party Java programs.
- The third way is through a fully-fledged GUI application that is installed and runs on the client workstation. This is the tool that allows correlating logging data with the information belonging to the Alarms and Post-Mortem systems.

2.2.5 HANDLING LONG-TERM STORAGE OF DATA

It is intended to keep the data available as long as possible in the logging system. Possibly archiving by means of off-line storage media may be required. However, it should be possible to retrieve archived data with the data extraction tool of the LHC Logging System. It is possible to achieve this by table partitioning and moving old partitions to off-line storage. This way no additional interface is required, only administration utilities (to keep the inventory of data).

2.3 SYSTEM BASIC ARCHITECTURE

The LHC Logging System is composed of a number of software components deployed on distributed hardware nodes connected through the standard CERN Intranet. The components, nodes, dependencies among the components and the connections among nodes are shown in the LHC Logging System deployment diagram (Figure 1 on page 14). This kind of diagram shows software components, which can be roughly understood as running processes, without getting into their inner structure. It also shows logical hardware nodes, but it does not exclude that several distinct logical nodes map into a single physical machine.

The core of the system is the **LHC Logging Data Server**. This is the node that runs an instance of Oracle 9i DBMS with the logging schema and low-level data manipulation PL/SQL software (used when plain SQL does not suffice).

The **LHC Logging Application Server** provides the intermediate layer used by the components that need more than pure data manipulation and query functionality. The components running at this node are implemented as Java servlets. As such they need a standard **Servlet Container** to run. At present **Tomcat** ([http://jakarta.apache.org/tomcat](http://jakarta.apache.org/tomcat)) seems to be the ideal candidate to take up that role. The **Data Loading Servlet** is a component that receives packets of data as XML files sent over HTTP, parses them, loads them to Oracle and returns the status information (loading success or failure, reasons for failure) to the component that initiated the transmission of data. The **Data Extraction and Manipulation Servlet** is a web application that gives the user basic data extraction and metadata manipulation functionality. The HTML/JavaScript is generated at the application server, while the client only needs a supported web browser to use the application.

The role of the **Data Extraction and Manipulation Station with Thin Client** is to run a web browser that will provide a visualization and user interaction facilities for running the web application (thin client).

The **Data Extraction and Manipulation Station with Fat Client** is the node that runs the fully-fledged **LHC Logging GUI Application**. This application, which we expect to be written in Java for sake of portability, provides all the special facilities necessary or desirable for efficient data visualization and analysis (efficient navigation, interactive charting, comprehensive data export, management of user profiles).
Last but not least is the **Data Producer Station**. This is the node where the data is converted to the agreed-upon XML format (by the **Native-to-XML Data File Translator**) and submitted over HTTP by the **Logging Data Transmitter** to the Data Loading Servlet.
Figure 1. LHC Logging: deployment diagram
All these software components are based on a number of building blocks, or packages. The packages are not necessarily executable modules, but also libraries, drivers, database schemas, etc. Whenever possible, LHC Logging System features will be provided using off the shelf packages “glued” by custom built solutions.

The package diagram (Figure 2 on page 15) shows the main packages into which the LHC Logging System has been subdivided. The packages have been grouped in logical layers. Packages are allowed to use services provided by other packages on the lower layers and on the same layer, but not on the higher layers. The layers are ordered by the level of generalization, the most general-purpose layer being the lowest.

Figure 2. LHC Logging System packages

A brief description of the layers and the packages is provided hereafter, while the next chapter will contain a detailed description of the features included in the packages.

2.3.1 BASE TOOLS

The bottom layer contains packages that are essentially off-the-shelf components. The Logging System just provides packaging and distribution support for these components when it is needed for run-time installations. These packages provide the low-level general-purpose infrastructure for the development of specialized services related to logging.
• **Oracle 9i DBMS**  
  Oracle Database Management System, version 9i. Provides data management infrastructure for the persistent data.

• **Oracle OLAP**  
  OLAP: On Line Analytical Processing. Oracle OLAP provides a complete range of analytic functionality including multidimensional and predictive functions and support for rapid query response times such as those that are normally associated with specialized analytical databases. In addition, starting with Oracle 9i it provides the functionality for retrieving and manipulating timestamped data with sub-second granularity.

• **JDBC**  
  JDBC technology is an API that lets you access virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL databases.

• **Java XML Parser**  
  Allows to validate and extract information from an XML document from within a Java program.

• **Servlet Container**  
  A Java application running Java servlets and implementing a particular version of the servlet API. A servlet container may run standalone, also serving static web pages and maybe other forms of dynamic content; it may run as a "plug-in" for a web server.

• **Java™ 2 Platform, Standard Edition**  
  Also known as J2SE™. Provides the essential compiler, tools, runtimes, and APIs for writing, deploying, and running programs in the Java programming language.

• **Charting API**  
  Real-time Java charting API. Generates dynamic charts. Can be deployed in Java applications, Java applets, JavaServer Pages (JSP), servlets, and through HTML.

• **Web App Development Framework**  
  Framework for building Web applications. Set of classes providing the essential underpinnings for building web applications (form data validation and management, dynamic content generation...).

### 2.3.2 CORE COMPONENTS

The second layer provides the packages that are essential for the development of any logging service or application. Included in this layer are the packages that make it possible to create and manipulate database objects, to handle timestamp manipulations, to record and handle abnormal conditions.

• **Time-Series Manipulation**  
  Creates and manipulates the time-series (timestamped data). Creates database objects necessary for storing timestamped data. Provides methods for inserting, updating and deleting timestamped data. PL/SQL package.

• **Time-Series Query**  

• **Metadata Manipulation**  
• **Metadata Query**  

• **User Profile Data Manipulation**  
Creates and manipulates the user profile data (i.e. optional per-user information: authentication, preferences, routine queries). Creates database objects necessary for storing user profile data. Provides methods for inserting, updating and deleting user profile data. PL/SQL package.

• **User Profile Data Query**  
Methods for the retrieval of user profile data (i.e. optional per-user information: authentication, preferences, routine queries). PL/SQL package.

• **LHC Date/Time API**  
An API responsible for ensuring a coherent UTC time-stamping of the logging data, independent of the local time, and for providing the possibility to visualize logged data either in UTC or in local time.

• **Administrative Tools**  
Tools for gathering information about the behavior of the Logging System and history of user interactions with the system. PL/SQL package.

### 2.3.3 APPLICATIONS AND PUBLIC API’S

The third and last layer provides high-level API’s, tools and applications. These are the packages that make the Logging System really useful and usable. They provide standard data loading and extraction mechanisms as well as data access API’s for users that require programmatic interface to the Logging System.

• **XML Data File Loader**  
Parses an XML file with a well-defined structure containing the logging data and inserts data to the logging system using the Time-Series Manipulation and Metadata Manipulation packages. Returns the status (success or failure, reasons for failure) to the caller. Java servlet.

• **HTTP File Upload**  
Submits a file over HTTP to a specified URL using the multipart/form-data encoding.

• **Data Access API**  
Provides programmatic read-only access to the Logging System from the third-party Java programs. Java API.

• **LHC Logging GUI Application**  
Graphical user interface to extract, visualize and analyze the logging data. Installed and running on the client workstation.

• **Data Extraction and Manipulation Servlet**  
Lightweight data extraction and visualization client running in a web browser and generated on the server-side.
3. COMPONENT DESCRIPTION

This chapter describes each package of the system in more detail. For clarity, the description of each package starts after a page break and contains the following parts if appropriate:

- **Type**
  Says what kind of package it is: device driver, executable, library, etc.

- **Purpose**
  Traces the component to the software requirements.

- **Dependencies**
  Describes the preconditions for using this component.

- **Function**
  Says what the component does. More detailed discussion of the component. Prose, diagrams, pseudocode...

- **Remarks**
  Any comment or observation considered relevant to this component.
3.1 ORACLE 9i DBMS

3.1.1 TYPE

The Oracle 9i DBMS is a Database Management System, a commercial software package from Oracle Corporation.

3.1.2 PURPOSE

The purpose of the Oracle DBMS is to store persistently the logging data, i.e. time-series data, its metadata as well as user profile data. Additionally, programming logic to access and to manipulate the data is stored in the Oracle DBMS as well.

3.1.3 REMARKS

Due to the extended functionality with respect to the present production version at CERN —Oracle 8i—, the version 9i of the Oracle DBMS is a prerequisite for the initial implementation. Throughout the lifetime of the Logging System, the DBMS might be upgraded to a higher version for technological reasons.

Following base technologies are integral parts of the Oracle DBMS:

- SQL (Structured Query Language) which is used to create, store, modify, retrieve, and manage information in an Oracle database.
- PL/SQL Oracle’s procedural language extension to SQL.
3.2 ORACLE OLAP

3.2.1 TYPE

The Oracle OLAP is a set of Oracle SQL functions and PL/SQL packages.

3.2.2 PURPOSE

The Oracle OLAP (Online Analytical Processing) provides the elementary tools for analysis of the data stored in the database. OLAP tools enable users to analyse different dimensions of multidimensional data. For example, it provides time series and trend analysis views.

3.2.3 FUNCTION

As from the Oracle version 9i the functionality that used to be provided by Oracle Time Series is included directly in the SQL analytic functions.

Some of the functionality of the analytic functions of interest to this project includes:

- Scaling-down/scaling-up of time series data
- Interpolation or repeating closest preceding value for scaling-down
- Averages for scaling-up
- Moving averages
- Generating derived time series

3.2.4 REMARKS

According to the Oracle White Papers on the topic [Ref 3, Ref 4], it is believed that the performance of the Oracle OLAP is adequate for our requirements, but this needs to be checked and confirmed.
3.3 JDBC

3.3.1 TYPE

JDBC (Java Database Connectivity) is a standard Java API.

3.3.2 PURPOSE

JDBC enables Java programs to execute SQL statements. This allows Java programs to interact with Oracle or any SQL-compliant database. JDBC is similar to ODBC (Open Database Connectivity), but is designed specifically for platform-independent Java programs, whereas ODBC is language-independent.

3.3.3 INTERNALS

The JDBC 3.0 API is comprised of two packages:
1. The java.sql package
2. The javax.sql package, which adds server-side capabilities
3.4 JAVA XML PARSER

3.4.1 TYPE

The XML parser is a java library, which includes standard XML APIs: SAX (Simple API for XML) and DOM (Document Object Model) API.

3.4.2 PURPOSE

The parser enables easy and standardized interpretation of XML documents from java language and is used by the XML Data File Loader to parse the logging data files.

3.4.3 FUNCTION

The parser will be used for effective parsing of large XML documents. Thus mainly the event-based parser is considered - SAX 2.0. The SAX parser is not memory consuming and it is highly recommended when large documents, with fixed structure, are parsed and during parsing no updates of XML data are performed.

The SAX parser generates events; e.g. start-tag or end-tag is reported to the application through callbacks. The XML Data File Loader, which uses the parser, should implement handlers to take accurate actions for different events.

3.4.4 DEPENDENCIES

This library depends on:

- J2SE (Java™ 2 Platform, Standard Edition)

3.4.5 REMARKS

Most of the currently available XML parsers (e.g. Oracle parser (Oracle XML Developer's Kit for Java), Xerces (Apache XML Project)) support W3C recommendations as well as Sun's JAXP (Java API for XML Processing).

It is important to mention that new Sun's Java 2 SDK v1.4 added support for XML processing, but leaving the possibility to use any vendor parser implementation.
3.5 SERVLET CONTAINER

3.5.1 TYPE
Servlet containers are usually written in Java. A servlet container may be an independent part of the Web server (e.g. Apache + Tomcat) or included in a Web server (e.g. BEA Weblogic Server).

3.5.2 PURPOSE
The servlet container provides the necessary infrastructure to run the following servlets:
- Data Extraction and Manipulation Servlet
- Data Loading Servlet

3.5.3 FUNCTION
Unlike a Java client program, a servlet has no static main() method. Therefore, a servlet must execute under the control of an external container.
Servlet containers are designed to run servlets as well as JSPs (Java Server Pages) in Web applications.

3.5.4 DEPENDENCIES
The container depends on:
- J2SE (Java™ 2 Platform, Standard Edition)
3.6 JAVA 2 PLATFORM, STANDARD EDITION

3.6.1 TYPE

The J2SE is a development environment for building applications, applets, and components using the Java programming language.

3.6.2 PURPOSE

J2SE is required as a runtime environment for all Java programs, which will be developed in the framework of the project. The use of the J2SE is essential because the following packages require some of the tools (e.g. a compiler) and libraries, which are not present in the runtime environment (J2RE\(^1\)):

- Servlet Container
- Web App Development Framework

3.6.3 DEPENDENCIES

- None

---

\(^1\) The Java\(^{TM}\) 2 Runtime Environment is the minimum standard Java computing environment for running applications written in the Java programming language. It contains the Java virtual machine, Java core classes, and supporting files.
3.7 CHARTING API

3.7.1 TYPE
This package is a Java class library.

3.7.2 PURPOSE
This library is a real-time Java charting API. It is used by the Data Extraction and Manipulation Servlet for visualising the logging data.

3.7.3 DEPENDENCIES
The package depends on:

- J2SE (Java™ 2 Platform, Standard Edition)

3.7.4 FUNCTION
The package Charting API is used to dynamically generate charts. It must provide the support for time series data. The charts may be deployed in Java applications, Java applets, JavaServer Pages (JSP), servlets, and through HTML.

3.7.5 REMARKS
Several commercial products exist today that provide this package (e.g. Oracle Chart Builder, Sitraka [formerly KL Group] JClass ServerChart). The actual choice will depend on the performance, which must be adequate for online applications.
3.8 WEB APP DEVELOPMENT FRAMEWORK

3.8.1 TYPE
This is a framework (i.e. a set of custom Java libraries), based on standard technologies like Java Servlets, JavaBeans and XML.

3.8.2 PURPOSE
The main reason to introduce this kind of framework is to ease and to add more control to the development and management of Web application. We expect to use this framework to develop Data Extraction and Manipulation Servlet.

3.8.3 FUNCTION
Web App Development Framework is based on Model-View-Controller (MVC) design paradigm. MVC is an architecture, which separates the three portions of the paradigm:
- View represents the part of the framework, which changes most frequently – the presentation layer. It consists of three parts:
  - The general presentation structure, which is usually handled using JSPs.
  - Graphics and binary data generation – handled by servlets.
  - Support for data structure changes – when data exists in XML, the multiple views of these data can be provided by XSLT (eXtensible Stylesheet Language Transformations) usage.
- Model encapsulates the business objects and API for application’s functionality.
- Controller section is the glue between model and view. It allows execution of the business logic in the model as a response to user requests. It also decouples data presentation from business data and logic.

Very important functionalities provided by the framework and required for HTML Form handling for this project are:
- Dynamic HTML Form generation,
- Verification of entered data,
- Programmatic retrieval of data entered by the user from the forms.

3.8.4 DEPENDENCIES
This framework is dependent on:
- J2SE (Java™ 2 Platform, Standard Edition)
  - Java XML Parser
- Servlet Container
3.9 TIME-SERIES MANIPULATION

3.9.1 TYPE

This package is a PL/SQL package.

3.9.2 PURPOSE

This package is responsible for the creation and manipulation of time-series data. Time-series are the essence of logging data. To put it simply, time-series is an ordered collection of timestamp-value pairs. More detailed discussion of timestamps and values comes later in this chapter.

3.9.3 DEPENDENCIES

This package depends on the following packages:

- Oracle 9i DBMS
- Oracle OLAP
- Administrative Tools

3.9.4 FUNCTION

The package **Time-Series Manipulation** is responsible for creating, inserting to, deleting from and updating the database objects implementing the time-series functionality.

Time-series is an ordered collection of timestamp-value pairs. The following class diagram (Figure 3 on page 27) describes the types of objects constituting the time-series.

![Time-series class diagram](image)

**Figure 3. Time-series class diagram**

The package provides the following functionality:
- Create and drop database objects for the storage of time-series.
- Insert values to an instance of time-series. For higher efficiency the package supports bulk insertions. It is possible to configure the behavior with respect to violation of the unique timestamp constraint: if the clash occurs, depending on the mode, either the insertion is rejected, or the old value is overridden with the new one.
- Update values in an instance of time-series.
- Delete values from an instance of time-series that match the specific timestamp criteria (timestamp equal to the given timestamp, most recent timestamp preceding the given timestamp, having timestamps within the given time bounds).
3.10 TIME-SERIES QUERY

3.10.1 TYPE
This package is a PL/SQL package.

3.10.2 PURPOSE
This package is responsible for the retrieval of time-series data. Time-series data model is described in Section 3.7 on page 25.

3.10.3 DEPENDENCIES
This package depends on the following packages:
- Oracle 9i DBMS
- Oracle OLAP
- LHC Date/Time API
- Administrative Tools

3.10.4 FUNCTION
The package **Time-Series Query** provides a set of methods to retrieve the time-series and information on the time-series. Methods from **Time-Series Query** are usually used in the combination with **Metadata Query** package (time-series data is pretty much useless by itself).

The package provides the following functionality:
- Get a subset of time-series intersecting a time range.
- From an instance of time-series intersecting a time range, derive time-series scaled up to a given frequency. For example time-series with irregular timestamps having values on average every 25 seconds are scaled up to 1-minute regular intervals; the values returned are 1-minute averages.
- From an instance of time-series intersecting a time range, derive time-series scaled down to a given frequency using interpolation or repeating the last actual value. For example we have original time-series with a dead band set so that when value does not move a sample is recorded every 10 minutes but during the rapid changes it is recorded as often as every second. We want to scale it up to the frequency of 1 Hz so that we get equally spaced values but we do not lose any information.
- Get count of values (or timestamps) in an instance of time-series intersecting a time range. This query executes much faster than the ones that return actual data.
3.11 METADATA MANIPULATION

3.11.1 TYPE

This package is a PL/SQL package.

3.11.2 PURPOSE

Creates and manipulates the metadata (data that describes time-series). Creates database objects necessary for storing metadata. Provides methods for inserting, updating and deleting metadata.

3.11.3 DEPENDENCIES

This package depends on the following packages:
- Oracle 9i DBMS
- Time-Series Manipulation
- Time-Series Query
- LHC Date/Time API
- Administrative Tools

3.11.4 FUNCTION

The package Metadata Manipulation provides a layer on top of the time-series data model giving a meaning to the timestamped data. Time series is a simple object with a unique identifier that keeps a reference to an ordered sequence of points (see Figure 3 on page 27). In order to make this data meaningful, we have to be able to:
- Associate time-series with a sensor or other entity representing a value that changes over time (let’s call it variable).
- Define a grouping or a hierarchy of variables.
- When we store raw data (as opposed to data expressed in engineering units) we need to store the sufficient amount of information (conversion information) to be able to convert it to on the fly to the engineering units.

Metadata must have a period of validity, or the start and end date between which it is valid for a specific time-series or set of time-series. This feature caters for the fact that the systems being logged are dynamic entities and they change over time.

To summarise, the package provides the following functionality:
- Create and drop database objects for the storage of metadata.
- Create, delete and update variables and their attributes.
- Associate variables with time series.
- Group variables into sets; create hierarchy of sets.
- Create, delete and update conversion definitions (classes of conversions).
- Create, delete and update conversion parameters for individual variables with individual periods of validity.
3.12 METADATA QUERY

3.12.1 TYPE
This package is a PL/SQL package.

3.12.2 PURPOSE
This package handles the retrieval of metadata.

3.12.3 DEPENDENCIES
This package depends on the following packages:
- Oracle 9i DBMS
- Time-Series Manipulation
- Time-Series Query
- LHC Date/Time API
- Administrative Tools

3.12.4 FUNCTION
The package Metadata Query provides a set of methods to retrieve the metadata. It also provides methods to use metadata in combination with time-series data to get the derived time-series.

The package provides the following functionality:
- Get information about a variable.
- Get variable’s children.
- Get variable’s parent.
- Get the identifier of the time-series associated with a variable.
- Get variable’s conversion data as of specified date and time.
- Apply a conversion to a time-series and get the derived time-series.
3.13 USER PROFILE DATA MANIPULATION

3.13.1 TYPE

This package is a PL/SQL package.

3.13.2 PURPOSE

Creates and manipulates the user profile data (per-user information: authentication, preferences, routine queries). Creates database objects necessary for storing user profile data. Provides methods for inserting, updating and deleting user profile data.

3.13.3 DEPENDENCIES

This package depends on the following packages:

- Oracle 9i DBMS
- Meta Data Query
- Time-Series Query
- Administrative Tools

3.13.4 FUNCTION

The package provides the following functionality:

- Create and drop database objects required for storing user profile data.
- Store and manipulate information about the user: id, name, password.
- Store and manipulate information about the user’s preferences: chart styles, data file formats for extracted data.
- Store and manipulate information about the routine queries and their attributes: variable lists, complete data requests (e.g. for daily monitoring).
3.14 USER PROFILE DATA QUERY

This package is a PL/SQL package.

3.14.1 PURPOSE

This package handles the retrieval of user profile data.

3.14.2 DEPENDENCIES

This package depends on the following packages:
- Oracle 9i DBMS
- Meta Data Query
- Time Series Query
- Administrative Tools

3.14.3 FUNCTION

The package provides the following functionality:
- Given a user identification, retrieve the information about the user.
- Authenticate the user.
- Retrieve the information about the user’s preferences: chart styles, data file formats for extracted data.
- Retrieve the information about the routine queries and their attributes: variable lists, complete data requests. Allows executing the stored queries or quickly setting the parameters for the new queries on the basis of stored ones.
3.15 LHC DATE/TIME API

3.15.1 TYPE

The LHC Date/Time API is a PL/SQL package with a middle-layer Java wrapping.

3.15.2 PURPOSE

The purpose of this API is two-fold:
1. To ensure a coherent UTC time-stamping of the logging data, independent of the local time.
2. To provide the possibility to visualize logged data either in UTC or in local time.

3.15.3 FUNCTION

This API converts a UTC timestamp (i.e. date-time group with optionally fractional seconds) from its native format at the data source (i.e. well-defined number or character string format) to the value of the appropriate internal timestamp data type in the logging system.

Inversely, this API converts the internal timestamp to a value usable by the applications and public APIs.

3.15.4 DEPENDENCIES

This package depends on:
- Oracle 9i DBMS
- JDBC
- J2SE

3.15.5 INTERNALS

The Java API will wrap PL/SQL code in order to access the Oracle data types from within the Java environment. A native Oracle datatype will be used for the timestamp data. The most likely candidate is the Oracle9i TIMESTAMP datatype, which is an extension of the DATE datatype. The TIMESTAMP datatype can cope with fractions of a second down to the nanosecond.

The Oracle functions TO_TIMESTAMP() and TO_CHAR() are the basic functions to convert timestamp data.

For the conversion from UTC time to local time, a dedicate piece of PL/SQL code will be developed.
3.16 ADMINISTRATIVE TOOLS

3.16.1 TYPE
This package is a PL/SQL package.

3.16.2 PURPOSE
Tools that gather information about the behaviour of the Logging System and history of user interactions with the system. These tools will allow the Logging System Administrator to:

• Improve the data loaders.
• Detect anomalies in the transfer of logging data to the system (unusually high data rates, holes in data) and alert the administrators of data sources.
• Monitor user interactions with the system to improve the user interfaces and performance of queries.

3.16.3 DEPENDENCIES

• Oracle 9i DBMS
• Oracle OLAP

3.16.4 FUNCTION
The package provides the following functionality:

• Record data loading sessions and their attributes: what was logged, by whom, how much, etc.
• Record data extraction sessions: who, what queries.
• Record aggregates statistics about logged data: volume, value counts.
• Provide analysis tools for all recorded information.
3.17 XML DATA FILE LOADER

3.17.1 TYPE
This package is a Java servlet.

3.17.2 PURPOSE
This servlet receives XML files containing the logging data over HTTP encoded as multipart/form-data (as specified in RFC 1867 [Ref 5]), parses them and loads their content to the Logging System database.

3.17.3 DEPENDENCIES
This package depends on the following packages:
- Oracle 9i DBMS
- JDBC
- Java XML Parser
- Servlet Container
- J2SE
- Time-Series Manipulation
- Metadata Manipulation
- LHC Date/Time API

3.17.4 FUNCTION
The following activity diagram models a typical data file loader session.
Figure 4. Data file loader session activities.
3.18 HTTP FILE UPLOAD

3.18.1 TYPE

This package is an executable (script).

3.18.2 PURPOSE

Http File Upload is required for wrapping-up a file to be uploaded into a specified URL. The URL can be e.g. the address of XML Data File Loader.

3.18.3 FUNCTION

It encapsulates the XML data in order to prepare a multipart/form-data (as specified in RFC 1867 [Ref 5]) http post request, which is then submitted to the specified URL. Http File Upload also receives and outputs the confirmation and error messages of the invoked components.

3.18.4 DEPENDENCIES

None
3.19 DATA ACCESS API

3.19.1 TYPE
This package is a Java class library.

3.19.2 PURPOSE
This library provides programmatic read-only access to the Logging System from the third-party Java programs.

3.19.3 DEPENDENCIES
This package depends on the following packages:
- Oracle 9i DBMS
- JDBC
- J2SE (Java™ 2 Platform, Standard Edition)
- Time-Series Query
- Metadata Query
- User Profile Data Query

3.19.4 FUNCTION
The package Data Access API is a Java wrapper for the PL/SQL packages providing the read-only access to the logging database. It provides the sum of their functionalities and makes them available as Java classes and methods callable from any Java code.
3.20 DATA EXTRACTION AND MANIPULATION SERVLET

3.20.1 TYPE

This package is a Java servlet.

3.20.2 PURPOSE

This servlet is a lightweight data extraction and visualization client running in a web browser and generated on the server-side.

3.20.3 DEPENDENCIES

This package depends on the following packages:

- Oracle 9i DBMS
- JDBC
- Servlet Container
- J2SE
- Time-Series Query
- Metadata Query
- User Profile Data Query
- User Profile Data Manipulation
- LHC Date/Time API
- Charting API

3.20.4 FUNCTION

The package **Data Extraction and Manipulation Servlet** is a server-side Web application providing a browser-based graphical user interface to the logging system. User interactions are translated into HTTP requests that trigger the generation of dynamic content that provide the following functionality:

- Navigation through component hierarchies.
- Form-based extraction of metadata.
- Form-based extraction of time-series data.
- Browsing of time-series data.
- Managing user profile data.
- Dynamic chart generation.

3.20.5 REMARKS

The **Data Extraction and Manipulation Servlet** will be the first tool that will be provided to the end-users and will therefore have the highest priority during the implementation phase of the data extraction methods.
3.21 LHC LOGGING GUI APPLICATION

3.21.1 TYPE
This is a client-side Java Application.

3.21.2 PURPOSE
This Graphical User interface should permit the end-user to extract, visualize and analyse the logging data. The application is installed and running on the client workstation.

3.21.3 DEPENDENCIES
This package depends on the following packages:
- Oracle 9i DBMS
- JDBC
- J2SE
- Time-Series Query
- Metadata Query
- User Profile Data Query
- User Profile Data Manipulation
- LHC Date/Time API
- Charting API

3.21.4 FUNCTION
The LHC Logging GUI Application is a client-side Java application that provides a graphical user interface to the logging system package. The functionality overlaps and goes beyond of what the Data Extraction and Manipulation Servlet provides. This tool will allow correlating logging data with the information belonging to the Alarms and Post-Mortem systems.

3.21.5 REMARKS
The detailed functionality is not explained here, as it could be a sub-project on its own. Moreover, the dedicated features catering for special requirements will surely evolve over time as soon as more logging data providers sign up.

For the Graphical User Interface of the LEP Logging System, a complete set of documentation was produced [Ref 6] from user requirements up to detailed design. The implementation resulted in the GUILS product and its Java-GUILS successor.
4. OTHER ISSUES

4.1 SECURITY
The logging data must be preserved and secured during the lifetime of the LHC and beyond. No direct access to the data will be given to any user. The read-only usage of the stored logging data will be publicly available to end-users within the CERN domain by means of the provided tools.

4.2 PERFORMANCE
For the automated data transfer mechanisms to the database, data storage rates must be higher than what the data provider sends.
Data transfer rates to the end-users must be perceived as adequate and acceptable.

4.3 AVAILABILITY AND RELIABILITY
The logging system is a chain of components from data provider to data storage and further to the end-user. The network is clearly the key component in the communication channels, and we are counting on the network as a continuous service. Nevertheless it must be anticipated that the delivery of the data to the storage fails. Therefore a local buffering of the data to be logged on the data provider side must be foreseen.
For the DBMS, we are counting as usual on the natural reliability and the support of CERN's IT/DB group. We are committed to following IT/DB's recommendations and guidelines with respect to server platform, database software tools, and backup strategy.
A dedicated Service Level Agreement (SLA) must be determined with the required services from CERN's IT Division.

4.4 SCALABILITY
It is clear that the data volumes and data rates will be much higher during the LHC beam commissioning than during the QRL acceptance tests and surely with respect to the logging systems that we dealt with up to now. Therefore the issue of scalability is of high importance.
As far as the DBMS is concerned, it is our understanding that efforts in the IT/DB group are ongoing, compliant to Oracle's strategic direction of Real Application Clusters (RAC). Inherent to this concept is the ease of scalability by simply “plugging in” the required processing and disk space resources.

4.5 TESTING ISSUES
According to the current project planning, the logging system should be available for the QRL by mid May 2003 and accepted by December 2003. The idea is to capture available data as soon as possible and feed it through the different components of the Logging System. In the case of a lack of available data from the QRL systems, it is envisaged to use String2 data.
Dedicated testing procedures should cover all issues mentioned above: security, performance, availability and scalability. These procedures will simulate situations of high data input and output rates in order to explore the limitations; simulations of typical failure modes to verify data recovery.
Also, the quality of the data needs to be preserved throughout the logging system and this needs to be proven through tests on data samples.