**eTMF Electronic Loader**

**UPD - CAT 5[A] Custom Software (Bespoke)**

**SLC - Software Design**

**Specification**

|  |  |
| --- | --- |
| Document Version: |  |
| Document Date: | 04-OCT-2017 |
| Document Author: |  |

TABLE OF CONTENTS

1. Introduction 4

2. System Overview 4

2.1 Significant Decisions and Recommended Approach 4

3. Design Considerations 5

3.1 Assumptions 5

Constraints 5

3.2 System Environment 5

3.3 Policies and Tactics 5

3.3.1 Choice of Which Specific Product to Use 5

3.3.2 Standards or Conventions to Use 6

3.3.3 Authentication and Authorization 6

3.3.4 Choice of Particular Algorithm or Design Pattern 6

3.3.5 Hierarchical Organization of the Source Code 7

3.3.6 Source Control Methodology 7

3.3.7 Software Build and Deployment Methodology 7

3.3.8 Plans for Requirements Traceability, Unit Testing, and Maintenance 8

4. High Level Architecture 9

5. High Level Design 10

5.1 Requirements 10

5.2 Subsystem, Components, and Interfaces 10

5.2.1 Subsystems 11

5.2.2 EEL3 Structures Design 11

5.2.3 EEL Engine Design 11

5.3 Data Model 12

5.3.1 Study Profile 13

5.3.2 DIA Template 14

6. Design 15

6.1 EEL Engine 15

6.1.1 Web Service 15

6.1.1.1. Project Build Request 16

6.1.1.2. Project Creation Job Execution 17

6.1.1.3. Folder Request 18

6.1.2 Principle Classes 18

6.1.3 REST Service Structure 20

6.2 EEL3 Structures 22

6.2.1 User Interface 22

6.2.2 Code Packages and Classes 26

7. Definitions, Acronyms and Abbreviations 28

8. References 28

9. Document Revision History 28

Appendix A Jira User Stories (Requirements) 29

Appendix B EEL3 Database Schema 30

Authorization 31

# Introduction

The eTMF Electronic Loader (EEL) Structures system is a collection of components that provide user interfaces, web services, and scheduled batch jobs that automate the management of clinical trial documentation within the eTMF repository. The system implements several key functions for automation of:

1. Central File Folder structure creation implementing the DIA reference model
2. folder structure creation can be executed at study start-up or on demand (just-in-time),
3. Central File Folder structure updates and modifications,
4. study close activities,
5. periodic verification and update of Central File Folder structures based on specified site status changes, and
6. transfer of study data to the cloud TMF service (Wingspan).

# System Overview

The system will be comprised of 4 major components:

1. a pluggable core engine (accessed through web services) for creating complete central files structures or specific file structure branches on demand (EEL Engine)
2. a web-based interface for managing study structure profiles and creating/maintaining structure templates (EEL3 Structures)
3. interface to the ELVIS content repository
4. relational database storage for all system information

## Significant Decisions and Recommended Approach

This release of EEL Structures will be a complete re-implementation of the current EEL Structures application. It will provide improved usability for the ELVIS support team and reduce the time required for study set-up.

| # | Approach | Benefits | Drawbacks |
| --- | --- | --- | --- |
| 1 | The user interface will be implemented with the Vaadin framework version 7 using a MVP (model-view-presenter) pattern. | Provides a modern user interface | Introduces a new framework |
| 2 | Application logging will be based on the LOGBack Project. | Compliance with the SLF4J API. |  |
| 3 | Scheduled job management will utilize Quartz Scheduler. | Solution is already used by other Quintiles projects. |  |
| 4 | System will be hosted on the Apache Tomcat 7 platform in a clustered configuration using F5 for traffic management. | Allows flexible hardware configuration to scale capacity and load increases. |  |

# Design Considerations

## Assumptions

|  |  |
| --- | --- |
| # | Description |
| 3.1.1. | The Content Server 10 security structure will be used for authorization |
| 3.1.2. | 4 different schemas will be used:   * ELVIS master index schema (CS10) * EEL3 Structures schema * EEL Structures schema (legacy structure profiles) * Elvis Reference (customer, project, site data) |
| 3.1.3. | Authentication will be handled by a single sign-on using Quintiles Active Directory (Windows AD). The system will provide single-sign-on using Waffle Authentication framework. |

## Constraints

|  |  |
| --- | --- |
| # | Description |
| 3.2.1. | All users must be Quintiles staff with access to the system via Active Directory authentication. |
| 3.2.2. | The EEL 3 Structures system will provide a web-based interface and display all text in English (including labels, messages, button text, and data previously entered by the users). |
| 3.2.3 | The user interface will operate with Microsoft Internet Explorer Version 11 or Chrome web browsers. |
| 3.2.4 | The system will operate on any OS certified with Tomcat Version 7. |

## System Environment

|  |  |
| --- | --- |
| # | Description |
| 3.3.1. | Database is Oracle version 12c |
| 3.3.2. | Content Server 10.5 (ELVIS and Infosario ECM clusters) |
| 3.3.3 | Java Version 1.7.x |
| 3.3.4 | Server: Windows Server 2012R2 |
| 3.3.5 | Client: Windows XP, Windows 7 (browser Internet Explorer 11, Chrome) |
| 3.3.6 | Cloud Service Provider: Wingspan TMF 2.4, 2.5 |

## Policies and Tactics

### Choice of Which Specific Product to Use

The following tools have been selected for project development:

* Application server: Tomcat 7
* User interface: Vaadin 7.x
* Dependency framework: Spring 3.2
* Job scheduling: Quartz Scheduler 2.1.7
* Logging: LOGBack 1.0.9
* Single sign-on: Waffle 1.4
* Data access layer:
  + Oracle JDBC implementation
  + Livelink API (LAPI) access to content server

### Standards or Conventions to Use

The Sun, Inc. (Oracle) conventions for Java coding have been adopted for EEL 3. A reference copy is available in Elvis at: <http://elvis.quintiles.com/livelink/cs.exe/Open/101347188> .

### Authentication and Authorization

Authentication will be handled by a single sign-on using Quintiles Active Directory and the Waffle integrated web authentication framework. The system will authenticate users against AD via Waffle (using either negotiate, NTLM, or Kerberos) and pass the credentials (user principle) through to the application.

Authorization to access to EEL3 Structures functions (user interface pages) will be based on Content Server security groups. User accounts that are known to Content Server and are members of a defined security group will be allowed access to the application.

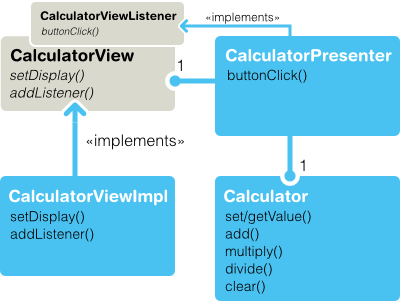
### Choice of Particular Algorithm or Design Pattern

The complete system will be multi-tier with the separate user interface, business model, and data access layers. Key system components will be designed so that they can be deployed to a variety of targets: user application, job scheduler, and web services.

Overall system design will follow a lightweight approach. System components will be implemented as “plain old Java objects” (POJO) instead of relying on technologies like EJB. The Spring framework will manage dependency among system components.

Application data will reside in an Oracle database. Access will be through Oracle JDBC calls. All SQL code will be encapsulated in stored procedures that are built in a single package. This will isolate the data layer from the application code making it possible to update or tune data access without needing to deploy a new version of the application.

Interaction between the user interface and the domain or data access layers will follow the Model-View-Presenter (MVP) pattern. MVP provides for a complete separation between the domain model and view; all interaction is managed by a presenter.



### Hierarchical Organization of the Source Code

Source code will be organized in packages with ***quintiles.com*** as the base. Data and core business objects will be segregated from UI and web service code to allow packaging business functionality it JARs that can be deployed independently of the complete application.

### Source Control Methodology

All project source code will be stored in the Quintiles Subversion source code repository. Code check-in will be accompanied by descriptive, explanatory comments for the current change including the current development sprint.

Code will be managed against the **trunk** with significant project milestones tracked as tags or branches as appropriate.

### Software Build and Deployment Methodology

The integrated development environment (IDE) will be Eclipse (version Indigo or Juno). Development builds can be executed and debugged directly from Eclipse using a referenced Tomcat installation on the development workstation.

Builds to the staging, test, and production servers will be based on an SVN version and executed on Bamboo. A Bamboo build job (or jobs) will extract the source code from SVN, compile all projects components, and then assemble a deployable WAR file. The WAR artifact from a successful build will be transferred to the EEL deployment server.

### Plans for Requirements Traceability, Unit Testing, and Maintenance

Project development will follow the Quintiles Q-Agile methodology. User requirements will be captured as user stories (issues) in Jira and tracked against specific development and validation testing tasks.

# High Level Architecture

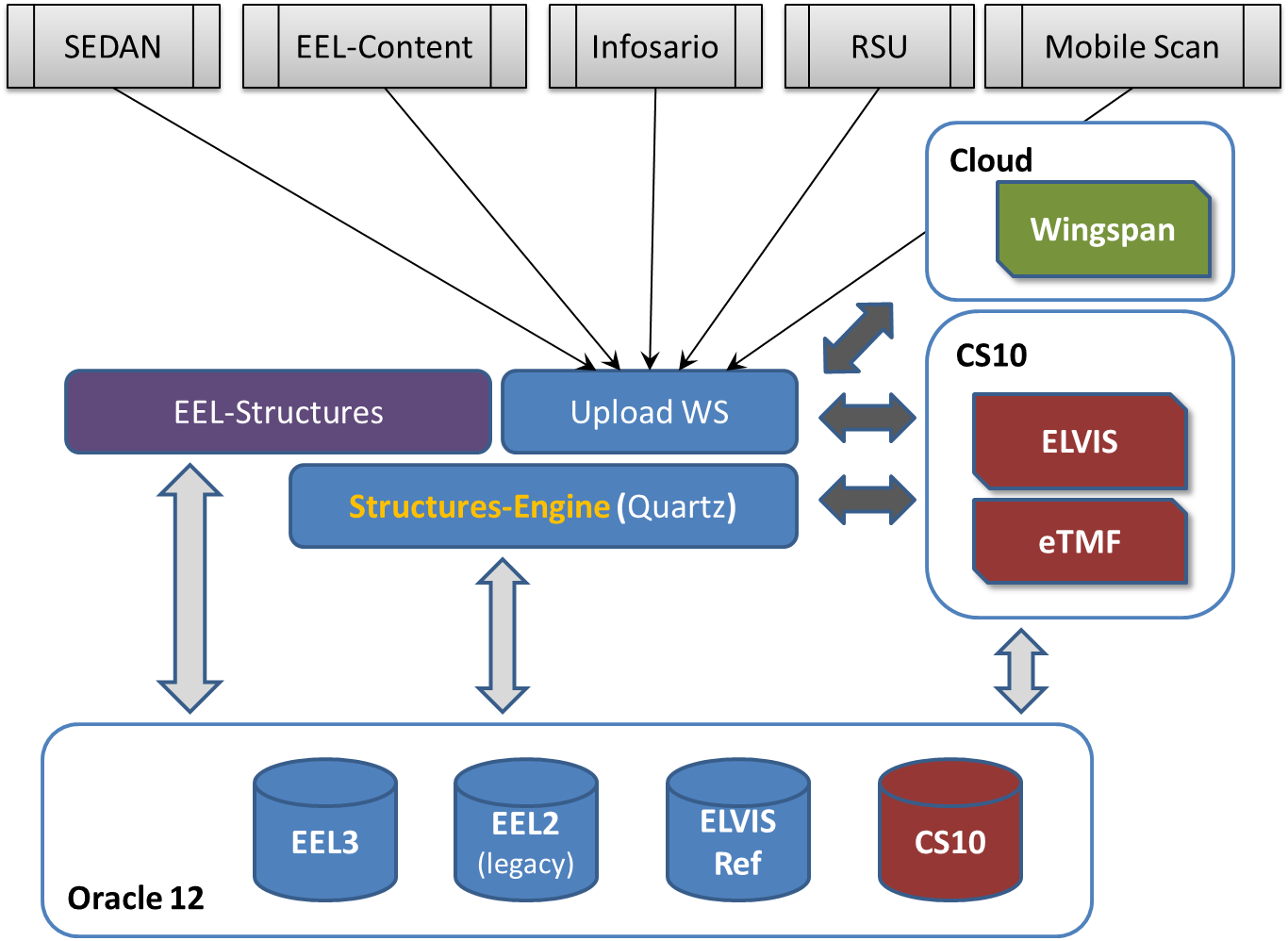
EEL Structures is one component of the Quintiles eTMF product. The following diagram illustrates how E-Structures fits within the complete solution.



# High Level Design

EEL Structures is a part of the overall ELVIS Electronic Loader (EEL) solution. Structures facilitates the automated building of TMF file structures in Content Server that define document organization, metadata, and security constraints.

The following diagram shows EEL Structures relationship to related systems:



## Requirements

This project will be developed following the Q-Agile methodology. All requirements will be tracked in JIRA stories that will be linked to Top Team and related test cases. The requirements and functional mapping will not be detailed in this document.

## Subsystem, Components, and Interfaces

The components that will be developed as part of this project are EEL-Structures, the Structures-Engine, and updates to the Upload Web Service.

### Subsystems

|  |  |
| --- | --- |
| Subsystem | Responsibilities |
| EEL3 Structures | Web browser based user interface |
| EEL Engine | Build structures in content server and implement just-in-time folder requests |

### EEL3 Structures Design

Composition

The EEL3 Structures application is a web application that allows users to create and manage study profiles. Creating or updating a profile will trigger the EEL Engine to build a new or update an existing project structure in Content Server.

The application will also provide a status view for monitoring running and completed jobs.

The web application will be developed using the Vaadin UI framework to manage the HTML interface. The Spring framework will be used to implement dependency injection in the application.

Resources

The application uses the EEL3 and ELVIS\_REF database schemas.

Interface / Exports

The EEL3 application is dependent on the Oracle database and the EEL3 engine.

### EEL Engine Design

Composition

The EEL3 Engine is a web application that implements REST web services and a job scheduling mechanism. It provides the following basic services:

* + - * build a study structure based on specified profile, or
      * provide the reference object ID for a DIA artifact folder.

The REST services are accessed by the EEL3 Structures application, the ELVIS upload web service, and the DIA navigation tool within eTMF (Content Server 10).

Uses / Interactions

The EEL engine implements 2 basic functions: build a minimal study structure and provide reference IDs for artifact folders (where TMF documents will be stored). The engine will schedule a job using the Quartz scheduler and then process those requests when triggered. The jobs can build core projects or country/site sub-structures.

The folder retrieval feature allows folders to be built just-in-time (JIT) when required for document storage. It will also provide a mechanism to locate existing artifact folders.

Resources

The application uses the EEL3 and ELVIS\_REF database schemas. The application has a number of DAO classes that manage the relational database mapping and data persistence.

Interface / Exports

The upload service has three principle interfaces. The first is LAPI access to Content Server. The second is access to the EEL3 database using DAO objects from the EEL3 Structures library. The third is access to the Wingspan cloud service through the Wingspan REST API.

## Data Model

Release 4.0 of EEL Structures will have a separate database design from previous versions that will be managed in its own schema. The database is the source for EEL3 (DIA) study profiles that have been created and the reference templates for TMF content structures.

The complete schema reference is contained in Appendix B (embedded PDF).

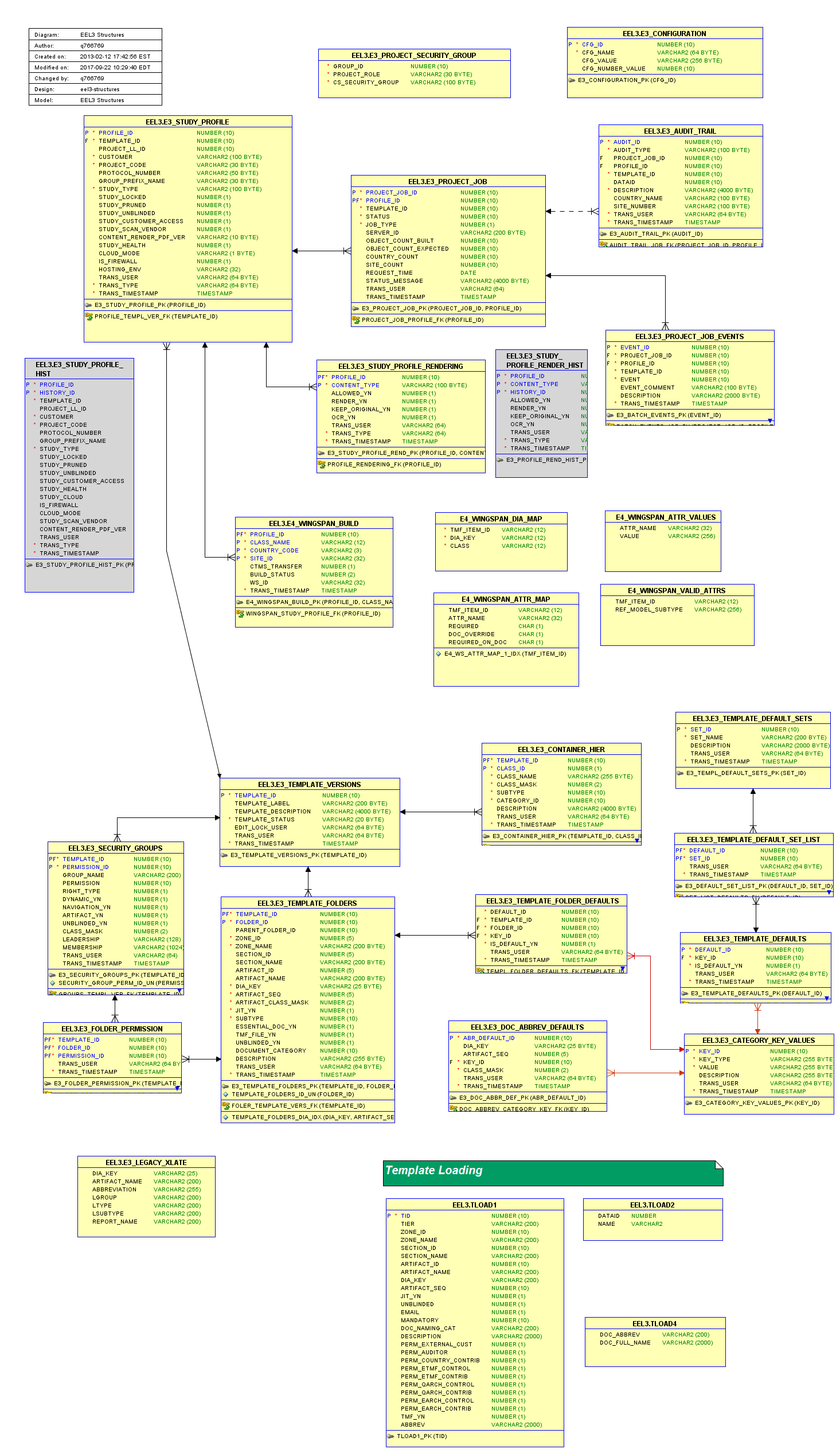
The data model has two primary sections: the study profile and DIA template.

### Study Profile

The study profile contains information about all studies that have been built in EEL3. In addition to sponsor and protocol information, the study type, template used to build the study, and electronic document support is detailed. This area also has a record of build jobs for each study with supporting audit trails.

The study profile also manages studies hosted in the cloud service and tracks key status and build information.

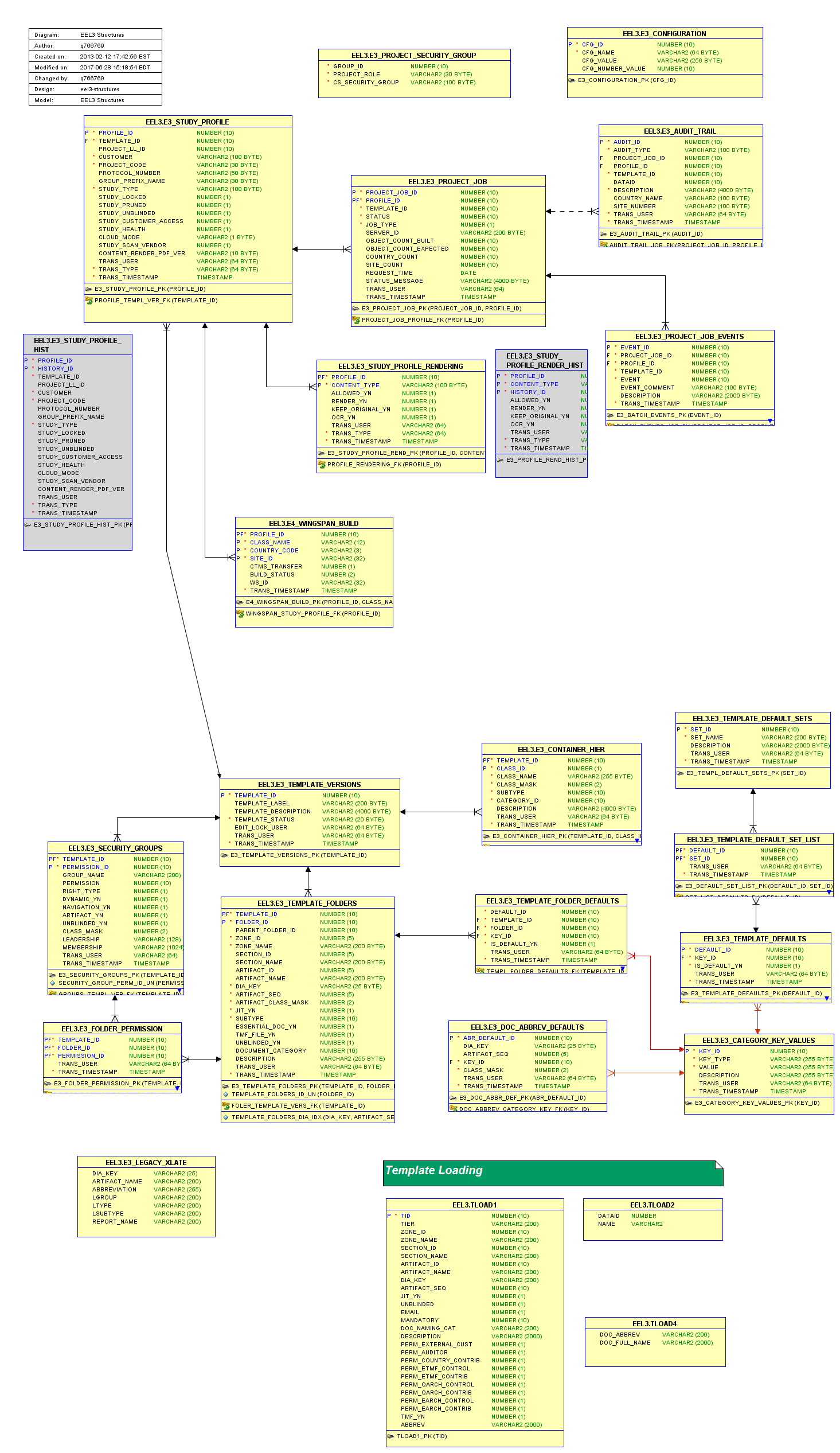
EEL3 Schema -- Study Profile



### DIA Template

The template area has details about the DIA templates supported by the system. A template is a hierarchy of zone, section, and artifact folders that contain study documents. The template also has details on security access and metadata elements for the artifacts.

EEL3 Schema -- DIA Template



# Design

## EEL Engine

The engine is the component that implements structure builds and just-in-time artifact creation. Requests are submitted via a web service for a specific study profile. A job is created from the request and then triggered by the Quartz scheduler.

Just-in-time (JIT) requests are also submitted through the web service. If the requested artifact already exists the folder data ID is returned, otherwise it is created on the fly and the new data ID returned.

### Web Service

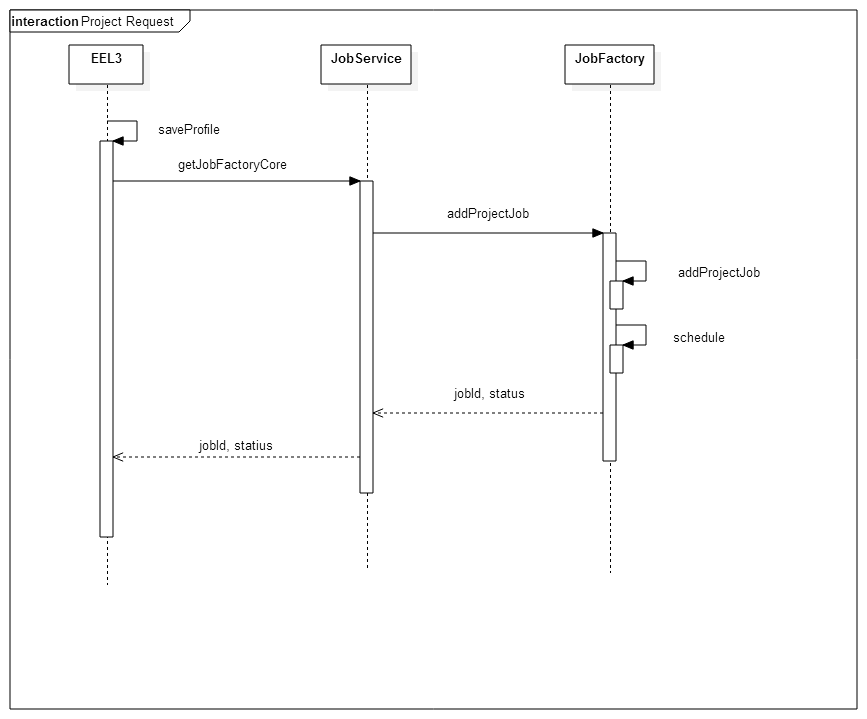
The web service is a REST implementation using the Apache CXF framework. The implementation follows a code-first model where the web service artifacts are generated from the annotated application.

The service uses DAO objects from EEL3 structures to access the database and LAPI to implement the build on CS10.

The following interaction diagrams show how the main flow for key system functions:

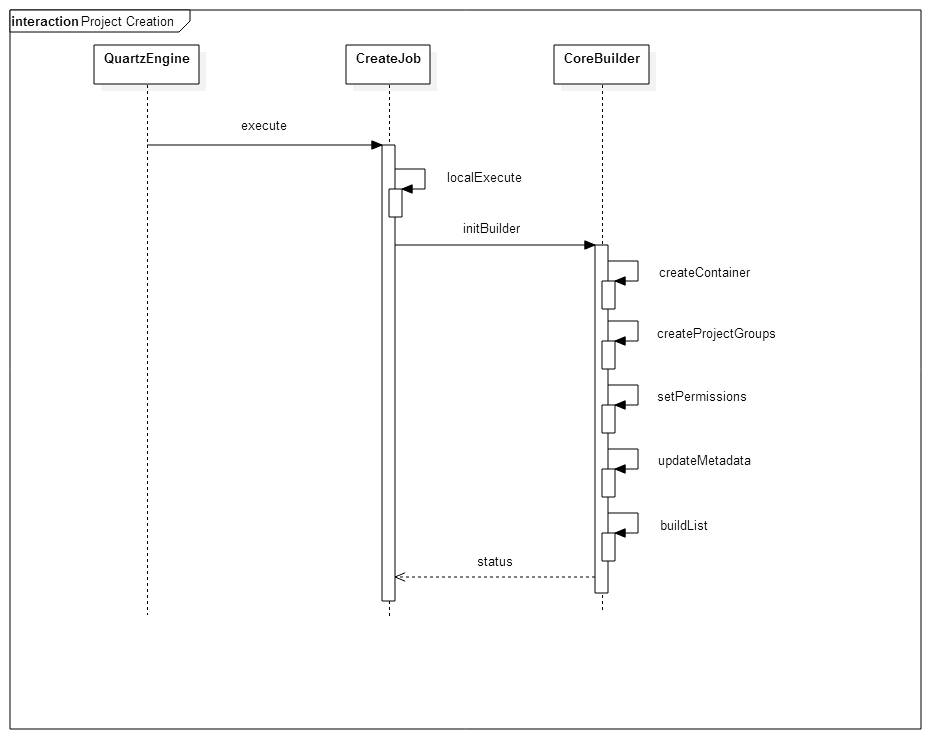
### Project Build Request

The EEL3 application stores a study profile and then requests a build from the engine. If the request is successful a job ID is returned.



### Project Creation Job Execution

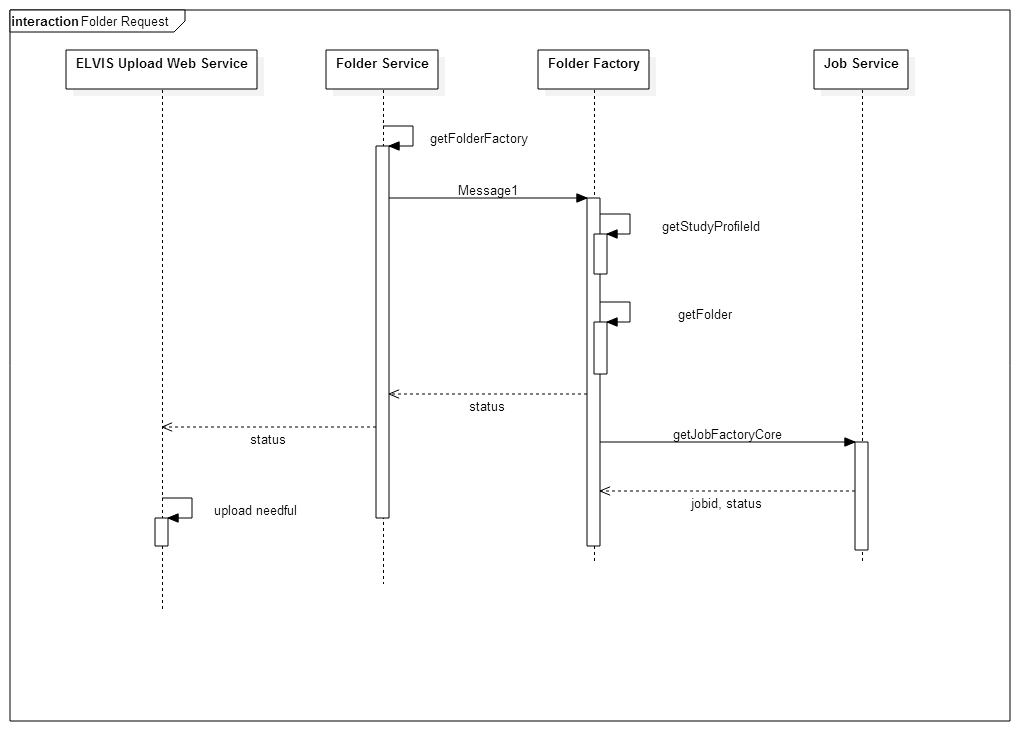
Once a job has been scheduled, the Quartz engine triggers the create job with the requested profile ID.



### Folder Request

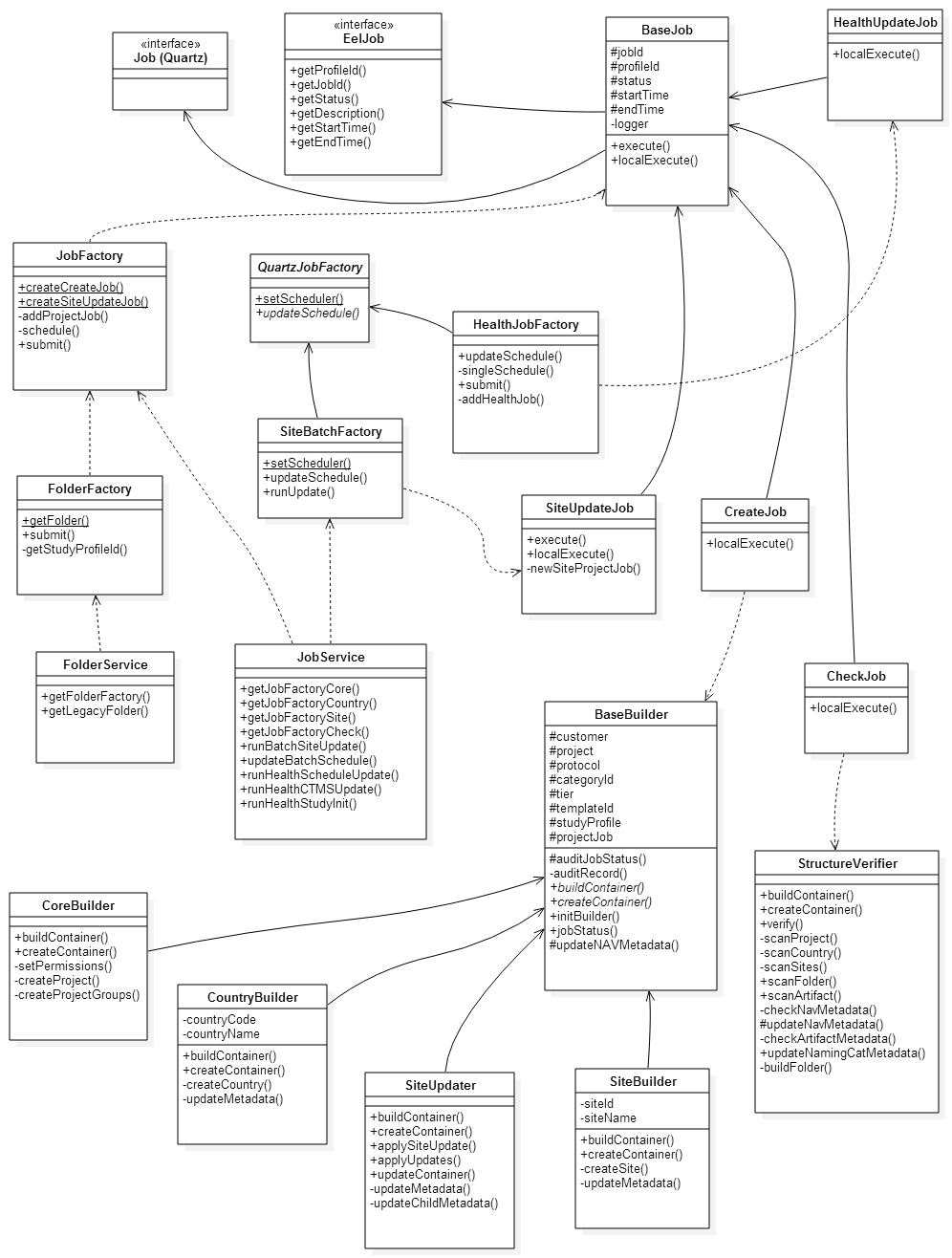
An external system will submit requests for the location (object ID) for a specific project artifact. The web service will locate the item (if it exists) or create it on the fly. In either case, the ID is returned with a status code.

In the event a required container is missing (country or site folder), the engine will schedule a job to build out that part of the structure.



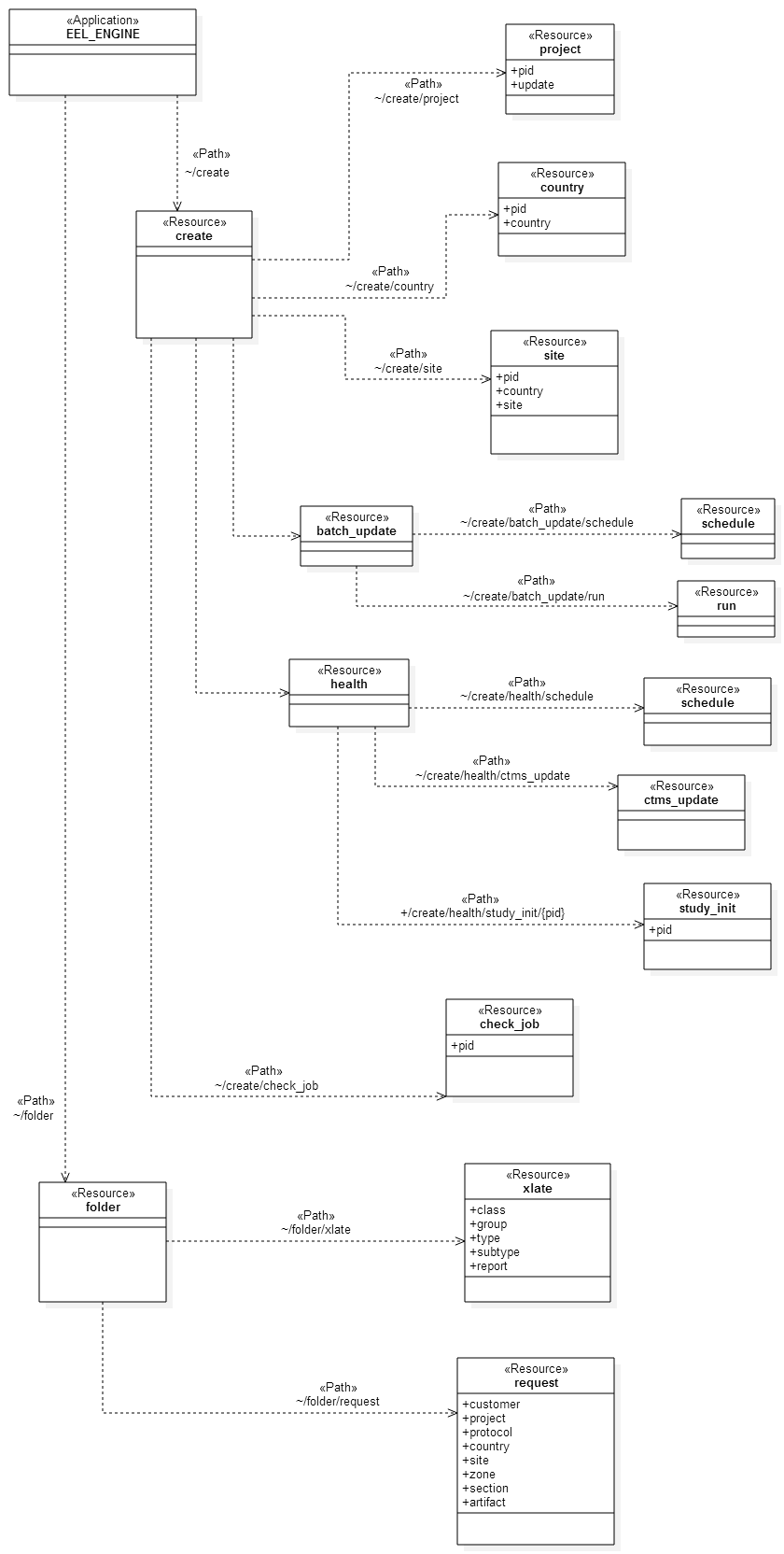
### Principle Classes

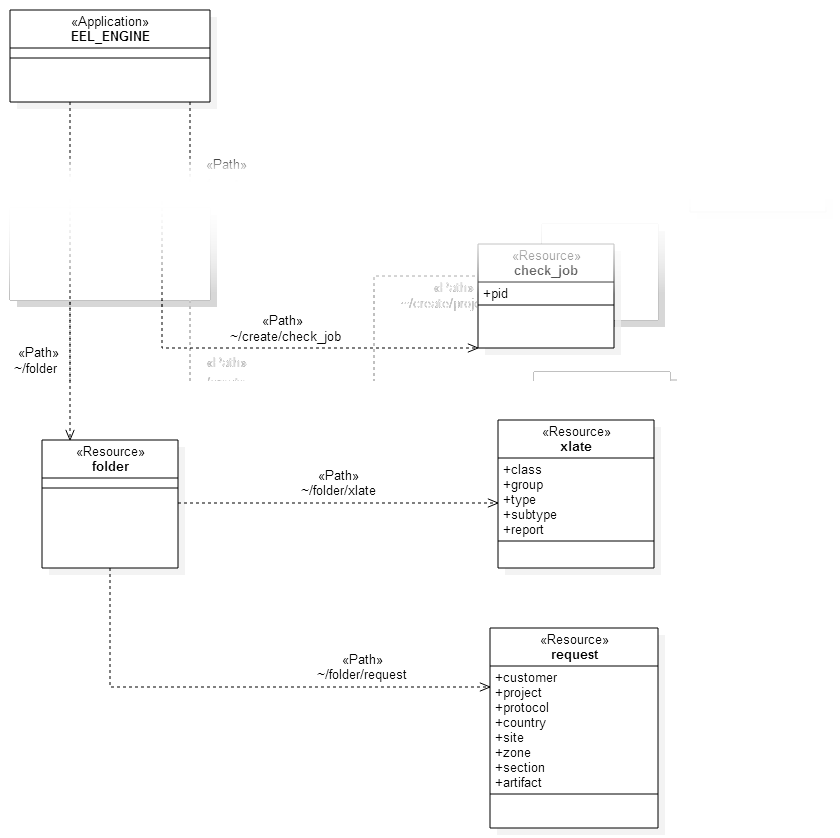
The following diagram illustrates the relationships between principle components in the system:



### REST Service Structure

The following diagrams show the REST service call hierarchy:





## EEL3 Structures

EEL3 Structures is the end user application for creating study profiles, managing the DIA templates, and general system administration. Users can initiate jobs to build study structures and monitor the status of running and completed jobs.

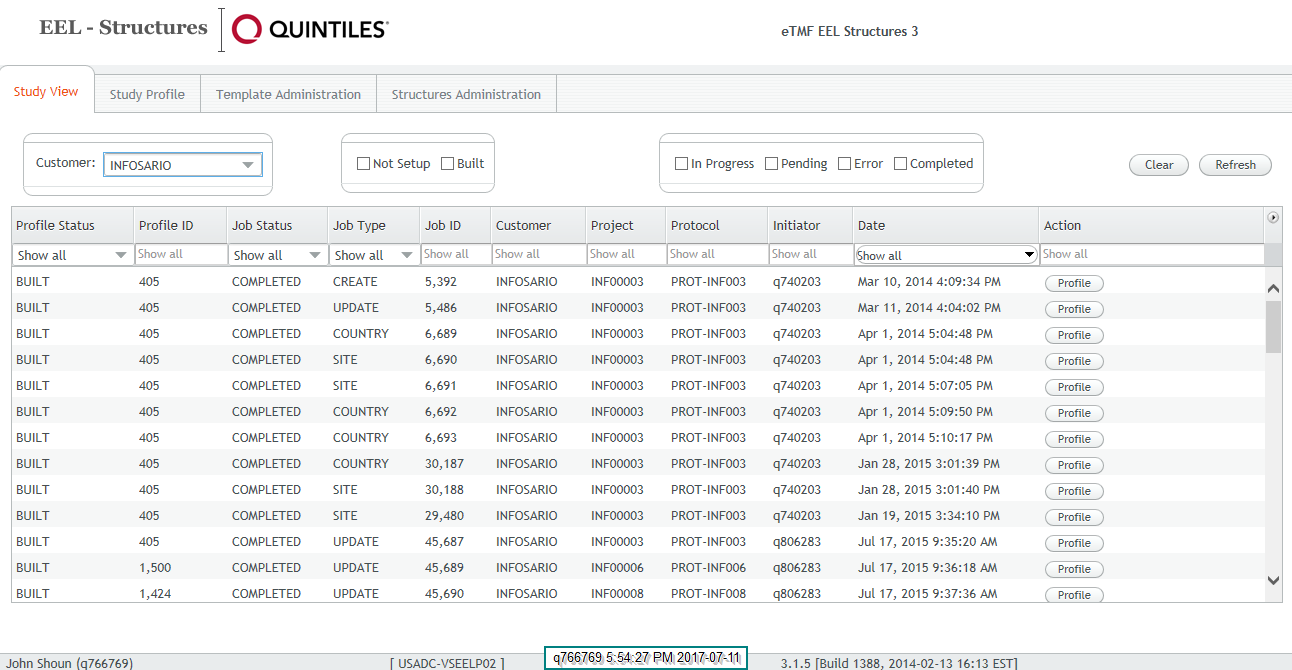
### User Interface

The user interface will be implemented with the Vaadin UI framework. This framework makes it possible to implement a modern user interface without dealing with the details of HTML and Javascript web page implementation. User interface coding follows patterns established with traditional client applications. The user interface has the following major components:

***Study View***

The study view provides a summary view of all studies that have been registered in ELVIS\_REF and whether study structures have been built in EEL2 (ELVIS) or EEL3 (eTMF).

The user can filter the results by sponsor (customer) and the status or the study or jobs for a specific study.

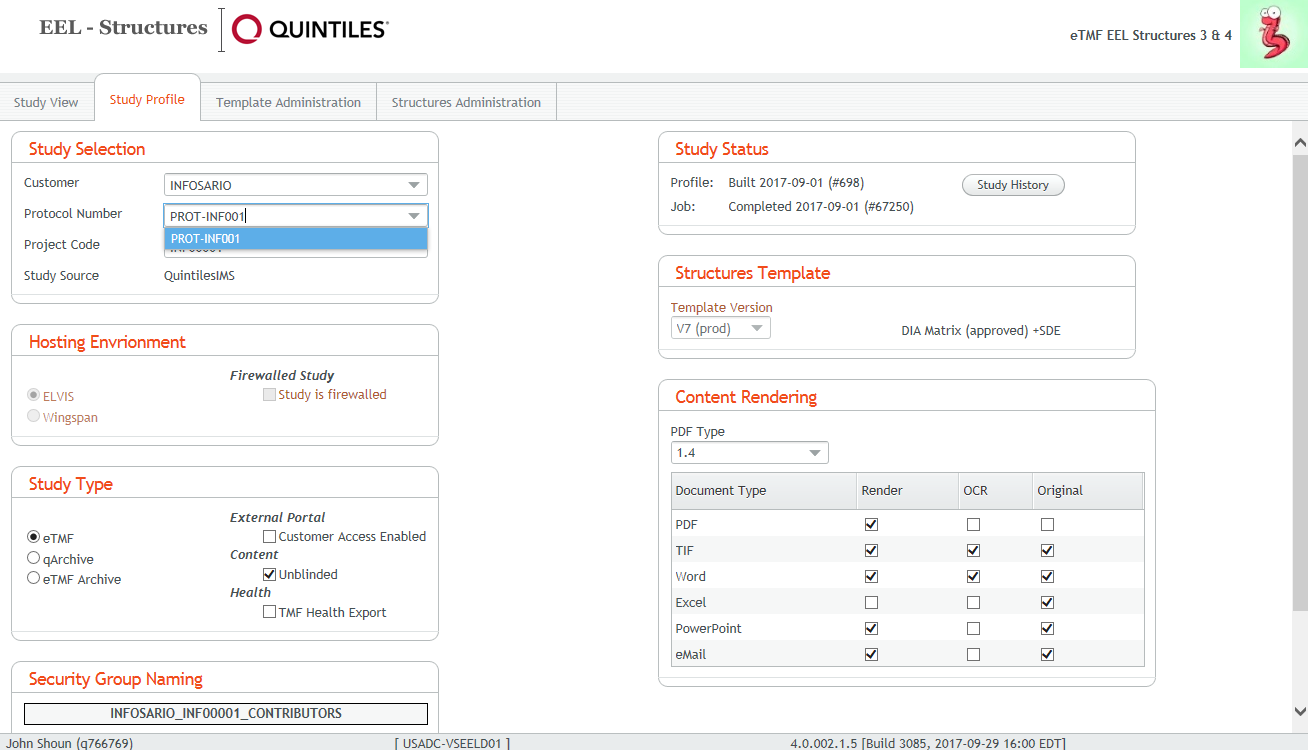


***Study Profile***

The profile view shows the detail for a selected study and allows the user to create a new build profile (if one does not exist), or update and existing EEL3 profile.

A profile can be created for a study that is hosted in an external cloud TMF. Registering the profile will direct CTMS data to be transmitted to the cloud service provider for the study.

The profile view shows the current status of the profile as well as any jobs that are currently running. There is a link to a CS10 report with the complete history of the profile.

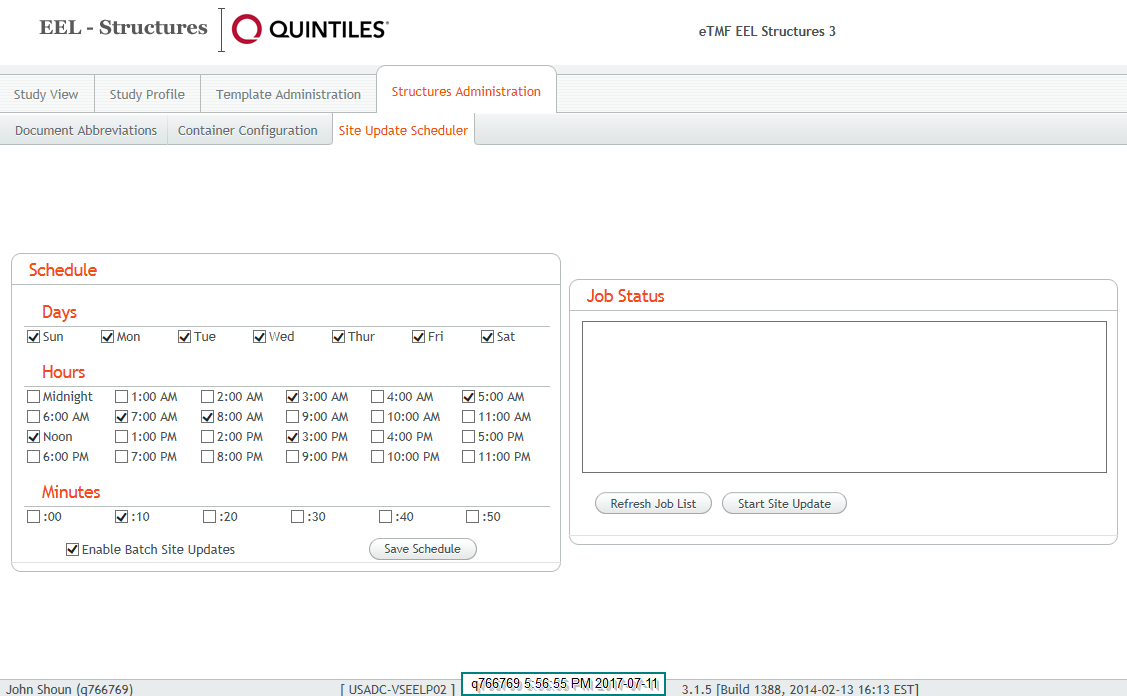


***System Administration***

The system administration page has tabs for various support functions that do not apply to a specific study.

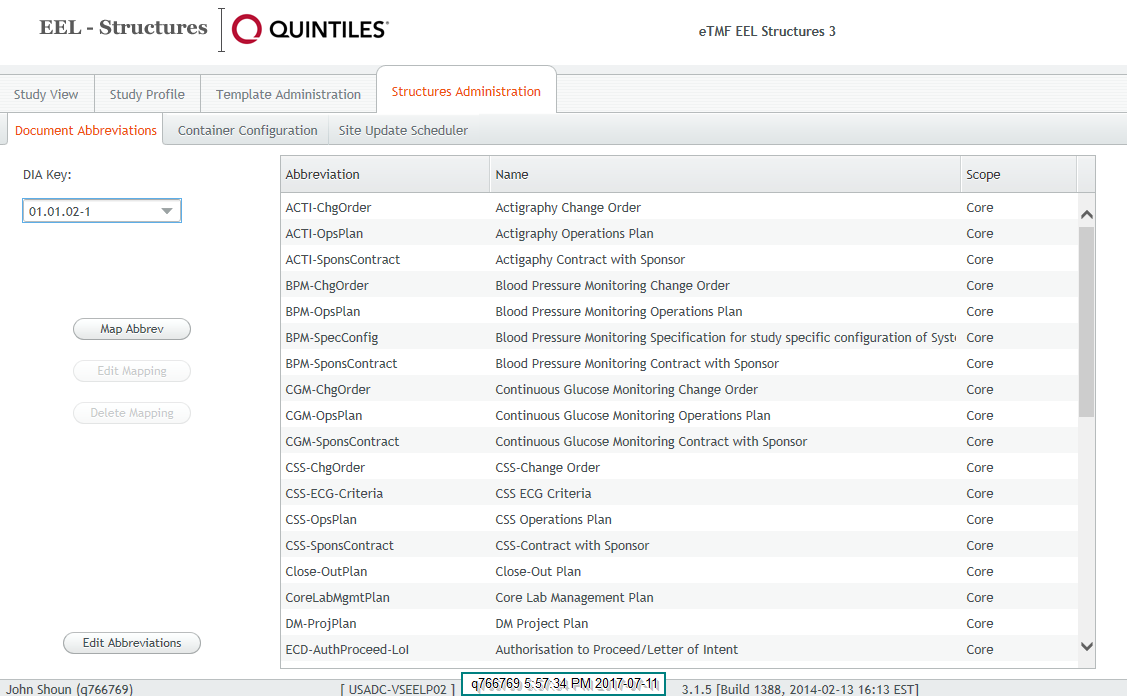
*Batch Job Schedule*

The Batch Job Schedule tab is used to set the run schedule of the automated site update job.



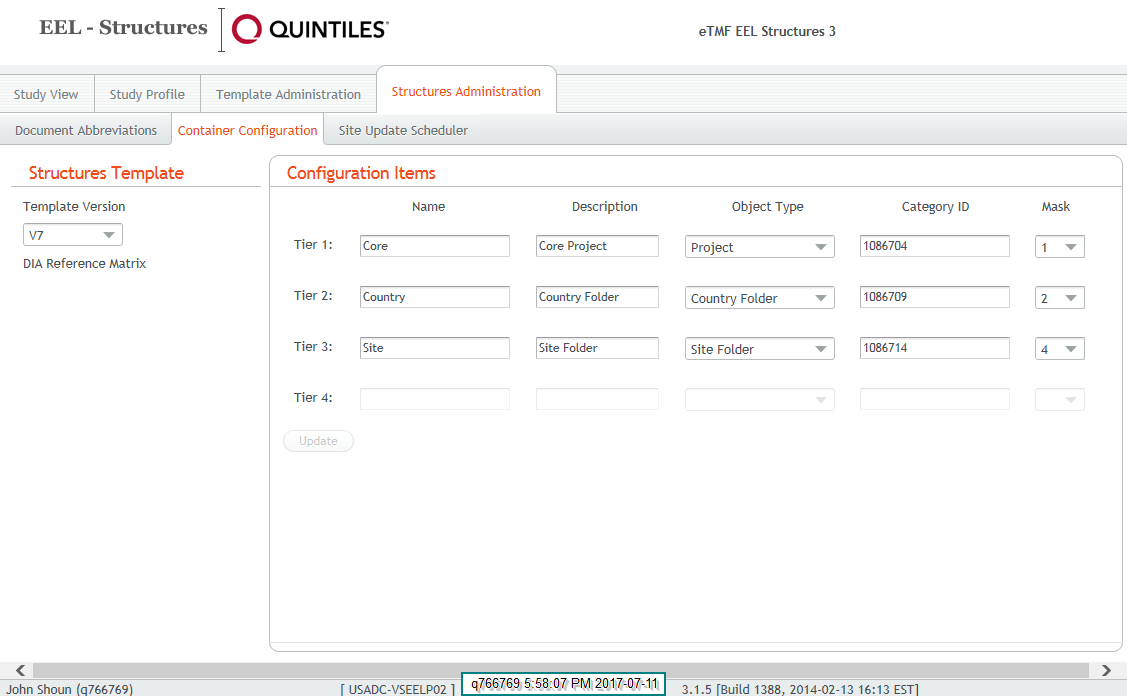
*Document Abbreviations*

The document name tab is used to define the valid document naming abbreviations for each DIA Key.



*Class Configuration*

The class configuration tab is used to define the CS10 object types associated with the different container classes (core, country, site) and define the associated metadata category.



### Code Packages and Classes

* com.quintiles.e3.core
* com.quintiles.e3.core.tree
* com.quintiles.e3.data
* com.quintiles.e3.data.dao
* com.quintiles.e3.data.dao.impl
* com.quintiles.e3.data.dao.model
* com.quintiles.e3.data.dao.model.nodb
* com.quintiles.e3.event
* com.quintiles.e3.exceptions
* com.quintiles.e3.service
* com.quintiles.e3.service.model
* com.quintiles.e3.ui
* com.quintiles.e3.ui.components
* com.quintiles.e3.ui.components.depended
* com.quintiles.e3.ui.components.depended.wrappers
* com.quintiles.e3.ui.providers
* com.quintiles.e3.ui.tabs
* com.quintiles.e3.ui.tabs.admin
* com.quintiles.e3.widgets
* com.quintiles.e3.util

# Definitions, Acronyms and Abbreviations

| Term, Acronym, or Abbreviation | Description |
| --- | --- |
| ELVIS | Electronic Library and Information Vaulting System |
| EEL | ELVIS Electronic Loader |
| DIA | Drug Information Association |
| SEDAN | CTMS (Clinical Trial Management System) Document Interface |

# References

Not Applicable

# Document Revision History

| Unique Identifier for this Revision | Date of the Document Revision  (YYYY-MM-DD) | Significant Changes from Previous Authorized Version |
| --- | --- | --- |
| V1.0 | 04-OCT-2017 | Revised for 4.0 release |
|  |  |  |
|  |  |  |

1. Jira User Stories (Requirements)

*For current release.*

| Key | Summary | Description | Notes |
| --- | --- | --- | --- |
| [IEC-11](http://jira.quintiles.net/browse/IEC-11) | Create study profile for Wingspan TMF. | Study details will be registered in the EEL3 database. | Study profile page will allow selecting the environment (ELVIS or Wingspan) which will then display the appropriate configuration items. |
| [IEC-12](http://jira.quintiles.net/browse/IEC-12) | CTMS study data transfer | On profile creation the CTMS data for the study will be transferred to Wingspan through the REST API. | CTMS data updates will be transmitted incrementally based on a schedule set by system administrators. |

1. EEL3 Database Schema

The database is modelled in Oracle’s Data Modeler.  
The complete schema report is contained in this *embedded PDF*.



# Authorization

|  |  |  |  |
| --- | --- | --- | --- |
| Business Solutions Development Lead (or designee) - This signature indicates review and approval of the content of this document, and that the detail design supports the architectural specification. | | | |
| Sign |  | Date |  |
| Name | Jonathan Kinnaird | Title | Architect 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Business Solutions Development Engineer (or designee) - This signature indicates review and approval of the content of this document, and that the detail design supports the architectural specification. | | | |
| Sign |  | Date |  |
| Name | John Shoun | Title | Architect 1 |