|  |
| --- |
| **Software Design Specification** |

Study Design Web Service

Create Date: 11/2/2011

Created By: Uttara Sakhadeo

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Uttara Sakhadeo | 11/16/11 | Updates for database schema, web service definition | 1.1.0 |
| Sarah Kreidler | 2/20/2012 | Migrated domain object descriptions to separate document “SDS Domain Layer” | 1.2.0 |
|  |  |  |  |

Table of Contents

[**Software Design Specification** 1](#_Toc314555176)

[1 Introduction 4](#_Toc314555177)

[1.1 Purpose of this document 4](#_Toc314555178)

[1.2 Scope of the development project 4](#_Toc314555179)

[1.3 Definitions, Acronyms, and Abbreviations 4](#_Toc314555180)

[1.4 References 4](#_Toc314555181)

[1.5 Overview of the document 4](#_Toc314555182)

[2 System Architecture Description 5](#_Toc314555183)

[2.1 Overview of modules / components 5](#_Toc314555184)

[2.2 Current Callback Structure 5](#_Toc314555185)

[3 Module and Component Descriptions 6](#_Toc314555186)

[3.1 Component overview 6](#_Toc314555187)

[3.1.1 Study Design Object 6](#_Toc314555188)

[3.1.2 Matrices 6](#_Toc314555189)

[3.1.3 Lists 7](#_Toc314555190)

[3.1.4 Class objects 9](#_Toc314555191)

[3.2 Database 10](#_Toc314555192)

[3.2.1 Table Design 10](#_Toc314555193)

[3.2.2 Scripts 14](#_Toc314555194)

[3.3 Testing 14](#_Toc314555195)

[3.3.1 Unit Testing 14](#_Toc314555196)

[3.3.2 Integration Testing 14](#_Toc314555197)

[3.4 Study Identifiers 14](#_Toc314555198)

[3.5 External API 37](#_Toc314555199)

[3.5.1 Study Design Commands 18](#_Toc314555200)

[3.5.2 Matrix Commands 21](#_Toc314555201)

[3.5.3 List Commands 22](#_Toc314555202)

[3.5.4 Between Subject Effect Commands 23](#_Toc314555203)

[3.5.5 Within Subject Effect Commands 24](#_Toc314555204)

[3.5.6 Hypothesis Commands 28](#_Toc314555205)

[3.5.7 Covariance Commands 30](#_Toc314555206)

[3.5.8 Power Curve Description Commands 35](#_Toc314555207)

[3.5.9 Confidence Interval Commands 36](#_Toc314555208)

[3.6 Automated Database Cleanup 42](#_Toc314555209)

[4 Reuse and Relationships to Other Products 43](#_Toc314555210)

[4.1 Reuse of existing code 43](#_Toc314555211)

[4.2 Third Party Dependencies 43](#_Toc314555212)

[5 Design Decisions and Tradeoffs 44](#_Toc314555213)

[6 Appendices 45](#_Toc314555214)

[6.1 Diagrams 45](#_Toc314555215)

# Introduction

## Purpose of this document

This document describes an architectural change in the GLIMMPSE system to improve stability. We introduce a new web service called the Study Design Web Service. The service will store study design definitions for users of the GLIMMSE interface, and will contain all information related to a power or sample size calculation. The Study Design Service will simplify communication between different screens in the user interface. This object will replace the existing callback structure in the GLIMMPSE web-based user interface.

The intended audience for this document is the software engineering team and the grant primary investigators.

## Scope of the development project

The project is an internal architectural change for the user interface. No external functionality will be affected.

## Definitions, Acronyms, and Abbreviations

**CRUD** – An acronym for “create, read, update, delete”. These are the main “verbs” for creating and modifying study designs in the Study Design Service.

**Restlet** – a third party framework for implementing Restful web services.

**Hibernate** – a third party framework for mapping Java objects into relational database tables

**EER diagram –** Entity Relationship model is used to produce a type of conceptual schema of a system.

**JSON –** Java Script Object Notation. A Java/C/C++/and many other language objects can be displayed as text format usin JSON.

## References

## Overview of the document

The document provides a brief review of the existing callback structure in the GLIMMPSE interface. We define the Study Design Service and describe how it will replace the existing callback structure. We sketch the class structure and database schema for the Study Design Service and changes required for existing user interface.

# System Architecture Description

## Overview of modules / components

The GLIMMPSE web user interface (web-GUI) is the browser-based front end of the GLIMMPSE software system. The web-GUI elicits information from the user about the study design and hypothesis of interest. The study design information is converted into a series of matrices used to calculate power or sample size for the general linear multivariate model. The user interface is responsible for:

1. Collecting information from the user
2. Validating the information entered by the user
3. Converting the inputs into matrices required for a power or sample size calculation
4. Submitting the matrices to the Power Service to perform the calculation
5. Receiving and displaying the results to the user
6. Optionally, the user interface may submit the information to the Chart Service and display the resulting power curve.

## Current Callback Structure

Communication between screens in the GLIMMPSE user interface occurs via several listener classes. Screens send callbacks to one another to synchronize the user experience, but there is no central object containing all information about the overall study design. This design underlies several existing bugs in the user interface. In addition, although users can download and upload study designs from GLIMMPSE website, there isn’t any facility for global storage of study designs.

To solve these problems, GLIMMPSE version 2 will add a web service called the Study Design Service. The Study Design Service will provide a central repository of study design information. User interface screens will call into the Study Design Service to access required information about the current study design. This will eliminate the current callback structure and reduce the potential for screen synchronization bugs.

The service will be implemented as a Restlet, running in a Tomcat server. The service will provide a CRUD API for creating and modifying study designs, which will be stored in a MySQL database. Object relational mapping will be implemented with Hibernate.

# Module and Component Descriptions

## Component overview

Study Design service will provide a container, the Study Design, for holding information in the form of lists, matrices, objects and primitive data types. The Study Design contains the following information:

### Study Design Object

The study design object includes some basic information about the type of calculation, and maps to a group of matrices, lists, and other meta information related to the study design. The subcomponents of the study design object are described in each section below. The simple fields for the study design object are:

#### Solving For type :

The solving for flag indicates whether the user is solving for power, sample size, or detectable difference. The value will be an ENUM field, coded as follows.

Selection Value

Power 1

Total Sample Size 2

Detectable Difference 3

#### Gaussian Covariate Flag :

This flag indicates that the user wishes to control for a Gaussian covariate in the power calculation. This will be a Boolean field

### Matrices

Several matrices are associated with the Study Design object. Muller and Stewart provides details on matrix definitions for the univariate and multivariate linear model. The matrices which can be stored with a study design object are listed in the table below

|  |  |  |  |
| --- | --- | --- | --- |
| **Matrix** | **Rows** | **Columns** | **Conformance Details** |
| X: Design essence matrix | N | Q | Columns equal number of rows in β |
| C: Between subject contrast | A | Q | Columns equal number of rows in β  Rows must be one less than number of rows in X |
| U: Within subject contrast | P | B | Rows equals the number of columns in β |
| Θ0: Null hypothesis values | A | B | Rows equals the number of rows in C Columns equals the number of columns in U |
| β: Regression coefficient estimates | Q | P | Rows equals the number of columns in X (number of predictors) Columns equals the number of columns in U (number of outcomes) |
| **GLMM(F) Designs** | | | |
| Σerror: covariance matrix of residuals | P | P | Matrix is square and symmetric. Rows/columns equal the number of columns in β |
| **GLMM(F,g) Designs** | | | |
| ΣG: Covariance of Gaussian predictor | 1 | 1 | Since only a single baseline covariate is allowed, this matrix is 1x1. |
| ΣYG: Covariance of Gaussian predictor with outcomes | P | 1 | Rows equals the number of columns in β |
| ΣY: Covariance of outcomes | P | P | Rows/columns equal the number of columns in β (i.e. number of outcomes) |

### Lists

Using these lists users can specify several statistical tests, power methods (for designs with a baseline covariate), power values, alpha levels, and per group sample size values etc.

Users can scale the regression coefficients in the beta matrix and the variability specified in the sigma matrix. The following lists are stored with the Study Design;

#### Alpha List :

The alpha list specifies various type I error rates.

#### Relative Group Size List :

This list tells the application-middle layer whether the groups are of equal size or of unequal sizes by letting users specify ratio between the entities.

Let us consider an example; suppose there are two groups A and B

Case 1: GLIMMPSE allows users to specify the equality by putting value as 1 in each of the text boxes.

Case 2: Inequality is given by ratios for example 4:1 or 2:1 or so on

#### Per Group Sample Size List :

The per group sample size list includes all desired group sample sizes (reference REPN in powerlib). It is valid for power or detectable difference requests (it is ignored for sample size requests). The total sample size for the power calculation is determined by multiplying the per group size by the number of rows in the design matrix.

For example, consider the following design matrix for a one-way ANOVA with 3 groups:

For a per group sample size of 25, the total sample size would be 75.

#### List of Responses :

This list includes different responses taken at the time of each measurement (e.g. Heart Rate, Blood Pressure) – cannot contain <>, possible not spaces. Add rules for string input.

Validation rules :

1. The responses string or category string cannot contain following symbols <, >
2. No spaces are allowed in between the words.

#### Beta Scale List :

The beta scale list includes all scale factors for the regression coefficient, or beta, matrix. The beta scale list is valid for power and sample size requests (it is ignored for detectable difference requests).

#### Sigma Scale List :

The sigma scale list includes all scale factors for the error covariance matrix. The sigma scale list is required for all requests.

#### Test List :

The test list includes all statistical tests to be run. It is required for all requests. The test list is specified as:

Power values should be equivalent for univariate designs regardless of the test selected. However, no uniformly most powerful test exists for the multivariate linear model. The following tests are supported:

unirep - Univariate approach to repeated measures (uncorrected)

unirepBox - Univariate approach to repeated measures with Box correction

unirepGG - Univariate approach to repeated measures with Geisser-Greenhouse correction

unirepHF - Univariate approach to repeated measures with Huynh-Feldt correction

wl - Wilks' Lambda

hlt - Hotelling-Lawley trace

pbt - Pillai-Bartlett trace

#### Power Method List:

The power method lists includes any combination of the conditional, unconditional, or quantile power methods. It is valid only for study designs involving a baseline covariate (ignored otherwise). Note that only the Hotelling-Lawley and univariate approach to repeated measures are supported for designs with a baseline covariate.

The following power methods are supported:

conditional - Power values conditional on the specific realization of the baseline covariate values

unconditional - Power based on numerical integration over possible values of the non-centrality parameter

quantile - Power based on specific quantiles of the non-centrality parameter's distribution

#### Quantile List :

The quantile list specifies all quantiles for use with the "quantile" power method. This list is valid only for study designs with a baseline covariate, which specify quantile power in the power method list.

### Class objects

For Matrix Mode, only the matrices need to be stored with the study design. For Guided Mode, uses specify additional information about predictor and response names, hypotheses, etc. For study designs produced by Guided Mode, the following objects are stored with the Study Design;

#### Object describing Between Subject Effect:

Between subject effects describe fixed predictors and the associated list of possible values. This object contains following fields.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Description** |
| Predictor name | String | Name of the between participant factor |
| Category list | List | Names of all valid categories for this factor |

This object is required for producing the X matrix. For Guided Mode, both the between subject effect information and associated X matrix will be stored in the database.

#### Object describing Within Subject Effect:

This object contains following details.

1. Information about repeated measures.
2. Information regarding clustering.

See the “Software Design Specification: Multilevel and Longitudinal Study Designs” for full details on within subject effect information

#### Hypothesis Object:

This object contains information regarding the primary hypothesis of interest. Hypothesis objects describe the type of hypothesis, and all variables associated with the hypothesis.

See the “Software Design Specification: Full Factor Hypothesis Selection” for full details on hypothesis specification.

#### Covariance Information

The Study Design object will store information regarding the covariance for multilevel and longitudinal designs.

Covariance information can be specified either as structured or unstructured, and parameterized by covariance or correlation plus standard deviation. For matrix mode, only the sigma matrices will be stored. For guided mode, Lear model information for structured matrices will be stored.

See the “Software Design Specification: Complex Covariance Specification” for full details on hypothesis specification

#### Power Curve description Object:

This object holds information required for drawing power curve. This includes the following information:

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Description** |
| XAxis Type | String | Variable to display on the X-axis |
| Stratification Variable | String | Displays multiple curves based on the stratification variable |
| Remaining Variable Values | List | List of values which are fixed for the remaining variables |

#### Confidence Interval Object:

This object describes the type of confidence interval to produce around each power value. It contains the following variable instances;

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Description** |
| BetaFixed | Boolean | Indicates if the beta matrix is assumed fixed |
| SigmaFixed | Boolean | Indicates weather Sigma matrix is fixed |
| LowerTailProbability | Double | Lower tail probability for the confidence interval |
| UpperTailProbability | Double | Upper tail probability for the confidence interval |
| Sample Size | Int | Sample size of the data from which the beta and sigma values were obtained |
| Design matrix rank | Int | Rank of the design matrix for the model from which the beta and sigma values were obtained |

## Database

### Table Design

All the matrices, lists and objects will be stored in proper data base tables. The Database Table design is available in the **StudyDesignService\_version\_1\_0\_0\_databaseSchema.mwb** file. See the attached PDF document for a global view of the database schema.

Please refer this Entity-Relationship-Diagram file for more details;

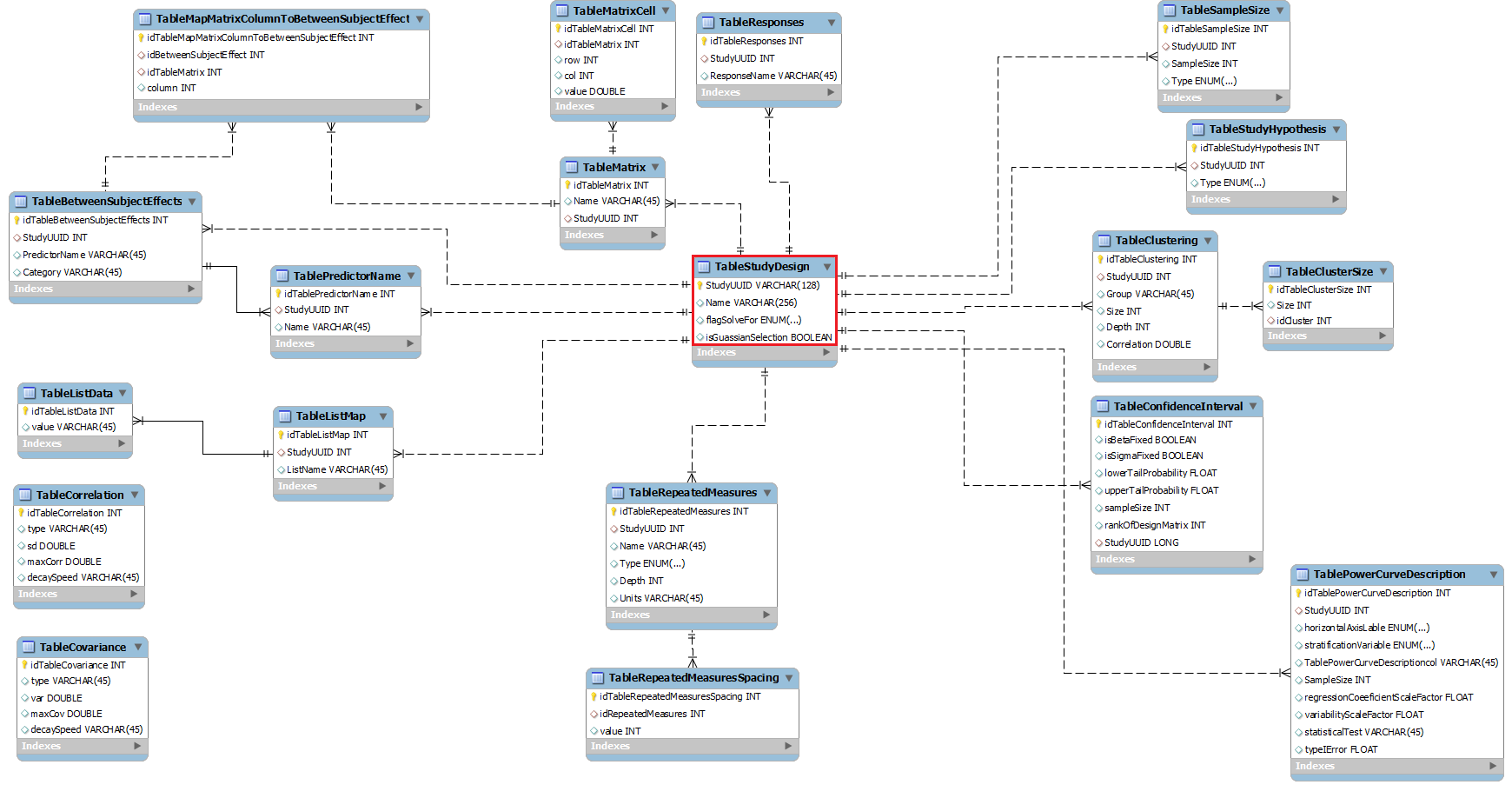


Figure1 : Complete Data Base Schema

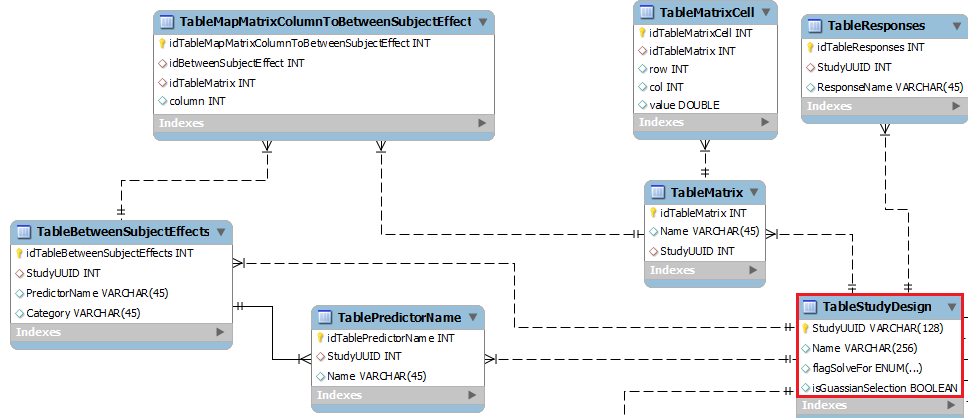


Figure2 : Data Base Schema : Part of complete schema

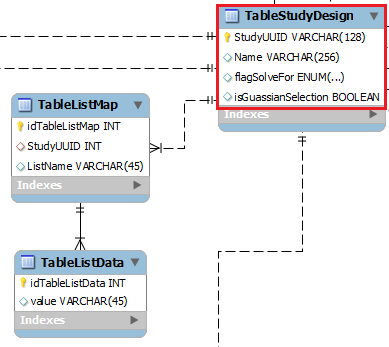


Figure3 : Data Base Schema : Part of complete schema

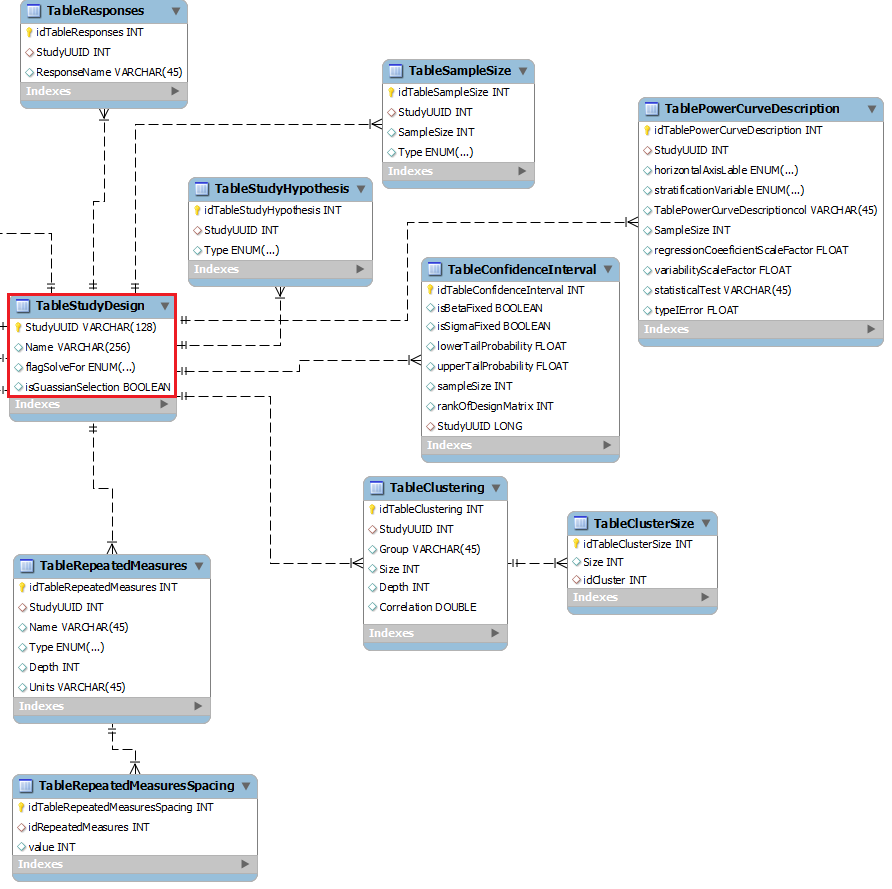


Figure4 : Data Base Schema : Part of complete schema

EER diagram description :



Two related entities shown using Crow's Foot notation. In this example, an optional relationship is shown between Artist and Song; the symbols closest to the song entity represents "zero, one, or many", whereas a song has "one and only one" Artist. The former is therefore read as, an Artist (can) perform(s) "zero, one, or many" song(s).

Brief description/glossary of SQL diagram

### Scripts

Scripts will be provided to perform the following database activities:

* Creation of a new database and all associated tables
* A script to remove expired study designs from the database
* Need erase history function

## Testing

### Unit Testing

Here newly created modules will be tested for against their functionality. This testing will be done as per predefined test cases (e.g. for functions etc.)

#### Unit tests for Hibernate

With the help of JUnit data access layer can be tested. Hibernate has lots of files, hence unit testing for each of such file is required. Here are some of the file types used in Hibernate

1. Persistent (POJO) classes – Plain Old Java Object classes are used to map to database table.
2. Hibernate Mapping file (hbm.xml) – This file maps POJO class object with corresponding table.
3. Hibernate Configuration File(hibernate.cfg.xml) – This file is used to create connection pool and to setup required environment.
4. Java files for accessing data from tables – Actual queries will be written here.

#### Unit tests for Protocols

Protocols like xml parsing are also needed to be tested. Some existing methodology for such a unit testing will be reused here.

### Integration Testing

All the separate modules after integrating together will be tested to assure that required functionality is provided.

## Study Identifiers

When a user creates a new study design, a UUID will be assigned to it. The study design UUID will be used to synchronize the overall study design across multiple database tables. The GLIMMPSE UI screens will retrieve study design information via the study UUID. Users can retrieve stored study designs by providing this UUID.

Structure of UUID:

In general UUIDs are either 16-byte (128-bit) or 32-byte long.

Here are some examples of UUIDs:

067e6162-3b6f-4ae2-a171-2470b63dff00

54947df8-0e9e-4471-a2f9-9af509fb5889

Expected Problems:

Now there are two important issues with such a UUID based design;

1. User has to remember each of such UUID. This information may be stored as session cookies. In subsequent versions, these may be associated with a user email or login/password combination.
2. Any user can access any study design. This can lead to a problem of security of such study designs.

UUID duplication probability:

These UUIDs are supposed to be ‘practically unique’ rather than ‘guaranteed unique’.

There are 16^32 = 340,282,366,920,938,463,463,374,607,431,768,211,456 possible UUIDs

Thus the probability of creating a few tens of trillions of UUIDs in a year and having one duplicate is 0.00000000006

Note:

Each created study design will be removed from data base after a month.-with erase all history option

## Objects in the Study Design Service

All the screens will communicate with Study Design Service through JSON. These screens will populate all the information in the objects and then forward these objects using JSON to Study Design Service. Study Design Service then will take these objects and store them in database using Hibernate. Or otherwise Study Design Service will fetch the objects from data base using Hibernate and send back to screens using JSON.

Following objects will be referred in Study Design Service;

### Study Design Object

#### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| uuid | Byte[] | For persistence UUID to byte conversion is saved in here. |
| studyUUID | UUID | Holds newly created UUID |
| name | String | It describes name of the study design. |
| flagSolveFor | Enum | It describes the solving type is POWER or SAMPLE SIZE. |
| hasGuassianCovariate | bollean | It is a flag for Gaussian Covariate. |

#### Sub-objects

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| lists | List<NamedList> | Named lists like alpha list, sample size list etc. are stored in this list.  This list will contain total nine named lists. |
| matrices | List<NamedRealMatrix> | Named matrices like Theta, Between subject contrast etc. are stored in this list.  This list will contain total ‘’ named matrices. |
| confidenceInterval | ConfidenceInterval | This object stores information such as type of beta and sigma matrices (fixed/variable), sample size, probabilities etc. |
| betweenSubjectEffect | BetweenSubjectEffect |  |
| withinSubjectEffect | withinSubjectEffect |  |
| repeatedMeaseues | RepeatedMeasures |  |
| powerCurve | PowerCurve |  |
| responseList | ResponseList |  |
| hypothesis | Hypothesis |  |

### Confidence Interval Object

#### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| id | int | This id will be used as primary key for this object. |
| studyUUID | UUID | Holds newly created UUID. |
| isBetaFixed | Boolean | Describes the fixed structure of beta matrix. |
| isSigmaFixed | Boolean | Describes the fixed structure of sigma matrix. |
| lowerTailProbability | Float | Probability of lower tail is stored in this variable. |
| upperTailProbability | Float | Probability of upper tail is stored in this variable. |
| sampleSize | Int | This field provides used the sample size. |
| rankOfDesignMatrix | Int | This filed stores rank of the design matrix. |

#### Sub-objects

None

### Power Curve Object

#### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| id | int | This id will be used as primary key for this object. |
| studyUUID | UUID | This field will reference UUID stored in study design object. |
| horizontalAxisLabel | Enum | This filed describes the label given to horizontal axis. |
| strtificationVariable | Enum |  |
| powerCurveDescriptionColumn | String | This field stores description for the power curve. |
| sampleSize | Int | This field provides used the sample size. |
| regressionCoefficientScaleFactor | Float | This field stores regression coefficient. |
| variabilityScaleFactor | Float | A variability scale factor is stored in this field. |
| statisticalTest | String |  |
| typeIError | Float |  |

#### Sub-objects

None

### Hypothesis Object

#### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| id | int | This id will be used as primary key for this object. |
| studyUUID | UUID | This field will reference UUID stored in study design object. |
| type | Enum | A type of study hypothesis will be stored in this field. Here is a list of possible types;  Main Effects, Interaction, linear trend etc. |

#### Sub-objects

None

### List Object

#### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| studyUUID | UUID | This field will reference UUID stored in study design object. |
| name | String | Name of the list is specified in here. |
| dataList | List<String> | This filed holds the list of the values in given named list object. |

#### Sub-objects

None

### Matrix Object

#### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| studyUUID | UUID | This field will reference UUID stored in study design object. |
| name | String | Name of the matrix is specified in here. |
| data | Double[][] | This filed holds the values in given named matrix object. |

#### Sub-objects

None

### Within Subject Effect Object

#### Fields

None

#### Sub-objects

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| responseList | ResponseList | This object stores the responses like Heart Rate etc. |
| repeatedMeasures | RepeatedMeasures | This object stores the repeated measures information. |
| clustering | ClusterNode | This object stores clustering information. |

##### ResponseList sub-object

###### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| studyUUID | UUID | This field will reference UUID stored in study design object. |
| name | String | Name of the response is stored here |
| dataList | List<String> | Values in this list are stored in this List object. |

###### Sub-objects

None

##### RepeatedMeasures sub-object

###### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| studyUUID | UUID | This field will reference UUID stored in study design object. |
| name | String | Name of the dimension is stored here |
| Type | String | Dimension type is stored here. |
| Count | int | Dimension count is stored here. |
| Spacing |  |  |

###### Sub-objects

None

### Between Subject Effect Object

One study design will contain many between subject effect objects.

#### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| Id | Int | This id will be used as primary key for this object. |
| studyUUID | UUID | This field will reference UUID stored in study design object. |
| predictorName | String | Name of the predictor |
| categoryList | List<String> | This filed holds the values in given named matrix object. |

#### Sub-objects

None

### Covariance Object

#### Fields

#### Sub-objects

### Study Design Commands

#### Create / Update

This command creates a study design with the specified study UUID from the existing study design with the specified UUID

**Create URI:** POST /study/study/

**Update URI:** PUT /study/study/<UUID>

**XML Request:**

*<study UUID=””>*

*<alphaList><v>value1</v>...<v>valueN</v></alphaList >*

*<relativeGroupSizeList><v>value1</v>...<v>valueN</v></relativeGroupSizeList>*

*<sampleSizeList><v>value1</v>...<v>valueN</v></sampleSizeList>*

*<responseList><v>value1</v>...<v>valueN</v></responseList>*

*<betaScaleList><v>value1</v>...<v>valueN</v></betaScaleList>*

*<sigmaScaleList><v>value1</v>...<v>valueN</v></sigmaScaleList>*

*<testList><v>value1</v>...<v>valueN</v></ testList>*

*<powerMethodList><v>value1</v>...<v>valueN</v></powerMethodList>*

*<quantileList><v>value1</v>...<v>valueN</v></quantileList>*

*<matrix name='design' rows='n' columns='q'>*

*<r><c>d1,1</c><c>d1,2</c>...<c>d1,q</c></r>*

*...*

*<r><c>dn,1</c><c>dn,2</c>...<c>dn,q</c></r>*

*</matrix>*

*<fixedRandomMatrix name='betweenSubjectContrast' combineHorizontal='true' >*

*<matrix name='fixed' rows='a' columns='q'>*

*<r><c>d1,1</c><c>d1,2</c>...<c>d1,q</c></r>*

*...*

*<r><c>da,1</c><c>da,2</c>...<c>da,q</c></r>*

*</matrix>*

*<matrix name='random' rows='a' columns='1'>*

*<r><c>d1,1</c></r>*

*...*

*<r><c>da,1</c></r>*

*</matrix>*

*</fixedRandomMatrix>*

*<matrix name='withinSubjectContrast' rows='p' columns='b'>*

*<r><c>d1,1</c><c>d1,2</c>...<c>d1,b</c></r>*

*...*

*<r><c>dp,1</c><c>dp,2</c>...<c>dp,b</c></r>*

*</matrix>*

*<matrix name='theta' rows='a' columns='b'>*

*<r><c>d1,1</c><c>d1,2</c>...<c>d1,b</c></r>*

*...*

*<r><c>da,1</c><c>da,2</c>...<c>da,b</c></r>*

*</matrix>*

*<fixedRandomMatrix name='beta' combineHorizontal='true' >*

*<matrix name='fixed' rows='q' columns='p'>*

*<r><c>d1,1</c><c>d1,2</c>...<c>d1,p</c></r>*

*...*

*<r><c>dq,1</c><c>dq,2</c>...<c>dq,p</c></r>*

*</matrix>*

*<matrix name='random' rows='1' columns='p'>*

*<r><c>d1,1</c><c>d1,2</c>...<c>d1,p</c></r>*

*</matrix>*

*</fixedRandomMatrix>*

*<!-- Specify sigmaError if all fixed predictors -->*

*<matrix name='sigmaError' rows='p' columns='p'>*

*<r><c>d1,1</c><c>d1,2</c>...<c>d1,p</c></r>*

*...*

*<r><c>dp,1</c><c>dn,2</c>...<c>dp,p</c></r>*

*</matrix>*

*<!-- Specify the following three sigma matrices if controlling for a baseline covariate -->*

*<matrix name='sigmaGaussianRandom' rows='1' columns='1'>*

*<r><c>d1,1</c></r>*

*</matrix>*

*<matrix name='sigmaOutcome' rows='p' columns='p'>*

*<r><c>d1,1</c><c>d1,2</c>...<c>d1,p</c></r>*

*...*

*<r><c>dp,1</c><c>dn,2</c>...<c>dp,p</c></r>*

*</matrix>*

*<matrix name='sigmaOutcomeGaussianRandom' rows='p' columns='1'>*

*<r><c>d1,1</c></r>*

*...*

*<r><c>dp,1</c></r>*

*</matrix>*

*…*

*</study>*

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<study UUID=””>*

*<matrix name=””…*

*<alphaList type=’’>…*

*…*

*</study>*

#### Read

Read the study design. The user may request either the matrix representation only, or the full study design including both matrices and meta information.

**Read full study design:**

**URI**: GET /study/study/<UUID>/verbose

**XML Request**: none, contained in URI

**XML Response**:

*<study UUID=””>*

*<matrix name=””…*

*<alphaList type=’’>…*

*…*

*</study>*

**Read study design in matrix representation:**

**URI**: GET /study/study/<UUID>

**XML Request**: none, contained in URI

**XML Response**:

*<study UUID=””>*

*<matrix name=””>*

*<matrix name=””>*

*…*

*</study>*

#### Delete

This command deletes a study design with the specified study UUID from the existing study design with the specified UUID

**URI:** DELETE /study/study/<UUID>

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<study UUID=””>*

*<matrix name=””…*

*<alphaList type=’’>…*

*…*

*</study>*

For administrative purposes, we also provide a command to delete all study designs prior to a specified date. This will be used for database cleanup. Dates must be ISO 8601 formatted (ex. 2012-01-01). The syntax for this command is

**URI:** DELETE /study/study/date/<iso-date>

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<studyDesignList count=”numberDeleted” />*

### Matrix Commands

The following commands allow you to modify matrices associated with an existing study design.

#### Create / Update

This command adds a matrix to the study design with the specified UUID.

**Create URI:** POST /study/study/<UUID>/matrix

**Update URI:** PUT /study/study/<UUID>/matrix

**XML Request:**

*<matrix name=’matrix’ rows=’p’ columns=’p’>*

*<r> <c> a1,1</c> <c> a1,2</c> ….. <c> a1,p</c> </r>*

*…*

*<r> <c> ap,1</c> <c> ap,2</c> ….. <c> ap,p</c> </r>*

*</matrix>*

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<matrix name=’matrix’ rows=’p’ columns=’p’>*

*<r> <c> a1,1</c> <c> a1,2</c> ….. <c> a1,p</c> </r>*

*…*

*<r> <c> ap,1</c> <c> ap,2</c> ….. <c> ap,p</c> </r>*

*</matrix>*

#### Read All Matrices

This command retrieves all matrices for the study design with the specified UUID.

**URI**: GET /study/study/<UUID>/matrix

**XML Request**: none, contained in URI

**XML Response**:

*<study UUID =” UUID”>*

*<matrix name=’name1’ rows=’p’ columns=’p’>*

*<r> <c> a1,1</c> <c> a1,2</c> ….. <c> a1,p</c> </r>*

*…*

*<r> <c> ap,1</c> <c> ap,2</c> ….. <c> ap,p</c> </r>*

*</matrix>*

*<matrix name=’name2’ rows=’p’ columns=’p’>*

*<r> <c> a1,1</c> <c> a1,2</c> ….. <c> a1,p</c> </r>*

*…*

*<r> <c> ap,1</c> <c> ap,2</c> ….. <c> ap,p</c> </r>*

*</matrix>*

*…*

*</study>*

#### Read a Single Matrix

This command retrieves a matrix with the specified name for the study design with the specified UUID.

**URI**: GET /study/study/<UUID>/matrix/<name>

**XML Request:** none, contained in URI

**XML Response:**

*<matrix name=’name’ rows=’p’ columns=’p’>*

*<r> <c> a1,1</c> <c> a1,2</c> ….. <c> a1,p</c> </r>*

*…*

*<r> <c> ap,1</c> <c> ap,2</c> ….. <c> ap,p</c> </r>*

*</matrix>*

#### Delete

This command deletes a matrix with the specified name from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/matrix/<name>

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<matrix name=’name’ rows=’p’ columns=’p’>*

*<r> <c> a1,1</c> <c> a1,2</c> ….. <c> a1,p</c> </r>*

*…*

*<r> <c> ap,1</c> <c> ap,2</c> ….. <c> ap,p</c> </r>*

*</matrix>*

### List Commands

The following commands allow the user to modify lists associated with an existing Study Design.

#### Create / Update

This command adds a list to the study design with the specified UUID.

**Create URI:** POST /study/study/<UUID>/list

**Update URI:** PUT /study/study/<UUID>/list

**XML Request:**

<*name*List><v>*value1*</v>...<v>*valueN*</v></*name*List>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*name*List><v>*value1*</v>...<v>*valueN*</v></*name*List>

#### Read a single list

This command retrieves the list of the specified name for the study design with the specified UUID.

**URI:** GET /study/study/<UUID>/list/<name>

**XML Request:** none, contained in URI

**XML Reposnse:**

<*name*List><v>*value1*</v>...<v>*valueN*</v></*name*List>

#### Read all lists

This command retrieves the list of the specified name for the study design with the specified UUID.

**URI:** GET /study/study/<UUID>/list

**XML Request:** none, contained in URI

**XML Reposnse:**

<study uuid=”uuid”>

<*name*List><v>*value1*</v>...<v>*valueN*</v></*name*List>

…

</study>

#### Delete

This command deletes a list with the specified name from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/list/<name>

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*name*List><v>*value1*</v>...<v>*valueN*</v></*name*List>

### Between Subject Effect Commands

#### Create/ Update

This command adds a ‘Between Subject Effect class object’ to the study design with the specified UUID.

**Create URI:** POST /study/study/<UUID>/predictorList

**Update URI:** PUT /study/study/<UUID>/predictorList

**XML Request:**

<*predictorList>*

*<predictor Name=”A”*>

<categoryList><v>*value1*</v>...<v>*valueN*</v></categoryList>

</predictor>

*<predictor Name=”B”*>

<categoryList><v>*value1*</v>...<v>*valueN*</v></categoryList>

</predictor>

…

</predictorList>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*predictorList>*

*<predictor Name=”A”*>

<categoryList><v>*value1*</v>...<v>*valueN*</v></categoryList>

</predictor>

*<predictor Name=”B”*>

<categoryList><v>*value1*</v>...<v>*valueN*</v></categoryList>

</predictor>

…

</predictorList>

#### Read

This command retrieves a ‘Between Subject Effect class object’ from study design with the specified UUID.

**URI:** GET /study/study/<UUID>/predictorList

**XML Request:** none, contained in URI

**XML Reposnse:**

<*predictorList>*

*<predictor Name=”A”*>

<categoryList><v>*value1*</v>...<v>*valueN*</v></categoryList>

</predictor>

*<predictor Name=”B”*>

<categoryList><v>*value1*</v>...<v>*valueN*</v></categoryList>

</predictor>

…

</predictorList>

#### Delete

This command deletes a ‘Between Subject Effect class object’ with the specified name from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/predictorList

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*predictorList>*

*<predictor Name=”A”*>

<categoryList><v>*value1*</v>...<v>*valueN*</v></categoryList>

</predictor>

*<predictor Name=”B”*>

<categoryList><v>*value1*</v>...<v>*valueN*</v></categoryList>

</predictor>

…

</predictorList>

### Within Subject Effect Commands

#### Response List

##### Create/ Update

This command adds a ‘Within Subject Effect’ class object’s Response list to the study design with the specified UUID.

**Create URI:** POST /study/study/<UUID>/responseList

**Update URI:** PUT /study/study/<UUID>/responseList

**XML Request:**

<*responseList>*

*<v>value</v>……<v>valueN</v>*

</responseList>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*responseList>*

*<v>value</v>……<v>valueN</v>*

</responseList>

##### Read

This command retrieves a ‘Within Subject Effect’ class object’s Response list from study design with the specified UUID.

**URI:** GET /study/study/<UUID>/responseList

**XML Request:** none, contained in URI

**XML Reposnse:**

<*responseList>*

*<v>value</v>……<v>valueN</v>*

</responseList>

##### Delete

This command deletes a ‘Within Subject Effect’ class object’s Response list with the specified name from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/responseList

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*responseList>*

*<v>value</v>……<v>valueN</v>*

</responseList>

#### Clustering

##### Create/ Update

This command adds a ‘Within Subject Effect’ class object’s clustering information to the study design with the specified UUID.

**Create URI:** POST /study/study/<UUID>/clustering

**Update URI:** PUT /study/study/<UUID>/clustering

**XML Request:**

<*clustering>*

*<cluster name=”Country1”>*

*<cluster name=”State1”><sample name=”participant” size=”50”/></cluster>*

*</cluster>*

*<cluster name=”Country2”>*

*<cluster name=”State1”><sample name=”participant” size=”50”/></cluster>*

*<cluster name=”State2”><sample name=”participant” size=”100”/></cluster>*

*</cluster>*

</*clustering* >

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*clustering>*

*<cluster name=”Country1”>*

*<cluster name=”State1”><sample name=”participant” size=”50”/></cluster>*

*</cluster>*

*<cluster name=”Country2”>*

*<cluster name=”State1”><sample name=”participant” size=”50”/></cluster>*

*<cluster name=”State2”><sample name=”participant” size=”100”/></cluster>*

*</cluster>*

</*clustering* >

##### Read

This command retrieves a ‘Within Subject Effect’ class object’s clustering information from study design with the specified UUID.

**URI:** GET /study/study/<UUID>/clustering

**XML Request:** none, contained in URI

**XML Reposnse:**

<*clustering>*

*<cluster name=”Country1”>*

*<cluster name=”State1”><sample name=”participant” size=”50”/></cluster>*

*</cluster>*

*<cluster name=”Country2”>*

*<cluster name=”State1”><sample name=”participant” size=”50”/></cluster>*

*<cluster name=”State2”><sample name=”participant” size=”100”/></cluster>*

*</cluster>*

</*clustering* >

##### Delete

This command deletes a ‘Within Subject Effect’ class object’s clustering information from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/clustering

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*clustering>*

*<cluster name=”Country1”>*

*<cluster name=”State1”><sample name=”participant” size=”50”/></cluster>*

*</cluster>*

*<cluster name=”Country2”>*

*<cluster name=”State1”><sample name=”participant” size=”50”/></cluster>*

*<cluster name=”State2”><sample name=”participant” size=”100”/></cluster>*

*</cluster>*

</*clustering* >

#### Repeated Measures

##### Create/ Update

This command adds a ‘Within Subject Effect’ class object’s repeated measures information to the study design with the specified UUID.

**Create URI:** POST /study/study/<UUID>/repeatedMeasures

**Update URI:** PUT /study/study/<UUID>/repeatedMeasures

**XML Request:**

<*repeatedMeasures>*

*<dimension name=”time” type=”numeric” count=”5”>*

*[<spacing><v>5</v>….<v>15</v></spacing>]*

*<dimension name=”timeOfDay” type=”numeric” count=”2”>*

*<dimension name=”bodyPosition” type=”categorical”count=”3”/>*

*</dimension>*

*</dimension>*

</*repeatedMeasures* >

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*repeatedMeasures>*

*<dimension name=”time” type=”numeric” count=”5”>*

*[<spacing><v>5</v>….<v>15</v></spacing>]*

*<dimension name=”timeOfDay” type=”numeric” count=”2”>*

*<dimension name=”bodyPosition” type=”categorical”count=”3”/>*

*</dimension>*

*</dimension>*

</*repeatedMeasures* >

##### Read

This command retrieves a ‘Within Subject Effect’ class object’s repeated measures information from study design with the specified UUID.

**URI:** GET /study/study/<UUID>/repeatedMeasures

**XML Request:** none, contained in URI

**XML Reposnse:**

<*repeatedMeasures>*

*<dimension name=”time” type=”numeric” count=”5”>*

*[<spacing><v>5</v>….<v>15</v></spacing>]*

*<dimension name=”timeOfDay” type=”numeric” count=”2”>*

*<dimension name=”bodyPosition” type=”categorical”count=”3”/>*

*</dimension>*

*</dimension>*

</*repeatedMeasures* >

##### Delete

This command deletes a ‘Within Subject Effect’ class object’s repeated measures information from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/repeatedMeasures

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*repeatedMeasures>*

*<dimension name=”time” type=”numeric” count=”5”>*

*[<spacing><v>5</v>….<v>15</v></spacing>]*

*<dimension name=”timeOfDay” type=”numeric” count=”2”>*

*<dimension name=”bodyPosition” type=”categorical”count=”3”/>*

*</dimension>*

*</dimension>*

</*repeatedMeasures* >

### Hypothesis Commands

The Study Design service will store the primary hypothesis selected by the user. The user can select a hypothesis to test a main effect, an interaction, a trend, or a trend by factor interaction.

#### Create/ Update

**Create URI:** POST /study/study/<UUID>/hypothesis

**Update URI:** PUT /study/study/<UUID>/hypothesis

Entity body – the contents of the hypothesis object will vary depending on the type of hypothesis

1. Main Effect Hypothesis –

A single between subject factor or within subject factor should be specified. If multiple factors are specified, the server will select the first and ignore the remaining factors.

**XML Request:**

<*hypothesis type=”main”>*

*<betweenFactorList><v>factor</v></betweenFactorList>*

</*hypothesis*>

OR

<*hypothesis type=”main”>*

*<withinFactorList><v>factor</v></withinFactorList>*

</*hypothesis*>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*hypothesis type=”main”>*

*<betweenFactorList><v>factor</v></betweenFactorList>*

</*hypothesis*>

OR

<*hypothesis type=”main”>*

*<withinFactorList><v>factor</v></withinFactorList>*

</*hypothesis*>

1. Interaction Hypothesis –

Any combination of between and within subject factors may be specified for an interaction hypothesis.

**XML Request:**

<*hypothesis type=”interaction”>*

*<betweenFactorList><v>factor1</v>…<v>factorN</v></</betweenFactorList>*

*<withinFactorList><v>factor1</v>…<v>factorN</v></withinFactorList>*

</*hypothesis*>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*hypothesis type=”interaction”>*

*<betweenFactorList><v>factor1</v>…<v>factorN</v></</betweenFactorList>*

*<withinFactorList><v>factor1</v>…<v>factorN</v></withinFactorList>*

</*hypothesis*>

1. Trend Hypothesis –

Any combination of between and within subject factors may be specified for an interaction hypothesis.

**XML Request:**

<*hypothesis type=”trend”>*

*<trend type=”polynomial | changeFromBaseline | average”>*

*<trendOptionList>*

*<v>linear</v><v>quadratic</v><v>cubic</v>*

*</trendOptionList>*

*<withinFactorList><v>factor</v></withinFactorList>*

*[ OR <betweenFactorList><v>factor</v></betweenFactorList>]*

*</trend>*

</*hypothesis*>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*hypothesis type=”trend”>*

*<trend type=”polynomial | changeFromBaseline | average”>*

*<trendOptionList>*

*<v>linear</v><v>quadratic</v><v>cubic</v>*

*</trendOptionList>*

*<withinFactorList><v>factor</v></withinFactorList>*

*[ OR <betweenFactorList><v>factor</v></betweenFactorList>]*

*</trend>*

</*hypothesis*>

1. TrendInteraction Hypothesis –

The user will specify the type(s) of trend being tested and the factor for which the trend will be tested. In addition, the user will specify an interaction variable to compare trends. The interaction variable must not equal the trend variable.

**XML Request:**

<*hypothesis type=”trendInteraction”>*

*<withinFactorList><v>factor</v></withinFactorList>*

*[ OR <betweenFactorList><v>factor</v></betweenFactorList>]*

*<trend type=”polynomial | changeFromBaseline | average”>*

*<trendOptionList>*

*<v>linear</v><v>quadratic</v><v>cubic</v>*

*</trendOptionList>*

*<withinFactorList><v>factor</v></withinFactorList>*

*[ OR <betweenFactorList><v>factor</v></betweenFactorList>]*

*<interaction>*

*<withinFactorList><v>factor</v></withinFactorList>*

*[ OR <betweenFactorList><v>factor</v></betweenFactorList>]*

*</interaction>*

*</trend>*

</*hypothesis*>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*hypothesis type=”trendInteraction”>*

*<withinFactorList><v>factor</v></withinFactorList>*

*[ OR <betweenFactorList><v>factor</v></betweenFactorList>]*

*<trend type=”polynomial | changeFromBaseline | average”>*

*<trendOptionList>*

*<v>linear</v><v>quadratic</v><v>cubic</v>*

*</trendOptionList>*

*<withinFactorList><v>factor</v></withinFactorList>*

*[ OR <betweenFactorList><v>factor</v></betweenFactorList>]*

*<interaction>*

*<withinFactorList><v>factor</v></withinFactorList>*

*[ OR <betweenFactorList><v>factor</v></betweenFactorList>]*

*</interaction>*

*</trend>*

</*hypothesis*>

#### Read

**URI:** GET /study/study/<UUID>/hypothesis

**XML Request:** none, contained in URI

**XML Response:**

<*hypothesis type=”main | interaction | trend”>*

*…*

</*hypothesis*>

#### Delete

The following command will delete the hypothesis stored with the specified study design

**URI:** DELETE /study/study/<UUID>/ hypothesis

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*hypothesis type=”main | interaction | trend”>*

*…*

</*hypothesis*>

### Covariance Commands

The Study Design service will store covariance matrices and associated meta information for the study design. Users can save variability information either in terms of covariance or correlation. Both input modes are supported in this API.

Covariance API consists of three types of objects;

1. Variability Lists
2. Covariance Objects
3. Correlation Objects

#### Variability Lists

Variability lists are used to enclose lists of covariance and correlation objects. The lists arise for multilevel and longitudinal designs using Kronecker product covariance structures. The variability list uses the following syntax:

*<variabilityList>…</variabilityList>*

The study design describes two possible types of variability:

1. Variability within a single research participant
2. Variability due to a baseline covariate (if present)

The API provides inputs for both types of variability.

##### Within Subject Variability

###### Create/ Update

Entity body – the list of variability components for within subject variability

**Create URI:** POST /study/study/<UUID>/variability/withinSubject

**Update URI:** PUT /study/study/<UUID>/variability/withinSubject

**XML Request:**

*<variabilityList>*

*<covariance>…</covariance>*

*<correlation>…</correlation>*

*…*

*</variabilityList>*

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<variabilityList>*

*<covariance>…</covariance>*

*<correlation>…</correlation>*

*…*

*</variabilityList>*

###### Read

**URI:** GET /study/study/<UUID>/variability/withinSubject

**XML Request:** none, contained in URI

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<variabilityList>*

*<covariance>…</covariance>*

*<correlation>…</correlation>*

*…*

*</variabilityList>*

###### Delete

This command deletes a ‘within subject variability’ from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/variability/withinSubject

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<variabilityList>*

*<covariance>…</covariance>*

*<correlation>…</correlation>*

*…*

*</variabilityList>*

##### Random Covariate Variability

###### Create/ Update

Entity body – the list of variability components for within subject variability. This must include

1. The standard deviation (or variance) of the random covariate represented as a 1x1 matrix
2. The correlation or covariance of the random covariate with each response as a 1xp matrix

**Create URI:** POST /study/study/<UUID>/variability/covariate

**Update URI:** PUT /study/study/<UUID>/variability/ covariate

**XML Request:**

*<variabilityList>*

*<covariance>…</covariance>*

*<correlation>…</correlation>*

*</variabilityList>*

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<variabilityList>*

*<covariance>…</covariance>*

*<correlation>…</correlation>*

*</variabilityList>*

###### Read

Returned XML representation will always include two unstructured covariance or correlation matrices:

1. The standard deviation (or variance) of the random covariate represented as a 1x1 matrix
2. The correlation or covariance of the random covariate with each response as a 1xp matrix

**URI:** GET /study/study/<UUID>/variability/covariate

**XML Request:** none, contained in URI

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<variabilityList>*

*<covariance>…</covariance>*

*<correlation>…</correlation>*

*</variabilityList>*

###### Delete

This command deletes a ‘variability related to baseline covariate’ from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/variability/covariate

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<variabilityList>*

*<covariance>…</covariance>*

*<correlation>…</correlation>*

*</variabilityList>*

#### Covariance Objects

Covariance objects describe variability in terms of covariance. Covariance objects can be structured or unstructured.

1. The syntax for a structured covariance object is:

*<covariance type=”structured” var=”var” maxCov=”maxCov” decaySpeed=”decaySpeed” />*

1. The unstructured covariance matrix describes the full covariance matrix. The syntax for an unstructured covariance object is:

*<covariance type=”unstructured”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</covariance>*

##### Create/ Update

###### Structured covariance object

**Create URI:** POST /study/study/<UUID>/covariance

**Update URI:** PUT /study/study/<UUID>/ covariance

**XML Request:**

*<covariance type=”structured” var=”var” maxCov=”maxCov” decaySpeed=”decaySpeed” />*

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<covariance type=”structured” var=”var” maxCov=”maxCov” decaySpeed=”decaySpeed” />*

###### Unstructured covariance object

**Create URI:** POST /study/study/<UUID>/covariance

**Update URI:** PUT /study/study/<UUID>/ covariance

**XML Request:**

*<covariance type=”unstructured”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</covariance>*

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<covariance type=”unstructured”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</covariance>*

##### Read

###### Structured covariance object

**URI:** GET /study/study/<UUID>/covariance

**XML Request:** none, contained in URI

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<covariance type=”structured” var=”var” maxCov=”maxCov” decaySpeed=”decaySpeed” />*

###### Unstructured covariance object

**URI:** GET /study/study/<UUID>/covariance

**XML Request:** none, contained in URI

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<covariance type=”unstructured”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</covariance>*

##### Delete

###### Structured covariance object

**URI:** DELETE /study/study/<UUID>/covariance

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<covariance type=”structured” var=”var” maxCov=”maxCov” decaySpeed=”decaySpeed” />*

###### Unstructured covariance object

**URI:** DELETE /study/study/<UUID>/covariance

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<covariance type=”unstructured”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</covariance>*

#### Correlation Objects

Correlation objects describe variability in terms of standard deviation and correlation. Correlation objects can be structured or unstructured.

1. The syntax for a structured correlation object is:

*<correlation type=”structured” sd=”sd” maxCorr=”maxCorr” decaySpeed=”decaySpeed” />*

1. The unstructured correlation matrix describes the full covariance matrix. The syntax for an unstructured covariance object is:

*<correlation type=”unstructured” sd=”sd”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</correlation>*

##### Create/ Update

###### Structured covariance object

**Create URI:** POST /study/study/<UUID>/correlation

**Update URI:** PUT /study/study/<UUID>/correlation

**XML Request:**

*<correlation type=”structured” sd=”sd” maxCorr=”maxCorr” decaySpeed=”decaySpeed” />*

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<correlation type=”structured” sd=”sd” maxCorr=”maxCorr” decaySpeed=”decaySpeed” />*

###### Unstructured covariance object

**Create URI:** POST /study/study/<UUID>/correlation

**Update URI:** PUT /study/study/<UUID>/correlation

**XML Request:**

*<correlation type=”unstructured” sd=”sd”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</correlation>*

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<correlation type=”unstructured” sd=”sd”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</correlation>*

##### Read

###### Structured covariance object

**URI:** GET /study/study/<UUID>/correlation

**XML Request:** none, contained in URI

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<correlation type=”structured” sd=”sd” maxCorr=”maxCorr” decaySpeed=”decaySpeed” />*

###### Unstructured covariance object

**URI:** GET /study/study/<UUID>/correlation

**XML Request:** none, contained in URI

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<correlation type=”unstructured” sd=”sd”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</correlation>*

##### Delete

###### Structured covariance object

**URI:** DELETE /study/study/<UUID>/correlation

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<correlation type=”structured” sd=”sd” maxCorr=”maxCorr” decaySpeed=”decaySpeed” />*

###### Unstructured covariance object

**URI:** DELETE /study/study/<UUID>/correlation

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

*<correlation type=”unstructured” sd=”sd”>*

*<matrix name=”name” rows=”m” columns=”n”>…</matrix>*

*</correlation>*

### Power Curve Description Commands

#### Create/ Update

This command adds a ‘power curve’ class object’s description object to the study design with the specified UUID.

**Create URI:** POST /study/study/<UUID>/powerCurve

**Update URI:** PUT /study/study/<UUID>/powerCurve

**XML Request:**

<*powerCurve horizontalAxisLabel=”” verticalAxisLabel=”” sampleSize=n stratificationVariable=a statisticalTest=”unirep” variabilityScaleFactor=.8 typeIError=.4 regressionCoefficientScaleFactor=.5>*

</*powerCurve*>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*powerCurve horizontalAxisLabel=”” verticalAxisLabel=”” sampleSize=n stratificationVariable=a statisticalTest=”unirep” variabilityScaleFactor=.8 typeIError=.4 regressionCoefficientScaleFactor=.5>*

</*powerCurve*>

#### Read

This command retrieves a ‘power curve’ class object’s description object from study design with the specified UUID.

**URI:** GET /study/study/<UUID>/ powerCurve

**XML Request:** none, contained in URI

**XML Reposnse:**

<*powerCurve horizontalAxisLabel=”” verticalAxisLabel=”” sampleSize=n stratificationVariable=a statisticalTest=”unirep” variabilityScaleFactor=.8 typeIError=.4 regressionCoefficientScaleFactor=.5>*

</*powerCurve*>

#### Delete

This command deletes a ‘power curve’ class object’s description object from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/powerCurve

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*powerCurve horizontalAxisLabel=”” verticalAxisLabel=”” sampleSize=n stratificationVariable=a statisticalTest=”unirep” variabilityScaleFactor=.8 typeIError=.4 regressionCoefficientScaleFactor=.5>*

</*powerCurve*>

### Confidence Interval Commands

This command adds a ‘Confidence interval’ class object’s description object to the study design with the specified UUID.

**Create URI:** POST /study/study/<UUID>/confidenceInterval

**Update URI:** PUT /study/study/<UUID>/confidenceInterval

**XML Request:**

<*confedenceInterval isBetaMatrixFixed=’true’ isSigmaMatrixFixed=’true’ lowerTailProbability=.5 upperTailProbability=.6 sampleSize=100 rankOfDesignMatrix=2>*

</*confedenceInterval*>

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*confedenceInterval isBetaMatrixFixed=’true’ isSigmaMatrixFixed=’true’ lowerTailProbability=.5 upperTailProbability=.6 sampleSize=100 rankOfDesignMatrix=2>*

</*confedenceInterval*>

#### Read

This command retrieves a ‘power curve’ class object’s description object from study design with the specified UUID.

**URI:** GET /study/study/<UUID>/confidenceInterval

**XML Request:** none, contained in URI

**XML Reposnse:**

<*confedenceInterval isBetaMatrixFixed=’true’ isSigmaMatrixFixed=’true’ lowerTailProbability=.5 upperTailProbability=.6 sampleSize=100 rankOfDesignMatrix=2>*

</*confedenceInterval*>

#### Delete

This command deletes a ‘power curve’ class object’s description object from the existing study design with the specified UUID.

**URI:** DELETE /study/study/<UUID>/confidenceInterval

**XML Request:** no entity body

**HTTP Response:** 200 STATUS\_OK or 400 BAD\_REQUEST

**XML Response:**

<*confedenceInterval isBetaMatrixFixed=’true’ isSigmaMatrixFixed=’true’ lowerTailProbability=.5 upperTailProbability=.6 sampleSize=100 rankOfDesignMatrix=2>*

</*confedenceInterval*>

## External API

The API described below allows users to create, read, update and delete study designs and associated components of study designs. As a general rule, the following HTTP protocols are used for each operation

|  |  |
| --- | --- |
| **HTTP Request Type** | **Operation** |
| GET | Read |
| POST | Create |
| PUT | Create or Update |
| DELETE | Delete |

### Operations on Study Design Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/verbose | **-** | Study Design object | By this command, user may request the full study design including both matrices and meta information by providing a UUID. |
| GET | /study/study/<UUID>/ | **-** | Study Design object | By this command, user may request the matrix representation only for Study Design by providing a UUID. |
| POST | /study/study/ | Study Design object | Study Design object | This command creates a study design. This created study design is returned back to user. |
| PUT | /study/study/<UUID>/ | Study Design object | Study Design object | This command updates an existing study design specified by provided study UUID. This updated study design is returned back to user. |
| DELETE | /study/study/<UUID>/ | **-** | Study Design object | This command deletes a study design with the specified study UUID |
| DELETE | /study/study/date/<iso-date> | **-** | Study Design object | This command deletes all study designs prior to a specified date. |

### Operations on Confidence Interval Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/confidenceInterval | **-** | Confidence Interval object | This command retrieves a confidence interval object related to the study design specified by a UUID. |
| POST | /study/study/<UUID>/confidenceInterval | Confidence Interval object | Confidence Interval object | This command creates a confidence interval object for the specified study design. Created confidence Interval object is returned back to the user. |
| PUT | /study/study/<UUID>/confidenceInterval | Confidence Interval object | Confidence Interval object | This command updates a confidence interval object of the specified study design. Updated confidence Interval object is returned back to the user. |
| DELETE | /study/study/<UUID>/confidenceInterval | **-** | Confidence Interval object | This command deletes a confidence interval object for specified study design. |

### Operations on Power Curve Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/powerCurve | **-** | Power Curve object | This command retrieves a power curve object related to the study design specified by a UUID. |
| POST | /study/study/<UUID>/powerCurve | Power Curve object | Power Curve object | This command creates a power curve object for the specified study design. Created power curve object is returned back to the user. |
| PUT | /study/study/<UUID>/powerCurve | Power Curve object | Power Curve object | This command updates a power curve object of the specified study design. Updated power curve object is returned back to the user. |
| DELETE | /study/study/<UUID>/powerCurve | **-** | Power Curve object | This command deletes the power curve object for specified study design. |

### Operations on List Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/list/<name> | **-** | List object | This command retrieves the list with the specified name with the given UUID. |
| GET | /study/study/<UUID>/list | **-** | List object | This command retrieves all the lists of specified study design. |
| POST | /study/study/<UUID>/list | List object | List object | This command creates a list object for the specified study design. Created list object is returned back to the user. |
| PUT | /study/study/<UUID>/list/<name> | List object | List object | This command updates a list object of the specified study design. Updated list object is returned back to the user. |
| DELETE | /study/study/<UUID>/list/<name> | **-** | List object | This command deletes the list object with provided name for specified study design. |

### Operations on Matrix Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/matrix/<name> | **-** | Matrix object | This command retrieves the matrix with the specified name with the given UUID. |
| GET | /study/study/<UUID>/matrix | **-** | Matrix object | This command retrieves all the matrices of specified study design. |
| POST | /study/study/<UUID>/matrix | Matrix object | Matrix object | This command creates a matrix object for the specified study design. Created matrix object is returned back to the user. |
| PUT | /study/study/<UUID>/matrix/<name> | Matrix object | Matrix object | This command updates a matrix object of the specified study design. Updated matrix object is returned back to the user. |
| DELETE | /study/study/<UUID>/matrix/<name> | **-** | Matrix object | This command deletes the matrix object with provided name for specified study design. |

### Operations on Within Subject Effect Object

#### Response List Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/responseList | **-** | Response List object | This command retrieves the response list of the specified study design. |
| POST | /study/study/<UUID>/responseList | Response List object | Response List object | This command creates a response list object for the specified study design. Created response list object is returned back to the user. |
| PUT | /study/study/<UUID>/responseList | Response List object | Response List object | This command updates a response list object of the specified study design. Updated response list object is returned back to the user. |
| DELETE | /study/study/<UUID>/responseList | **-** | Response List object | This command deletes the response list object of specified study design. |

#### Clustering

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/clustering | **-** | Clustering object | This command retrieves the clustering object of the specified study design. |
| POST | /study/study/<UUID>/clustering | Clustering object | Clustering object | This command creates a clustering object for the specified study design. Created clustering object is returned back to the user. |
| PUT | /study/study/<UUID>/clustering | Clustering object | Clustering object | This command updates a clustering object of the specified study design. Updated clustering object is returned back to the user. |
| DELETE | /study/study/<UUID>/clustering | **-** | Clustering object | This command deletes the clustering object of specified study design. |

#### Repeated Measures

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/repeatedMeasures | **-** | Repeated Measures object | This command retrieves the repeated measures object of the specified study design. |
| POST | /study/study/<UUID>/repeatedMeasures | Repeated Measures object | Repeated Measures object | This command creates a repeated measures object for the specified study design. Created repeated measures object is returned back to the user. |
| PUT | /study/study/<UUID>/repeatedMeasures | Repeated Measures object | Repeated Measures object | This command updates a repeated measures object of the specified study design. Updated repeated measures object is returned back to the user. |
| DELETE | /study/study/<UUID>/repeatedMeasures | **-** | Repeated Measures object | This command deletes the repeated measures object of specified study design. |

### Operations on Between Subject Effect Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | URI | Input | Output | Description |
| GET | /study/study/<UUID>/predictorList | **-** | Predictor List object | This command retrieves the predictor list object of the specified study design. |
| POST | /study/study/<UUID>/predictorList | Predictor List object | Predictor List object | This command creates a predictor list object for the specified study design. Created predictor list object is returned back to the user. |
| PUT | /study/study/<UUID>/predictorList | Predictor List object | Predictor List object | This command updates a predictor list object of the specified study design. Updated predictor list object is returned back to the user. |
| DELETE | /study/study/<UUID>/predictorList | **-** | Predictor List object | This command deletes the predictor list object of specified study design. |

### Operations on Hypothesis Object

### Operations on Covariance Object

## Automated Database Cleanup

Study Designs older than one month will be removed from the database. A crontab entry will be added which will issue a ‘curl’ command to the Study Design service. The curl command will use the /study/study/date/<iso-date> API.

# Reuse and Relationships to Other Products

## Reuse of existing code

* The Study Design Service code will follow the Restlet format used for existing web services, but reuse of code will be minimal.
* Unit testing artifacts and test cases will reuse some of the existing documentation and described test cases.

## Third Party Dependencies

* MySQL Database Server – 5.5

MySQL Administrator version – 1.2.17

MySQL Work Bench version – 5.2.35

* Tomcat 6.0 or higher.
* Hibernate 4.0.0
* Restlet 2.x

# Design Decisions and Tradeoffs

The proposed architecture is not yet fully refined. The following represent open design questions.

1. Synchronization between the Study Design Service database and user-downloaded copies of the study design

Suppose that a user has created a study design which has been stored in the Study Design Service. The user then downloads a local copy of the study design as an xml file. Suppose the user modifies the study design through the user interface but does not save a new copy of the study design. Now the user wants to upload this data. The UUID of the study exists in the Study Design Service, but it out of sync with the uploaded copy. The application can either

* 1. Inform the user of the conflict through an alert box stating ‘This study already exists in table. Do you want to overwrite it?’ The user can indicate whether to overwrite the existing design.
  2. Always overwrite the existing design in the Study Design Service.

1. Doubt in Hypothesis object : ENUM('Main Effects','Interaction','Linear Trend','Quad Trend','Cubic Trend')

Possible solution for this can be use of passwords. But then some more issues come in picture e.g.

1. What is the secure method for storing Passwords in MySQL tables?
2. User has to remember different passwords for each of the study design.
3. And what if user doesn’t remember Password? How can he recover the Password?

# Appendices

## Diagrams

* ***Existing Structure***

TOMCAT

Client browser

Matrix Service

Apache Httpd

Power Service

Chart Service

File Service

* ***Proposed Version 2.0.0 Structure***

TOMCAT

Httpd

Client browser

Study Design Service

Matrix Service

Power Service

Databases

MYSQL

HIBERNATE

File Service