# PIRSFLoad Design Document

Author: M Walker

Created: July 15, 2005

Last Modified: October 28, 2005 10:08

## 1 Purpose of Document

This document will describe the design of the PIRSF load for the purpose of providing documentation to the product for cross team maintenance and usage. This load was developed to meet the requirements established in reference 1 and to implement the Data Load Architecture guidelines set forth in the DLA Standards document (reference 2). It is assumed the reader is familiar with these documents. The reader should also be familiar with the vocload and annotload documentation references 3 and 4. The vocload and annotload are responsible for write to the VOC_Term and VOC_Annot tables which are described in reference 5.

## 2 Introduction

## 3 Load Overview

The following subsections define each of the steps taken to complete the PIRSFLoad.

### 3.1 Obtaining and Parsing the Input Data from PIRSF

The iproclass file from PIRSF is specified in reference 1. This load utilizes the mirrorftp product to obtain the data from PIRSF. This is a compressed file larger than 7 gigabytes in size. At runtime it is gunzipped onto an output stream and subsequently piped into the standard input for this load.

The iproclass file is described in reference 1. The fields parsed are listed in table 1

**Table 1: data parsed from the iproclass input file**

| iproclass field name | definition |
|---|---|
| sprot | swissprot sequence associations |
| trembl | trembl associations |
| locusid | entrez gene id |
| pirsupfam | superfamily data which includes name and superfamily id |

**Table 1: data parsed from the iproclass input file**

| iproclass field name | definition |
|---|---|
| refseq-ac | refseq sequence associations |
| locus-id | Entrez Gene id |
| source-org | source organism which must be equal to mus musculus |

## 3.2 Mapping PIRSF Superfamilies to MGI Markers

The mapping between PIRSF superfamilies and MGI markers is acheived by parsing out the sequences associated to the PIRSF superfamily (swissprot, trembl, refseq and entrez gene) and looking up markers in MGI that are associated to these sequences. If a marker maps to more than one superfamily, these superfamilies are reported, but are not included in the mapping. Some superfamilies are named "not yet assigned" and these are recognized by the algorithm and not loaded.

## 3.3 Create Vocabulary Records in MGI for PIRSF Superfaimilies

The VocLoad is used to load the superfamily terms into MGI. This load must create a file for input as specified in reference 3. For each superfamily found to map to at least one MGI marker a VOC_Term database record is created to store the term along with an ACC_Accession record which assigns the superfamily id to the term. See reference 5 for information on these tables.

## 3.4 Create Annotation Records in MGI for Marker and Superfamily

The annotload is used to load superfamily associations to markers. This load creates a file for input into the annotload as specified in reference 4. For each marker associated to just one superfamily, a VOC_Annot record is created which ties the Voc_Term record from section 2.3 to the marker represented in the MRK_Marker table. See reference 5 for information on these tables.

## 3.5 Create QCReports

The following lists the qc reports created by this load.

Superfamilies which include mouse sequence associations but are not associated to any MGI Marker.

MGI markers which map to more than one superfamily. These annotations are not loaded into the database.

Superfamilies loaded into the database with duplicate term names.

Annotations created in the database to MGI markers of types other than gene.

# 4 Class Overview

The **PIRSFLoad** is an extension of **DLALoader** from the lib_java_dla product (see org.jax.mgi.shr.dla.loader). It is responsible for coordinating the input data, lookup data and running the mapping algorithm. It has a **PIRSFInputFile** instance for reading PIRSF data, a **EntrezGeneLookup** and a **ProteinSeqLookup** for looking up MGI markers associated to protein sequences and Entrez Genes. And for processing the data into the database, it has an instance of the **VocLoader** and the **Annot-Loader**.

The **PIRSFInputFile** provides a iterator for iterating over **PIRSFSuperfamily** Objects based on the iproclass xml input file from PIRSF.

The **PIRSFSuperfamily** represents a PIRSF superfamily and contains superfamily name, id and sequence associations to swissprot, tremble, refseq and Entrez Gene.

The **ProteinSeqLookup** provides a way to lookup MGI Markers associated to a given protein sequence. This is a extension of the FullCachedLookup class from the lib_java_dbsmgd product.

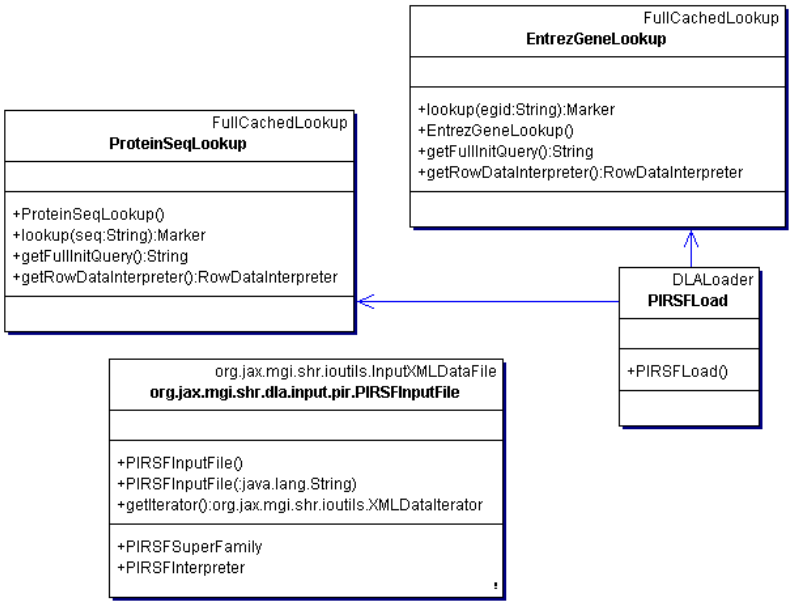
The **EntrezGeneLookup** provides a way to lookup MGI Markers associated to Entrez Genes. This is a extension of the FullCachedLookup class from the lib_java_dbsmgd product.

The **VocLoader** is a wrapper class around the vocload python application.

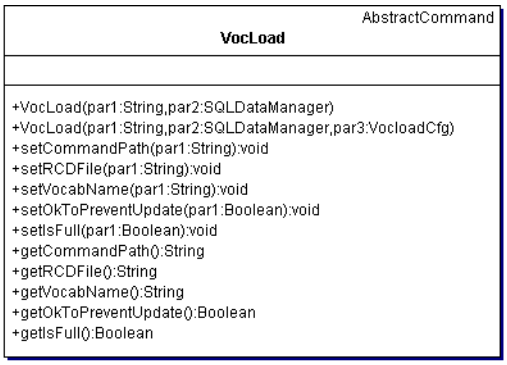
The **AnnotLoad** is a wrapper class around the annotload python application.

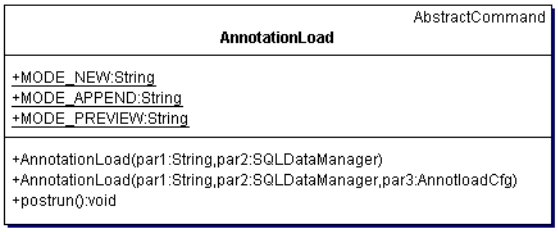
# 5 Class Diagrams

## 5.1 Loader and Components

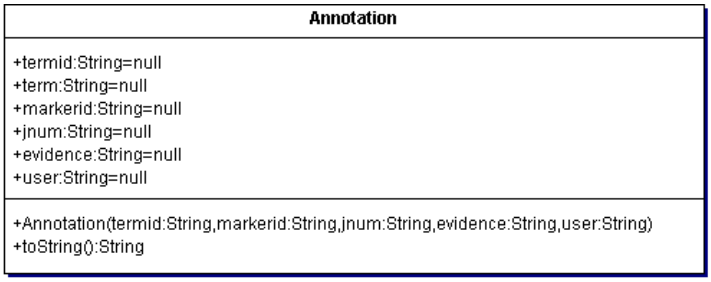


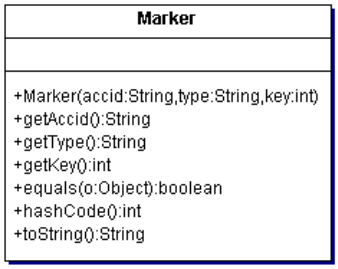
## 5.2 External Command Classes

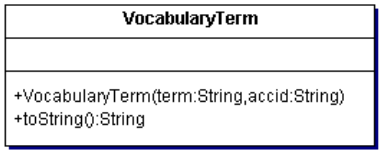



## 5.3 Java Bean Objects



# 6 References

PIRSF user requirements doc

DLAStandards

Vocload

Annotload

Schema Product