1 Introduction

This document describes the use cases and API's for the program Mosflm. Mosflm is an unusual case, because there are number of significant functions it performs - each of these will have to be a well identified use case.

1.1 Functionality

The program mosflm has the following key pieces of functionality which need to be exposed *via* the API:

- Autoindexing of diffraction patterns.
- Estimation of mosaic spread.
- Refinement of unit cell.
- Integration of diffraction patterns.
- Calculation of data collection strategies (deprecated.)

1.2 Input

Each of these tasks requires a certain amount of common input - it can only make sense to standardise on the API for this. Of particular importance are the content of the image header and the frames to use in processing.

2 Rules

2.1 Introduction

There are a couple of rules which are useful when running mosflm, and these should be encoded in the *wrapper* rather than the rest of the expert system because they are specific to this program.

These rules pertain to:

- Indexing errors, in particular lattice assignment.

2.2 Indexing Errors & Lattice Assignment

Prototype data set: 1VR9 native (12847) from the JCSG. This is correctly C2 symmetry, but is pseudo-I222.

Autoindexing with mosflm, even with the "refine" keyword assigned, gives a lattice of I222. However, when this is set the mosaic spread estimation & refinement fails, which is a pointer that there is something wrong. The estimation fails saying "you'll have to guess" or something, and is then refined to a negative value in cell refinement. This should be interpreted as a pointer that the lattice has too high a symmetry. Asserting a spacegroup of C2 corrects this, so clearly this is something important to build into the wrapper.

3 Making Sure Nothing is Lost

FIXED 16/AUG/06 the distortion & raster parameters decided on in the cell refinement stages need to be recycled to use in integration. This is demonstrated by running an interactive (through the GUI) mosflm autoindex and refine job, then dumping the runit script. The important information is in the following records:

```
 Final optimised raster parameters:   15   17   12    5    6
   => RASTER keyword
 Separation parameters updated to   0.71mm in X and  0.71mm in Y
   => SEPARATION keyword
   XCEN    YCEN  XTOFRA   XTOFD  YSCALE  TILT TWIST
 108.97  105.31  0.9980  149.71  0.9984   -13   -46
   => BEAM, DISTANCE, DISTORTION keywords (note that the numbers
      are on the next line here)
```

This should make the resulting integration more effective. The idea for this implementation is that the numbers end up in the "integrate set parameter" dictionary and are therefore recycled, in the same way that the GAIN currently works. Note well - some of this conflits with the definition of the FrameProcessor, in which case the relevant information should be passed back to that interface.

Have implemented an application xia2process, which should test out this wrapper. However, there seem to be some issues with the implementation when running with scripts, or Labelit as the indexer implementation, which I will need to fix.