API/metadata proposal

The actual modeling work lies in making the LIMS API, which also serves as metadata to store in LIMS systems. There is a lot to do, depending on how wide areas we want to cover, but here is a starting point. The generic LIMS skeleton that this is supposed to fit into is described elsewhere. It is important to note that the data schemas described here are not limited to fitting into a LIMS. They contain data, not linked objects, and they can be used anywhere convenient. For instance the schema describing an MX data acquisition dataset (a Sweep) can also be used in a diffraction plan, or as input in an experiment queue.

I have tried to build on existing modeling (ISPyB, the current MXCuBE LIMS API, the Sample handling model proposed by Ed Daniel and Karl Levik, ...), and use cases from e.g. the Global Phasing workflow experiments. I have also tried to avoid duplicate information storage (which leads to inconsistent data and unpredictable storage patterns), to make the models generic where this could be done without too much complexity, and felt free to propose tag names that fit better than the historic ones, or that simply makes the draft easier to understand. The current draft is in the attached spreadsheet (to allow editing), with relevant bits shown below. Once we have more agreement on the contents we can start tightening up the definitions and coding this in JSON schemas.

Note that the links given in the generic model are not repeated here. Most importantly this means that the link from the Job class to Sample is not mentioned – but it is there.

MX acquisition

So far the classes CrystallographyExperiment, Sweep, Scan, and UnitCell are worked in detail, and there is a draft for SampleData; there will be classes for other experiment types and other Dataset types.

CrystallographyExperiment

This class is a type of Job; the schema contains information and results that applies to the experiment as a whole, regardless of the number of sweeps. The ExperimentPlan could contain either generic Sweep records (such as ‘Characterisation’ and ‘Acquisition’) or a complete list of Dataset metadata records, such as could be submitted to an acquisition queue.

The target_resolution is the resolution that the user expects and wants to measure to, and does not have a direct relation to the detector distance. It would be preferable not to have to distinguish target and required resolution etc.

Input:		
sub_sample_id	Sample	identifier of SubSample (Crystal) within Sample
experiment_kind	str	Native, MAD, SAD, ...
experiment_strategy	str	MXPressE, GPhL_quick, ...
target_resolution	float	NB distinction between aimed and required resolution etc. ??
target_completeness	float	
target_multiplicity	float	
dose_budget	float	
characterisation_strategy	str	
recentring_strategy	str	
snapshot_count		
wedge_width	float	Input for interleaving strategy, actual scans are given explicitly in the sweep
experiment_plan	list	List of Dataset parameters (Scan, Centring, ..., complete or partial) describing experiment queue
Output:		
measured_flux	float	
radiation_dose	float	Total radiation dose over experiment
unit_cell	UnitCell	As determined during characterisation

Sweep

The Sweep is a Dataset. The record is based on MX, but should work also for multi-crystal, and even SSX or XFEL. In the latter two cases a lot of additional information would be needed. The file specification has barely been sketched in, and would need additional detail even for MX.

Note that the Sweep specifies a single, continuous sweep range, with equidistant images given by image_width, and all motor positions in axis_position_start. Axis_positions_end contain the end point of the sweep, and must have at least the value for the scan_axis; sweeps changing more than one motor (e.g. helical scan) can be represented by adding more values to axis_positions_end. The default number of images can be calculated from the sweep range and image_width. The actual number of images, the image numbering, and the order of acquisition (including interleaving) follows from the list of Scans.

General:		
role	Enum	E.g ‘Characterisation’, ‘Acquisition’
sub_sample_id	int	Identifier of actual crystal used in acquisition – if not given in the CrystallographyExperiment
annotation	str	
Acquisition:		
exposure_time	cloat	
image_width	float	width of scan.
energy	float	
transmission	%	

Detector:		
detector_distance	float	NB, we do NOT store resolution. The detector distance is clearly defined, the resolution is not.
detector_binning_mode	Enum	
detector_roi_mode	Enum	
beam_position	(float, float)	Position on detector, determines detector offset.
beam_size	(float, float)	
beam_shape	Enum	
Scan and position:		
axis_positions_start	dict	Starting position of all axes, rotations or translations, including detector distance, by name.
scan_axis	Enum	Name of main scanned axis
overlap	float	Overlap between successive images. May be negative if images are not contiguous. Is this still necessary??
axis_positions_end	dict	Positions at end of sweep for all axes that are scanned. NB scans may be acquired out of order, so this determines the limits of the sweep, not the temporal start and end points.
num_triggers	int	Instructions for how to set up detector and acquisition – do not modify meaning of other parameters.
num_images_per_trigger	int	*Ibid.*
scans	list[Scan]	Subdivisions of sweep – NB need not be contiguous, or in order, or add up to entire sweep.
Files:		Needs expansion for HDF5 and MX, and even more for e.g. SSX
prefix	str	
run_number	int	
path	str	

Scan

Part of a sweep, acquired contiguously. The ordinal gives the acquisition order across the entire experiment, not just the sweep, and so can be used to specify interleaving, incompletely acquired sweeps, etc.

Scan		Subdivision of sweep
scan_position_start	float	Value of scan_axis for first image
first_image_no	int	Image number to use for first image
num_images	int	
ordinal	int	acquisition order within experiment (not just within sweep)

Diffraction Plan

There is no separate schema(s) for a diffraction plan. Instead you simply make a CrystallographyExperiment record with attached Sweep records, which can be attached to the Sample specification as desired. The records for the diffraction plan can be filled in only partially, letting the beamline fill the rest from default settings.

Sample specification

The connections of crystals to containers and the links to Jobs and Datasets are described under the generic model. What remains is to specify the metadata. The different kinds of SampleHolder (used for Sample tracking) will need metadata to specify layout of the various containers (e.g. organisation of plates) and which other containers they can hold; Samples may need dat to describe atached images, and SubSamples will need metadata to connect them to holder positions and to particular sections of attached images.

The SampleData schema describes the actual contents of a Sample. For now the data in this draft are limited to the data from the Crystal and ProteinConstruct classes from the Icebear sample shipping model v0.3.0 (https://icebear.fi/shiplink/schemadoc/?schema=https://icebear.fi/shiplink/v0_3_0/schema.json), with some minor additions. One might, if desired, give a more detailed sample description, including concentration of the various components, cryoprotection, etc.

The SampleData schema describes the data that would be attached to the Sample object in the model skeleton, but they apply more widely. Processing Jobs and Datasets generally do not have a direct relation to a single Sample – but still need many of the same kinds of information – so SampleData records (fully or partially filled in) could be atatched to other schemas to carry this information.

SampleData

The contents are taken from the Icebear data model (see above). The tag names are changed, to avoid using ‘Protein’ and ‘ligand’ when this is not an accurate description of the stored data. The radiation sensitivity is added as a necessary parameter for strategy calculation that belongs in the sample description rather than the diffraction plan. As modeled here you can have a list of unit cells, to cater for situations where a given Sample might contain one of several crystal forms. If we do not want to keep this feature, we could move back to having only a single unit cell in the data.

SampleData		
unit_cells	List[UnitCell]	List of possible unit cells
radiation_sensitivity	float	Radiation sensitivity relative to standard crystal
molecule_name	str	Human-readable name of main molecule/complex (was ‘protein’)
molecule_acronym	str	Short code used to identify molecule for safety purposes
construct_name	str	Human-readable name of molecular construct
sequences	List[str]	The DNA or protein sequences of the construct.
components	List[str]	other components used in the Sample liquid (e.g. binders, substrates, metal ions, additives, ...)

UnitCell

The UnitCell is a standard data structure. We use space group names instead of space group numbers, to allow users to specify non-standard settings. The crystal_classes is a list of arithmetic crystal class names (see https://onlinelibrary.wiley.com/iucr/itc/Cb/ch1o4v0001/), that describe essentially the combination of point group and Bravais lattice, disregarding the location of screw axes or glide planes. The list of crystal class names serves to specify partial or ambiguous symmetry information.

UnitCell		
a	float	
b	float	
c	float	
alpha	float	
beta	float	
gamma	float	
space_group_name	Enum	Space group name (unlike the SG number, this supports non-standard settings)
crystal_classes	List[str]	Arithmetic crystal class names; to specify partial information or ambiguous symmetry