GENERAL RECOMMENDATIONS FOR WORK ON PE

1. **Functional naming.** Give names to variables and functions that make their functions as obvious as possible. Don't use cute, personal, or trivial names that are meaningless to others. Be specific enough to distinguish the function from related functions. In this spirit, I renamed (in shared/header.js) return_header(recordname) to get_records(recordname).

2. **Unique names.** When creating new variable or function names, use "meg" (megrep) to be sure you have picked a unique name. Using unique names makes it much easier to find all relevant portions of code later. For example, don't name code that deals with the header TITLE record "title" because there are already hundreds of irrelevant occurrences of "title" in protexpl and shared. "htitle" (Header TITLE) is unique (used in top.pdbheader[foi]["htitle"]).

3. **Avoid very short names (not unique).** Any variable that is used in more than a few lines should be given a multiple-character name that is easy to find. Too often I have not followed this recommendation myself -- nevertheless, I don't recommend local variables named "i", "j", "n", "x", and so forth. They are difficult to find and maintain when their use extends beyond a few lines. Use names such as "i1", "i2", or better, related to function such as "ikey", "jnewline", "xloc". Example: JPD used "fo" extensively. This caused a big maintenance problem. Eric has tried to find all of these and rename them to "foi" (unique). Although the reason for the name "fo" eludes Eric and Diana, Eric kept the "fo" in "foi" because we had worked with it for months and so became used to it. Changing it seemed more likely to confuse us.

4. **Comment, comment, comment.** Insert comments often to explain what code is doing and how it works. Do this in code you inherited as you figure it out, and in new code that you write. We all do this too little!

5. **PDB Format Specification.** The definitive PDB header format specification document is found by going to [www.pdb.org](http://www.pdb.org), clicking on FILE FORMATS, then looking for the current PDB format. Currently, it is at [http://www.rcsb.org/pdb/does/format/pdbguide2.2/guide2.2_frame.html](http://www.rcsb.org/pdb/does/format/pdbguide2.2/guide2.2_frame.html)

6. **Name according to PDB format specification.** In code and comments, refer to header records, fields, elements, etc. with the names used in the PDB format specification. Try not to make up your own names, and try to avoid using different names for the same thing in different places. For example, there is a date in the HEADER record. The PDB format specification calls this the "deposition date", so that is what we should call it. (There is also a "release date" when the PDB file is made public, but it is not in the HEADER record. And there are other dates we'll encounter.) This date was stored in top.pdbheader[foi]["entrydate"], and there was a function define_entrydate(), etc. Hmmm, is the entry date different than the
deposition date? No, so lets use "deptime" instead. Luckily, "entrytime" was unique so it was easy to find and rename.

7. **Locking files you work on.** As a team, we need to be careful to avoid independently modifying the same files, so we don't have to spend time later integrating different versions of the same files. So we need to communicate clearly which files each of us is working on, and once it is announced that certain files are being worked on by one of us, the others must consider those files "locked" until further notice. If you find a problem that needs fixing in a locked file, send us an email describing the problem. In general, Eric has permission to modify all files except those explicitly locked by Diana or Rosalia. Rosalia has permission to modify only: hcquery.js, header.js, features.htm, and featwin.htm. These are locked to Diana and Eric.

**OVERVIEW OF HOW FEATURES WORKS**

1. **hcquery.js** (in shared; Header Chime QUERY) fetches the header, separates the original header from the PIPE block (if present), cleans the header, and digests it. ("PIPE block" refers to header lines that specify the content for a Presentation In Protein Explorer (PIPE). PIPE support is in an early stage of development.)

2. **header.js** (in shared) defines arrays and functions that extract the desired records from the header.

3. **features.htm** (in protexpl) creates the Features control page in PE.

4. **featwin.htm** (in protexpl) is the document in a subwindow that generates displays of multiple-line header records.

5. **featdoc.htm** (in protexpl) is where documentation for each feature record type will be built. It is partially generated automatically from the header_records[] arrays. Details for each feature are supplied by **featdoc.js**.

**TOP PRIORITY FOR NEW RECORDS IN FEATURES (NOT IN ORDER)**

- “Replaced by” and “date replaced” can be found under “OBSLTE” record name, when applicable. *
- **DONE:** “Experiment” can be found under record name: “TITLE”
- **DONE:** “Molecule(s)” can be found under “COMPND”
- “Source, Species etc.” can be found under “SOURCE”
- “Citation” can be found under “JRNL”
- “Date revised” can be found under “REVDAT” and “Revision” can be found under “REMARK 860” (1AX8 is an example). *
- “Obsolete files” can be found under “SPRSDE” *

* These are a special category. When not found, they should not appear on the Features page (unlike e.g. "Resolution: not found."). They will be listed in featdoc.htm in the link at the bottom of the text features list Also not found.
PRIORITIES ADDED BY ERIC June 9, 2003

• AUTHOR should definitely be done!
• Support for multiple-line REMARK 2 records.
• REMARK 900 related entries. Example is 1H6M.

The remainder are low or uncertain priority.
It will help when we locate examples with some of the more obscure record types. That is why I've been bugging Jaime Prilusky to make a record name search slot in OCA. Until we can find how often these are used and look at some examples, it is hard to tell whether some of these are useful.

• REMARK 4? PDB format version.
• REMARK 5? (supposedly redundant with CAVEAT)
• KEYWDS ?
• HYDBOND?
• SALTBRG?
• REMARK 400? "further details on the compound"
• REMARK 550? SEGID
• REMARK 600?
• REMARK 850, 860? Corrections.
• REMARK 102, REMARK 103? Non-standard hbonds between nucleic acid bases
• CAVEAT?

At this time I am not ready to contemplate sequence-related records. Dealing with them will be very complicated. However, I hope that someday we can display a better Sequences page, and show discrepancies between the sequence of the coordinates and the SEQRES records.

ORIENTATION FOR ROSALIA

Don’t worry too much about order/appearance at the outset. Your initial goal should be to just to get the desired records displayed on the Features page. Later we can refine the order and formatting.

On the other hand, once you get a record displayed, feel free to think about what is the best way to format it, and go ahead and do that.

Make sure to read comments above list_text_feature() function in Features.htm
CHANGES ERIC MADE June 8, 2003 TO ADDRESS PROBLEMATIC CODE

DESIGN

Note: Eric didn't get around to debugging the extensive changes he made for IE. The current version works only in Netscape 4.8.

New array: header_records[ i][ ]

It would be helpful to have an array containing all the header record names we are using in Features. Associated with each record name can be other information about the record, such as the description for the record displayed on the Features page (for text records), etc. This array should not contain any molecule-specific information.

Eric created such an array named header_records[], in shared/header.js.

Immediate uses for header_records[] are:

1. It avoids naming the record, record description, etc. redundantly in multiple locations (hard to maintain).
2. It can be used to automate digestion of all records (formerly done by a manually constructed list of calls to define_xxx() functions).
3. It can be used to automate the generation of a document that lists all the records in Features. This will make it easy to find, for example, that information described as "Classification" is obtained from the record named "HEADER".

JPD and Diana made a series of functions define_xxx() that extracted each record from the header. These functions called return_header(), which scanned through the entire header for each record. Thus, as we add more and more records to Features, we end up scanning the entire header over 20 times. Javascript, being interpreted, is inefficient. I was concerned that this was taking too much time, and considered redesigning it to divide the header into blocks for each record name in a single pass. However, timing tests (see results in table below) showed that the bottleneck is passing the header from Chime to javascript, not processing the header once received. Therefore no improvements to the efficiency of the code were made.

<table>
<thead>
<tr>
<th>PDB Code</th>
<th>Header length</th>
<th>Msec to get header from Chime (Netscape)</th>
<th>Msec to clean and digest header</th>
</tr>
</thead>
<tbody>
<tr>
<td>1d66</td>
<td>7</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>1b07</td>
<td>25</td>
<td>60</td>
<td>110</td>
</tr>
<tr>
<td>2svw*</td>
<td>41</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>1gix</td>
<td>91</td>
<td>490</td>
<td>170</td>
</tr>
<tr>
<td>1aan</td>
<td>144</td>
<td>1,370</td>
<td>110</td>
</tr>
<tr>
<td>1ffk</td>
<td>210</td>
<td>2,800</td>
<td>270, 160</td>
</tr>
<tr>
<td>1g3i</td>
<td>275</td>
<td>5,770</td>
<td>170</td>
</tr>
<tr>
<td>1jj2</td>
<td>290</td>
<td>4,840, 5,380</td>
<td>1,590, 770</td>
</tr>
</tbody>
</table>

Multiple values are replicate observations.
The emphasis in Features should be on the specifics for the PDB file loaded, not on the record types displayed. Therefore I unbolded the record types (descriptions) and bolded the specific data.

Define_resolution( ) was converting the number with parseFloat( ). Not only is there no reason to convert the resolution to a floating point value, but this caused everything after the decimal point to be lost, including critically the word "ANGSTROMS". Diana and I decided not to add that automatically in cases someone ever gives resolution in a different unit.

CHANGES ERIC WANTS ROSALIA TO DO STARTING June 9, 2003

In header.js, expand the header_records[i][...] array to include ALL record names we currently support.

In header.js, go through each define_xxx( ) function making the following changes:
1. Eliminate nonsense indexOf(record_name) code. See old and new versions of define_classification for example.
2. Rename functions to replace xxx with the reccname in header_records[]. For example, define_classification() becomes define_header(). Already done: define_header() divides the HEADER into classification and deposition date fields, thereby eliminating the former define_entrydate().
3. Recode as needed to employ the new header_records[i][] array, using define_compnd() and define_title() as examples.

In hcquery.js:
4. Replace the list of calls to define_xxx( ) with the loop I have already put there. Thus, when a new record is added to header_records[], and after a new define_recordname() is created, it gets called automatically.

In features.htm: Change the old style list_text_feature( ) to the new style list_text_feature2( ) for all records that don't have sub-fields. (See comments for classification and deposition date, still using list_text_feature( ).)

TITLE: Notice that in multiple line titles, the line numbers are inappropriately displayed. Adapt the code in format_hcompnd() (in features.htm) to strip line numbers out of multiple-line records. Best to make a function that can be called to do this for use in future records with multiple lines, rather than duplicating the same code in multiple places (making it difficult to maintain).
COMPND: Presently, there is always a "View" link regardless of the length of the COMPND records. What I suggest is that when COMPND contains no "MOL_ID", and when it is less than 150 (?) characters in length, it be displayed just like TITLE, directly on the Features control panel. When it needs to be displayed in a window, I suggest that you extract the MOLECULE: lines and concatenate them delimited by "; " into a string that is displayed directly on the control panel. If this string is longer than 150 (?) characters, it should be truncated and " ..." added to the end. Following this list of "molecules" should be a link View all that should open the present subwindow.

RECORD HELP DOCUMENT: I have started a record help document (click on any text record in Features), featdoc.htm. It is generated in a partially automated manner from header_records[i][...]. As you expand header_records[], make sure each record name link works properly. Later we'll add specific information, and links to the Help/Index/Glossary where appropriate.

Create a link at the bottom of the text features list "Also not found". This should display a list of records that were looked for but are not listed individually as "not found" because they are not important when not found. Examples: OBSLTE, SPRSDE. This can be in the record help document.

A large number of magnesium ions in 1JJ2 reveal several bugs in the listing of ligand residues in the Ligands table. Long lines need to be broken into shorter lines (using string.lastIndexOf("match", starting_index)).

Resolution, for NMR, I think instead of saying "not found" we should show the actual record, typically "NOT APPLICABLE.". Also there seems to be some error in displaying Method for NMR, adding spurious text (compare Features display with raw header EXPDTA).

Other requests for Rosalia are in an earlier email from Eric.

TO ADD A NEW RECORD TO THE FEATURES PAGE

1. In shared/header.js, make a new call to define_header_record( ) to define a new set of elements in the array header_records[i] for the new record.

2. In shared/header.js create a new function named define_hxxx(), where xxx is the lowercase version of recname in header_records[], preceded by "h". For example for TITLE, the function is define_hsubtitle(). See existing define_hxxx() functions for examples of how to get desired substrings, e.g. define_hheader(). The goal of each function is to put the record(s) in top.pdbheader[foi]["hxxx"], where again xxx is the recname, for example ["htitle"]. In instances where multiple fields must be extracted from a single record (notably HEADER), separate elements such as ["hclassification"] and ["hdepdate"] will be needed.
3. Hquery.js needs to call the define_hxxx() functions. This is done automatically

4. In protexpl/features.htm, in write_text_features(), add a call to list_text_feature2( ) to display the new record.

5. Add specifics to FEATDOC.JS, which will then be shown in FEATDOC.HTM.