# User Guide for the HELM Antibody Editor V1.0

*Stefan Klostermann, Roche Innovation Center Penzberg, 19.12.2014*

Table of Contents

[User Guide for the HELM Antibody Editor V1.0 1](#_Toc406756921)

[General 2](#_Toc406756922)

[1st steps with the HELM (Antibody) Editor 2](#_Toc406756923)

[Run an example 2](#_Toc406756924)

[Sequence Loader 2](#_Toc406756925)

[Domain Detection Window 3](#_Toc406756926)

[Antibody Editing Window 3](#_Toc406756927)

[Details Box 4](#_Toc406756928)

[Connect domains manually 5](#_Toc406756929)

[Extend the example and add an oligo to a domain 5](#_Toc406756930)

[Options, parameters and processes 9](#_Toc406756931)

[Antibody Editor Settings 9](#_Toc406756932)

[Domain Detection Window 11](#_Toc406756933)

[Antibody Editing Window 12](#_Toc406756934)

[Libraries: Use, syntax and editing 13](#_Toc406756935)

[Domain library 13](#_Toc406756936)

[Mutation library 14](#_Toc406756937)

[Autoconnector rule set 15](#_Toc406756938)

[Contact 17](#_Toc406756939)

## General

The HELM Antibody Editor V1.0 is an extension of the HELM Editor V1.2 resulting in HELM Editor V1.3 to handle even complex antibody formats for their analysis and registration.

This document describes the new features and provides background details on the internal libraries in use and processing steps.

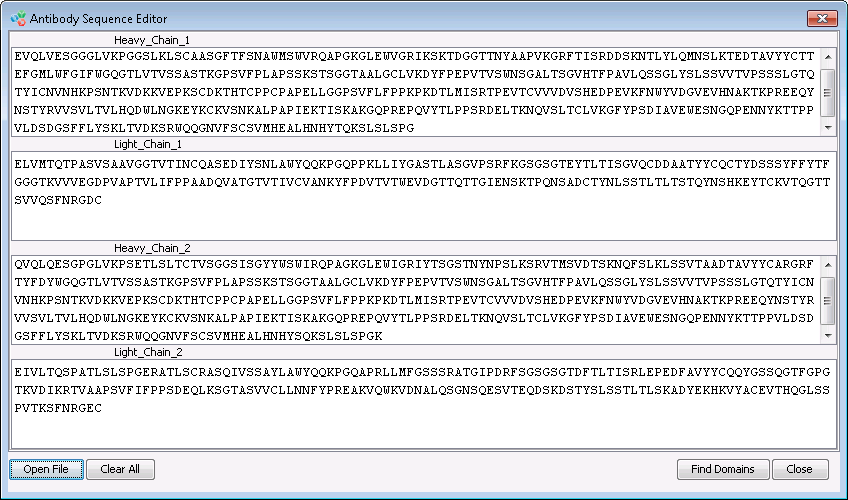
## 1st steps with the HELM (Antibody) Editor

* Build jar file
* Run HELMAntibodyEditor.jar 🡪 this should install a folder for the monomer library in C:\Users\<user>\.helm

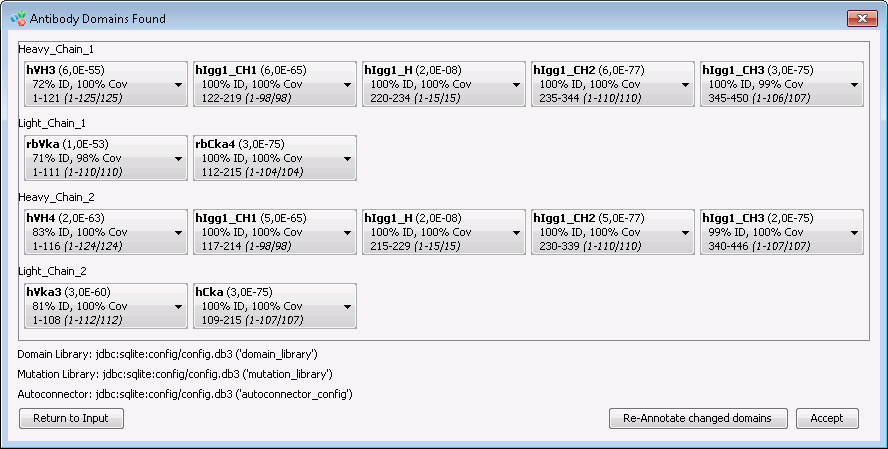
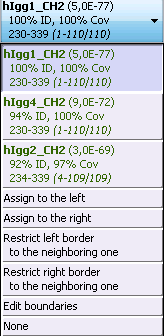
## Run an example

As a practical illustration of the new HELM Antibody Editor functionality the following section uses an provided example and showcases the steps and options.

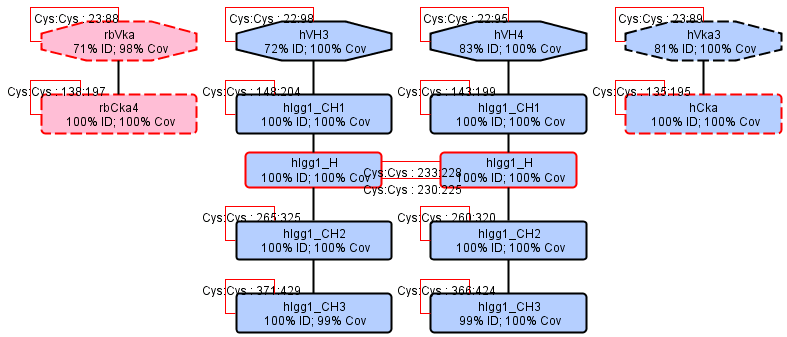
### Sequence Loader

* Open main menu > Tools > Antibody Tools > Launch Antibody Editor
* Click “Open File” and open “sample\AntibodyEditor\Test antibody”  
  to result in:  
  

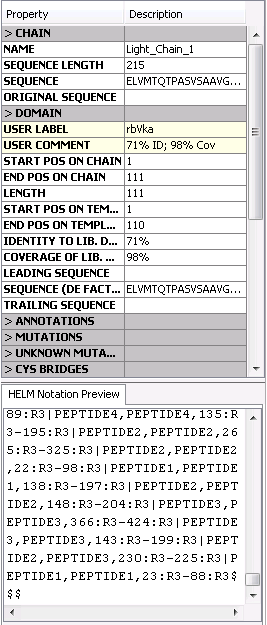
### Domain Detection Window

* Click “Find Domains” to result in:  
  
* Click on domains to see further options (other BLAST hits, assignment of the domain to neighboring ones, restricting domain boundaries to neighboring ones if BLAST hits overlap, edit boundaries manually, delete domain annotation = “None” 🡪 s. details in chapter “Domain Detection Window” below)  
  e.g.:  
  

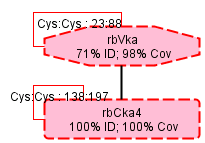
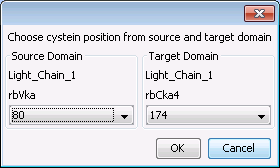
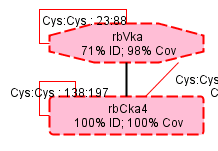
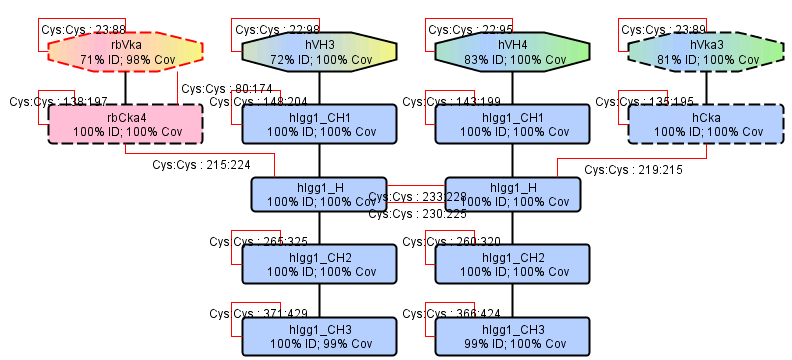
### Antibody Editing Window

* Click “Accept” to result in  
  
* This is a fake antibody with a rabbit light chain to the left. Note that for demonstration purposes in addition to the intradomain Cys-Cys bridge both “rbVka” and “rbCka4” domains contain an additional Cys that form an additional bridge and “rbVka” contains even another (4th) Cys located (located in the CDR3).
* Legend
  + **Shape**: octagons = variable domains, rounded squares = constant domains, small rounded squares = hinge regions, ovals = other domains (e.g. linkers, tags, sites, peptides incl. signal peptides)
  + **Border**: dashed = light chain, solid = heavy chain, red = free Cys residues
  + **Background color**: blue = human, light red = can be humanized (e.g. rabbit domains), gray = not human and can’t be humanized (e.g. linkers, peptides)
  + **Red lines**: indicate the Cys-Cys bridges 🡪 the application tries to automatically establish as many of these bridges as possible. Remaining ones have to be done manually.

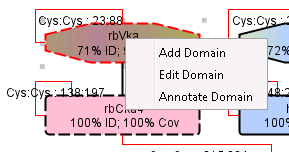
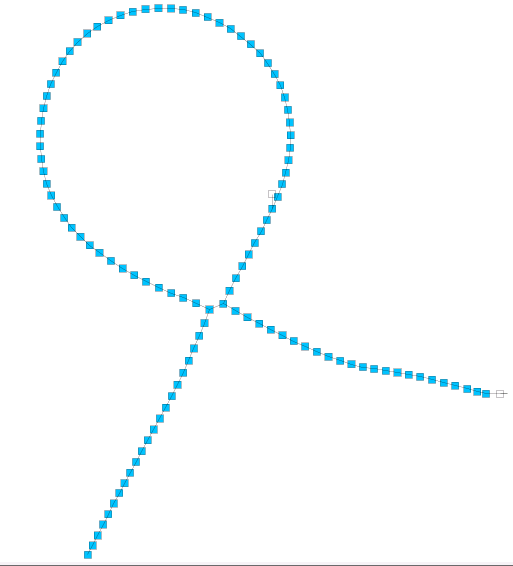
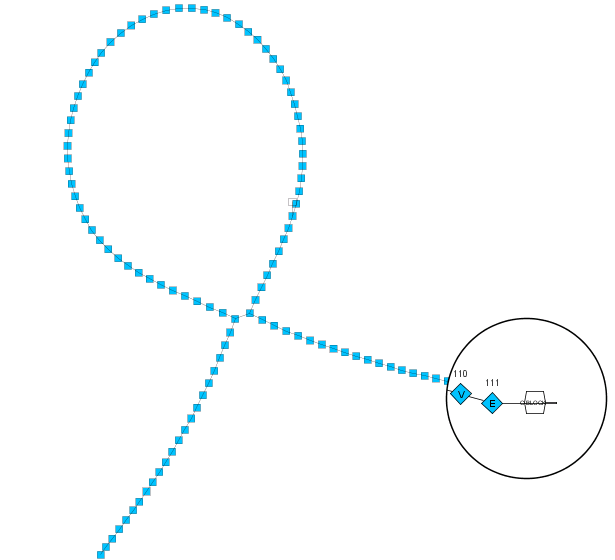
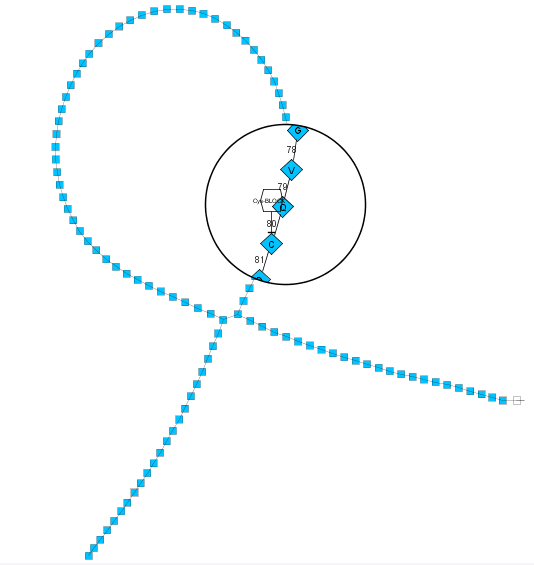
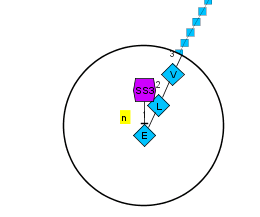
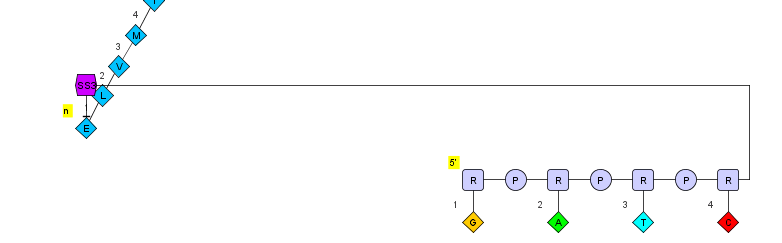
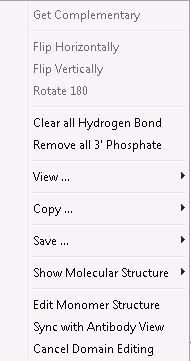
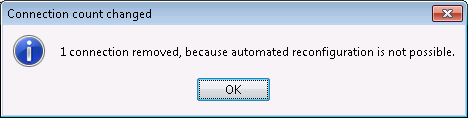
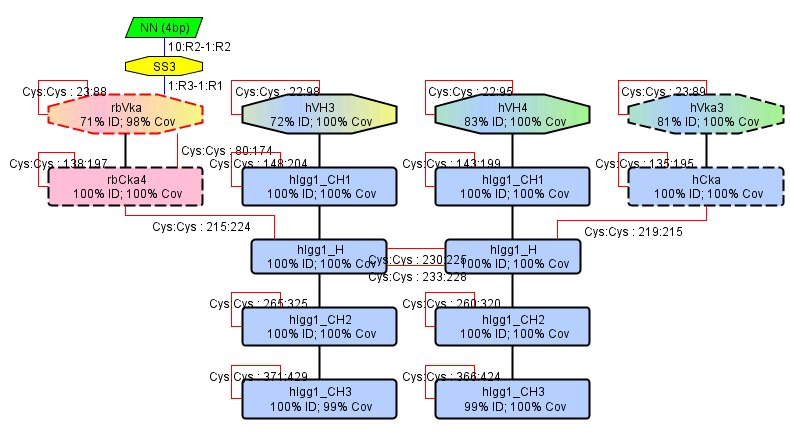
### Details Box

* Click on the upper left domain to see its details in a box to the right:  
    
  This box contains all data on the full chain (“>CHAIN”), the domain found as entered (“>DOMAIN”), known and unknown mutations (“>MUTATIONS”, “>UNKNOWN MUTATIONS”), all, connected, and free Cysteins (“>CYS BRIDGES”), and the domain from the domain library that has been used to annotate the given domain (“>RECOGNIZED DOMAIN FROM LIBRARY”).
* The always up-to-date HELM code for the complete molecule on the canvas is given in the lower right corner.

### Connect domains manually

* Draw a line between  to connect the additional (rabbit specific) Cys-Cys bond of these 2 domains. This opens  🡪 Click “OK” to see 
* In the same manner connect “rbCka4” and the left hinge “hIgg1\_H” and the right hinge “hIgg1\_H” with “hCka” to result in :  
    
  Note that the paired Fv are marked by a common background color.
* This antibody is now ready for registration.

### Extend the example and add an oligo to a domain

* Right-click the upper left domain  and choose “Edit Domain”
* In the upper canvas the selected domain is shown:   
  
* Note the 2 Cys residues that connect to other domains are blocked:
  + The domain C-terminus connected to the N-terminus of the “rbCka4” domain 
  + and the additional bridge into the “rbCka4” domain: 
* Add “SS3” from the monomers: “Chemical Modifier” > “Bi-Functional” > Select it and drag it onto the canvas. Connect it to the free N-terminus – while agreeing to both windows that pop up with “OK” to see: 
* Load the nucleotide sequence “GATC” onto the canvas and connect it to the “SS3” to see:  
    
  Note that you might also add peptides as well as CHEM payloads.
* Right-click on the canvas to see this context menu:  and select “Sync with Antibody View”. Click “OK” in the window  that indicates that the Cys-Cys bridge between “rbVka” and “rbCka4” had to be removed for technical reasons and thus has to be re-drawn. Once done this is the final molecule ready for registration:   
  

## Options, parameters and processes

The complete new HELM Antibody Editor functionality is available under the main menu > Tools > Antibody Tools:

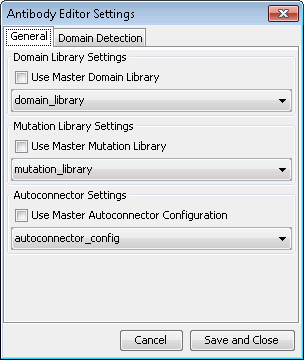
* **Launch Antibody Editor …** launches the main HELM Antibody Editor as shown in the example above. Files can be loaded by “Open File” or by drag & drop. File formats can be (multiple) FASTA or Vector NTI.
* **Load Antibody from XML …** loads antibodies from XML files saved from the Antibody Editing Window.
* **Launch Antibody Domains Library Editor …** displays the antibody domain library (s. below). Currently the editing functionality is disabled due to technical reasons.
* **Antibody Editor Settings** are detailed below.

### Antibody Editor Settings

In the HELM Antibody Editor settings (Main menu > Tools > Antibody Tools > Antibody Editor Settings) there are 2 tabs:

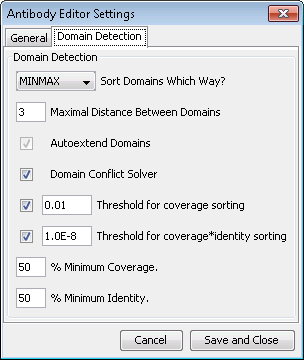
* The “General” tab provides the selection of the libraries to be used for domain detection, mutation annotation and assembly of modules.
* The “Domain Detection” tab summarizes parameters that influence the detection of domains within an entered peptide chain.

The tab “General” allows to select the libraries. In a company installation the 3 check boxes “Use Master …” select for a remote database system. As a starter the public version on GitHub contains a SQLite library file in config\config.db3 that might contain different instances of the 3 libraries detailed further below. In the distributed version there is only one instance each that is set default:



#### Domain Detection Settings

To understand the settings in this chapter first the domain detection procedure has to be briefly described: The entered peptide chains are chopped into domains using BLAST versus the domain library. Domain hits are filtered for their %identity and %coverage (s. parameters below), sorted and finally assembled. In the Domain Detection Window the entered chains are listed below each other and the domain hits represented next to each other in rows. These domain hits are sorted depending on different criteria that again depend on the BLAST e-value (given in brackets after the domain name) according to options in the Domain Detection settings (s. below).

The default (and recommended) window looks like  and contains the following options:

* **Domain Detection – Sort domains Which Way?**: The MINMAX algorithm combines the strength of the alternative algorithms MIN (finds domains as small / granular as possible) and MAX (finds domains as large as possible): It maps the filtered and sorted BLAST hits to the entered peptide sequence. Any domain with a lower sorting rank is either also mapped onto the entered peptide sequence as space allows (max. 1 residue overlap accepted to already mapped domains) or otherwise is placed below an already mapped domain and thus assigned to its given position. Thus the best (top sorted) hits determines the position and size of the detected domains.
* **Maximal Distance between Domains**: If BLAST can’t annotate stretches between domains these “domains” are labelled “NN (…AAs)”. In case the number of such residues is equal or smaller than this parameter this inter-domain stretch is not marked as a domain.
* **Autoextend Domains**: In case a given (entered) domain has terminal differences to the best matching domain from the domain library BLAST will fail to return the full alignment. If this option is checked the given domain is extended beyond the BLAST hit until either the length of the library domain is reached or other detected domains would cause an overlap conflict. This is useful to extend especially variable domains to their full size as the given antibodies might well diverge from the consensus sequences in the domain library. (This option is e.g. applied for the domain “rbVka” in the provided example.)
* **Domain Conflict Solver**: In case two domains share a common residue at their ends BLAST might return overlapping domains that are marked by a red border of the conflicting domains in the Domain Detection Window. The software automatically assigns this common residue to one of these domains if this option is checked. This option is quite useful to support batch registration of biologicals as it avoids further user interaction in case of conflicting domains. Yet there is no control to which domain the common residue is assigned.
* **Threshold for coverage sorting**:
  + BLAST hits with an e-value lower than this parameter are significant and sorted for their e-value.
  + BLAST hits with an e-value higher than this parameter are often linkers (e.g. 2G3S) and just sorted for their coverage (1st) and e-value (2nd) to select the best hit.
* **Threshold for coverage\*identity sorting**:
  + BLAST hits with an e-value lower than this parameter are highly significant (e.g. full domains) and sorted for their coverage\*identity. This helps to select the better match from close consensus sequences independent of their length – an attribute that influences the BLAST e-value in an non-controllable manner.
* **% Minimum Coverage**: Filters for BLAST hits equal or above % coverage specified by this parameter. “%Coverage” relates to the extent to which a domain from the domain library is used to annotate a sequence stretch of the input sequence. This parameter shall remove small stretches of domains from the library to be matched to the input sequence.
* **% Minimum Identity**: Filters for BLAST hits equal or above % identity specified by this parameter.

#### Sorting of hits

As detailed above 2 parameters influence the sorting of the hits. The following list illustrates the scale from low (significant) to high (not significant) e-values:

Coverage\*Identity sorting (for large domains)

…

**Threshold for coverage\*identity sorting** 🡪 to be applied only on large domains with focus on them.   
Alternatively this value can be set equal or close to “Threshold for coverage sorting” with the effect that large and small domains are treated the same way and thus the entered peptide chain might be chopped into a set of smaller domains instead of one modified larger one. This has to be evaluated for each user group / company independently.

…

e-Value sorting (for small domains)

…

**Threshold for coverage sorting**

…

Coverage / e-value sorting (for e.g. xGyS linkers)

### Domain Detection Window

Note: s. above the “Domain Detection Settings” for details on the domain detection by BLAST.

In the Domain Detection Window the entered chains are listed below each other and the domains are represented next to each other in rows.

In case two domains share a common residue at their ends BLAST might return overlapping domains that are marked by a red border in the Domain Detection Window. If the Domain Conflict Solver option is checked the software assigns this common residue to one of these domains.

When clicking on a domain further BLAST hits from the same regions are listed below according to their sorting order with their colors indicating their sorting blocks (s. sorting details above). Any of those domains can be selected to manually override the presented top hit.

Further options are:

* **Assign to the left/right**: assign a complete domain to the neighboring left/right one. Used if a (smaller) domain is obsolete and should be part of another neighboring one.
* **Restrict left/right border to the neighboring one**: adjust left/right domain boundary to the border of the neighboring one such that the overlap is solved. Used in case of overlapping BLAST hits.
* **Edit boundaries**: manually edit the boundaries of a domain. Used to have full control over the domain boundaries.
* **None**: delete domain assignment in case the annotation is wrong. This domain can later be manually annotated in the Antibody Editing Window.

Any “modified” domains are labeled as such to indicate this manual intervention. By clicking “Re-annotate modified domains” these domains are re-BLASTed and re-annotated.

The button “Return to Input” returns to the Sequence Loader and “Accept” transfers the domains to the Antibody Editing Window.

### Antibody Editing Window

Note: The legend (s. above), the Details Box (s. above), manual connection of domains (s. above) and adding oligos, peptides and/or chemicals to domains (s. above) are explained in the example described above.

From the Domain Detection Window the domains are analyzed for mutations and annotated accordingly (s. details below in the chapter “Mutation library”).

#### Cys-Cys bond creation

Next the antibody is assembled with as many Cys-Cys bonds automatically created as possible according to the following steps:

* Create intra-domain Cys-Cys bonds as defined by the **domain library**
* Create hinge to hinge Cys-Cys bonds (hard coded)
  + Only if same subtype, same number of Cys, sequences might differ
* Create hinge to constant light chains Cys-Cys bonds (hard coded)
  + Only if HC or LC or both are fully identical (assuming 2xHC + 2xLC)
* Create mutation defined inter-chain Cys-Cys bonds as defined by the **mutation library**
  + E.g. knob into hole
* Create intra-chain / inter-chain Cys-Cys bonds as defined by the **auto-connector rule set**
  + E.g. scFv, scFab

#### Canvas options

All bonds (thin red lines) can be selected, the position of their anchor points be adjusted and even deleted.

Domains can be right-clicked providing the following options:

* **Add Domain**: like “Edit Domain” but provides the domain to be added onto the upper canvas together with the edited domain itself
* **Edit Domain**: s. chapter “Extend the example and add an oligo to a domain” above.
* **Annotate Domain**: re-annotate domain by BLAST

A right-click on the canvas opens a menu with the following options:

* **Back to Domain Recognition**: opens the Domain Detection Window. This can be used to restore the automatically generated antibody assembly if needed.
* **Reset Layout**: resets layout to yFiles defaults
* **Reset Cystein Bridges (+Layout)**: resets layout and removes all intra-domain Cys-Cys bonds as defined by the domain library

Antibodies can be saved in and loaded from XML files using the options “Save Antibody” and “Load Antibody”.

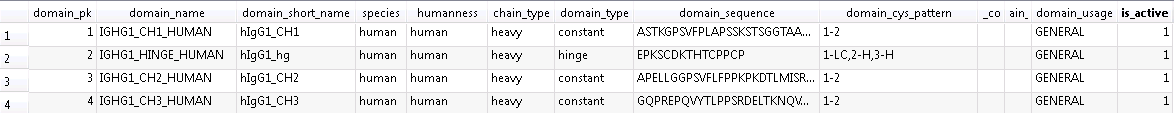
## Libraries: Use, syntax and editing

The three libraries described below are provided as SQLite file in config\config.db3

We can recommend <http://sourceforge.net/projects/sqliteman/> as easy and standalone viewer / editor for SQLite database files that does not has to be installed under Windows. This tool allows you to extend the library for e.g. linkers, novel Fc mutations , and rules that you use to detect special components or constructs.

### Domain library

The domain library is used to identify and annotate the domains as building blocks of peptide chains. All additional domains, linkers, peptides, conjugate proteins that are used should be entered in this library.

The domain library (only 1st 4 lines) looks like …

and thus contains the following columns:

* **domain\_pk**: used as internal primary key.
* **domain\_name**: full description of the domain. Name that is referred to in the mutation library.
* **domain\_short\_name**: name used in BLAST and in the GUI.
* **species**: human, mouse, rat, rabbit or empty / “-“ for e.g. linkers
* **humanness**: human, humanizable if not human yet, non-human if not humanizable as e.g. linkers
* **chain\_type**: heavy, kappa, lambda, empty / “-“
* **domain\_type**: variable, hinge, constant, empty / “-“
* **domain\_sequence**: amino acid sequence
* **domain\_cys\_pattern**: the pattern of Cys residues as ordinal numbers that pair within this domain, heavy chain domains might pair with light chains “LC” and light domains might pair with hinge “H”. Multiple bonds are separated by a comma.
* **domain\_comment**: an arbitrary comment for your convenience.
* **domain\_usage**: only “GENERAL” entries will be used by BLAST.
* **is\_active**: only entries with is\_active=1 will be used by BLAST. 0 means ‘inactive’.

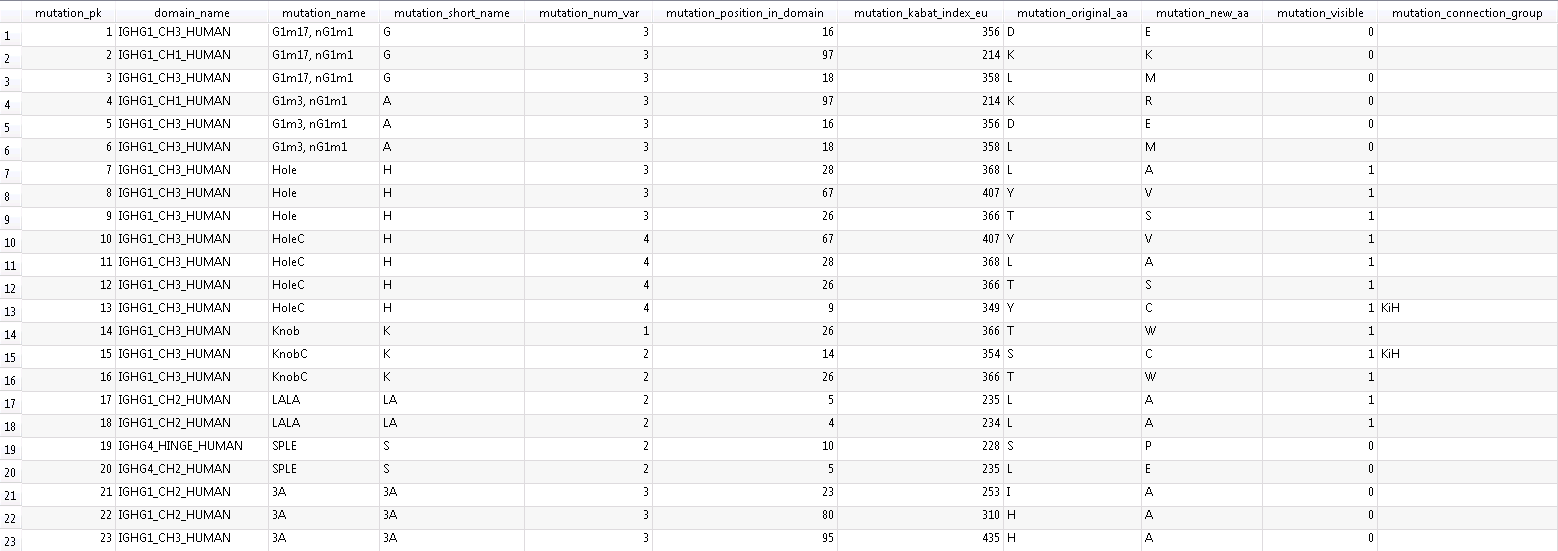
The domain library contains the variable and constant antibody domains for the species human, mouse, rat, and rabbit. The **variable domains** (heavy, kappa, lambda only for human) are provided as consensus sequences that have been generated from according IMGT entries with CDR3s represented by X. For human heavy and kappa chains the general consensus has been additionally split into gene family specific consensus sequences to allow a better characterization of such domains and to avoid a mistyping of more rare human domains as non-human in case e.g. the mouse consensus would be closer to a rare human variable region than the human main consensus itself.

The **constant regions** are based on Swissprot (that by itself has received the Ig constant regions in 2008 from IMGT) and IMGT and are chopped into domains like CH1, hg (hinge), CH2, and CH3.

* Lefranc, M-P.  
  Immunoglobulins: 25 years of Immunoinformatics and IMGT-ONTOLOGY.  
  Biomolecules. 2014, 4, 1102-1139; doi:10.3390/biom4041102 Open access,<http://www.mdpi.com/2218-273X/4/4/1102>
* Lefranc M-P, Giudicelli V, Duroux P, Jabado-Michaloud J, Folch G, Aouinti S, Carillon E, Duvergey H, Houles A, Paysan-Lafosse T, Hadi-Saljoqi S, Sasorith S, Lefranc G, Kossida S.   
  IMGT®, the international ImMunoGeneTics information system® 25 years on.   
  Nucleic Acids Res. 2014 Nov 5. pii: gku1056. [Epub ahead of print]   
  [PMID: 25378316](http://www.ncbi.nlm.nih.gov/pubmed/25378316)  
  [Free Article](http://nar.oxfordjournals.org/content/early/2014/11/05/nar.gku1056.abstract): <http://nar.oxfordjournals.org/content/early/2014/11/05/nar.gku1056.abstract>
* The UniProt Consortium  
  **Activities at the Universal Protein Resource (UniProt)**[Nucleic Acids Res. 42: D191-D198 (2014)](http://dx.doi.org/doi:10.1093/nar/gkt1140).

### Mutation library

The mutation library contains all deviations from the domains available in the domain library. Enter variations as e.g. mutations and allotypes of such domains to be identified. Mutations are grouped into sets (identical “mutation\_name”) that will only be recognized as such if all the listed mutations of a Mutation group are found - otherwise the mutations are listed as unknown mutations.

The mutation library looks like   


and thus contains the following columns:

* **mutation\_pk**: used as internal primary key.
* **domain\_name**: name of the domain as listed in the domain library under “domain\_name”
* **mutation\_name**: full name of the mutation (group)
* **mutation\_short\_name / mutation\_visible**: name that is shown in brackets within the domain as shown on the canvas if “mutation\_visible” is set to 1 else the mutation is not shown there but just in the details box to the right
* **mutation\_num\_var**: number of mutations (‘variation’) belonging to one mutation group
* **mutation\_position\_in\_domain**: absolute position of the mutation within the domain
* **mutation\_kabat\_index\_eu**: position of the mutation according to Kabat EU Index (not used yet)
* **mutation\_original\_aa / mutation\_new\_aa**: original and mutated amino acid
* **mutation\_connection\_group**: identical strings (not essentially numbers) indicate Cys residues to be connected if both Cys are found, and only if all mutations of the corresponding mutation group are found.

### Autoconnector rule set

Rules recognize modules like e.g. scFv and scFab and define the Cys bond pattern between domains of such modules or even beyond those modules. They are provided as text in a separate table (with line numbers to define their sequence) that look like:

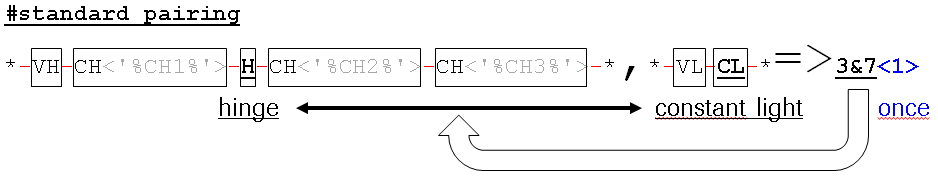
#standard pairing  
\*-VH-CH<'%CH1%'>-H-CH<'%CH2%'>-CH<'%CH3%'>-\*,\*-VL-CL-\*=>3&7<1>

#scFv  
VH-X<'?G?S'>-VL-\*=>1&3<n>  
\*-VH-X<'?G?S'>-VL=>1&3<n>  
VL-X<'?G?S'>-VH-\*=>1&3<n>  
\*-VL-X<'?G?S'>-VH=>1&3<n>

#scFab connect V  
VH-CH<'%CH1%'>-X-VL-CL-X<'?G?S'>-\*=>1&4<n>  
\*-X<'?G?S'>-VH-CH<'%CH1%'>-X-VL-CL=>2&5<n>  
VL-CL-X<'?G?S'>-VH-CH<'%CH1%'>-X-\*=>1&4<n>  
\*-VL-CL-X<'?G?S'>-VH-CH<'%CH1%'>-X=>1&4<n>

#scFab connect C  
VH-CH<'%CH1%'>-X-VL-CL-X<'?G?S'>-\*=>3&5<n>  
\*-X<'?G?S'>-VH-CH<'%CH1%'>-X-VL-CL=>4&6<n>  
VL-CL-X<'?G?S'>-VH-CH<'%CH1%'>-X-\*=>2&6<n>  
\*-VL-CL-X<'?G?S'>-VH-CH<'%CH1%'>-X=>2&6<n>

The connector rule for the standard pairing exemplifies the syntax as follows:



* Rules are processed bottom down.
* # precedes comment lines that are not processed.
* The rule notation separates domains by -.
* The domain acronyms refer to the domain library as follows:

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Chain** | **Domain** |
| VH | heavy | variable |
| H | heavy | hinge |
| CH | heavy | constant |
| VL | kappa or lambda | variable |
| CL | kappa or lambda | constant |
| X | any | any |

* Domains can be further specified by a text string to occur in the short name, set in <> brackets after the domain acronym, framed by ‘.
  + % is used as wildcard for 0-n characters.
  + ? is used as wildcard for a single character.
* X is used as wildcard for any domain but usually applies e.g. to linkers.
* N- or C-terminal ends might be extended by \* to indicate that the chain is allowed to be extended by e.g. modules like scFv but these \* domains do not count when numbering the domains.
* Different chains are separated by ,. Domains are continuously numbered across chains – just from left to right.
* The connection rule is given after the => symbol: The ordinal numbers of the domains to be connected by a Cys-Cys bridge are concatenated by &.
  + Per default it is assumed that such domains have only 1 free Cys left. Otherwise the ordinal number of the free Cys within a domain has to be specified after the domain number and separated by a #:  
    E.g. 3#2 specifies the 2nd free Cys of the 3rd domain.
* At the end of the syntax <1> indicates that the rule applies only if the given pattern is found once, otherwise the rule is not applied. A <n> specifies the pattern to be processed as often as it occurs.

## Contact

* Please contact[**Stefan.Klostermann@Roche.com**](mailto:Stefan.Klostermann@Roche.com?subject=Roche%20HELM%20Antibody%20Editor) in case of general and conceptional questions and feedback. Please contact Bernhard Schirm [**schirm@quattro-research.com**](mailto:schirm@quattro-research.com) for technical questions.