FlexX Command Line Documentation Version 6.6

Sascha Jung & Marcus Gastreich

September 3, 2026

©2026 BioSolveIT. All rights reserved.

Contents

1 Introduction **2**

2 Technical Prerequisites **3**
2.1 Required Software . . . 3
2.2 Licensing . . . 3

3 Jump Start: A Classical Docking **5**

4 Command Line Options **6**
4.1 Overview . . . 6
4.2 Program Options . . . 7
4.3 Configuration . . . 10
4.4 General Options . . . 12

5 Docking Types **13**
5.1 Standard Docking . . . 13
5.2 Template Docking . . . 14
5.3 Covalent Docking . . . 15

6 Re-Scoring With HYDE **19**

7 Further Reading, References **19**

8 Supported Warheads for Automated Covalent Docking **20**

1 Introduction

All links, references, table of contents lines etc. in this document are clickable.

FlexX is a tool for powerful protein-ligand docking at the command line.

FlexX is also a "component" within our flagship 3D modeling platform SeeSAR that has been conceived for drug researchers of any discipline and educational level (`https://www.biosolveit.de/SeeSAR`). Within SeeSAR's Docking Mode, you can easily define pharmacophore constraints, interactively inspect and post-process the results, e.g. by scoring the docking solutions with our HYDE scoring function.

FlexX at the command line features:

- Flexible ligand docking into target cavities
- Fuzzy template-based docking: Define a template (for example a common core), FlexX then takes your ligand input and places anything that is common onto each other and then places the rest of the ligand-to-dock into the cavity
- Covalent ligand docking with automated warhead detection and transformation to the protein-bound state
- Consideration of pharmacophore constraints *during* the docking (i.e., the pharmacophores are not acting as post-filters), thus accelerating the docking and delivering better results
- Optional enumeration of stereo isomers during the docking
- Processing of unlimited ligand numbers

FlexX does **not** contain our award-winning, desolvation-aware HYDE scoring. HYDE is available either within the graphical platform SeeSAR or as a standalone command line tool (from `https://biosolveit.de/download/?product=hydescorer`).

Please note that this package is a command line package.

FlexX is a super-fast anchor-and-grow docker. It operates based on an incremental construction algorithm to place a ligand into a binding site.[3] First, the ligand is split into fragments and an initial fragment or combinations of fragments are placed into different positions of the pocket. These initial placements are then scored with a very fast pre-scoring scheme. From these initial anchors the ligand is fully constructed by sequentially adding the remaining fragments and by scoring the intermediate solutions against each other. The top scored final solutions are retained and given to you as output.

2 Technical Prerequisites

2.1 Required Software

FlexX is a command line application. Control through a graphical user interface (GUI) is available in SeeSAR, BioSolveIT's 3D flagship modeling suite. We highly recommend you also install SeeSAR to swiftly carry out all sorts of preparational steps for the receptor, notably any relevant pharmacophore definitions etc.; the preparation can then be written to a `*.flexx` file and used at the command line with FlexX (this package).

Technically, you will need:

- The FlexX **application package** (from `https://biosolveit.de/download/?product=flexx`). Depending on your operating system, some libraries may have to be installed. Get in touch with us if that is the case:
  `mailto:support@biosolveit.com`. Please mention any errors/warnings that you see in your mail.
- A **shell** (Linux/Unix) or a terminal (macOS), or a command line environment (Windows; e.g.: cmd.exe or PowerShell)
- A valid **license** (from `mailto:license@biosolveit.de`), see below.

2.2 Licensing

FlexX needs a license to operate which is available from us. There are various sorts of licenses, but in most of the cases your early testing will employ a license file that simply needs to be put next to the executable, see below.

The license setup instructions will come with the license that we will send out — or that has already been sent out to you. In case you do not have a license yet, please get in touch with us at `mailto:contact@biosolveit.de`, and provide us with the necessary information. Please note: a SeeSAR license will be read as "valid" by FlexX and HYDE.

License File Locations A "test license" that you can request online and that is sent to you instantaneously can simply be placed next to the executable (`flexx.exe`, `FlexX` or `flexx` — depending on your operating system). For macOS please read on...

---

macOS Specialties On MacOS, the executable will typically reside inside the *.app package:

```
/Applications/FlexX.app/Contents/MacOS/FlexX
```

To place the short term test license there, you will have to go into the *.app package using a right mouse click (or CTRL-click) on FlexX.app in the Finder, and click on "Show package contents". In there, you will see the Contents/ subfolder, in there the MacOS subfolder, and in there, the FlexX executable. If you are about to use the **test license**, place is right there, next to the executable. A longer term license will be handled separately, we will tell you how when we send that very license.

When you call FlexX for the first time, go to the Finder, and navigate to the Applications folder. Do a right(!) click on FlexX.app, and — if applicable — confirm that you want to open the program. It will flash up once, and you are good to go at the terminal prompt from there on.

Obtaining a License File Using `--license-info` you can obtain information about the specification of your license server machine, the searched directories, and the validity of the currently used license files. This may also be useful when FlexX is not starting up as you would expect it to.

Call FlexX with the `--license-info` option, to see an output like this:

```
./flexx --license-info
================================================================================
 License Information:
================================================================================
Host-ID: "28f0763fa408 38c98636b9cb"
BIOSOLVE_LICENSE_FILE: /Applications/BioSolveIT/License
LM_LICENSE_FILE:

Currently used key/path after environment variables:
/software/flexx-6.5.0/flexx

FlexX:
 >> Valid key, expires on Tuesday, 31 December 2028

Request a license:
***
*** https://www.biosolveit.de/license/?product=flexx&operating_system=darwin...
***
```

Request an evaluation or longer-term license using the link that is provided at the very bottom of the output. Also, this output may help us to find out if there are any problems with your license or its setup.

3 Jump Start: A Classical Docking

To run a docking at the command line with all defaults, you will need at least:

- An input file containing the compounds-to-dock (`.sdf`, `.mol` or `.mol2` format, with 3D coordinates for every compound)

(a) Protein and reference ligand

- A protein input file (`.pdb`) **and**
- An input ligand file (i.e., the so-called reference ligand) with 3D coordinates in a binding cavity of the input protein (`.sdf`, `.mol` or `.mol2` format). The reference ligand helps to define the binding site.

(b) A docking definition file

- A docking definition file (`.flexx`) that you have exported from within SeeSAR's Docking Mode.

Now run this — with the file names replaced by your own names of course:
In case (a), that is, if you provided a protein file and a reference ligand file:

```
./flexx -i My3DLibrary.sdf -p MyProtein.pdb -r My3DReferenceLigand.sdf
        -o MyDockingOutput.sdf
```

In case (b), that is, if you have prepared and exported a docking definition file from SeeSAR:

```
./flexx -i My3DLibrary.sdf --docking-definition MyDefinitionFile.flexx
        -o MyDockingOutput.sdf
```

Depending on the operating system you use, please certainly also adapt the command line usage, e.g., use a slash or a backslash etc.

The above calls run the docking and create the output file `MyDockingOutput.sdf` that, by default, contains 10 docking solutions (poses) for every ligand — given that the docking was successful. The SD output file will also contain the respective docking score for every pose in a separate SD property field named `<BIOSOLVEIT.DOCKING_SCORE>` for post-processing purposes.

Please note:
We *highly* recommend that you re-score the docking poses with our award-winning, desolvation-aware HYDE scoring to improve the ranking. Information is here: `https://www.biosolveit.de/products/#HYDE`. Alternatively, you can load your docking results into SeeSAR for HYDE post-scoring, and take a look at the colored spheres to rationalize the findings — and make informed decisions on further steps you may want to take.

4 Command Line Options

4.1 Overview

An overview of all command line options is available by calling FlexX with **`--help`**. Default values are bracketed:

```
./flexx  -h

 Program options:
-i [ --input ] arg                  Library input molecule file. Supported file types are *.mol2, *.mol
                                    and *.sdf.
                                    Note: 3D coordinates must be provided, otherwise molecule is skipped.
-o [ --output ] arg                 Output file, only *.sdf is supported.
-p [ --protein ] arg                Protein file. Supported file types are *.pdb, *.ent, *.cif, *.mcif and
                                    *.mmcif.
                                    Note: Can't be used together with '--docking-definition'.
-r [ --refligand ] arg              Reference ligand file or files. Supported file types are *.mol2, *.mol
                                    and *.sdf.
                                    Note: Reference ligands must have 3D coordinates. The binding site for
                                    docking is defined as the pocket in which the ligand lies in the
                                    protein. If multiple reference ligands are provided, the references
                                    must lie in subpockets that overlap - the binding site will be the
                                    pocket comprising all the subpockets.
                                    Note: Can't be used together with '--docking-definition'.
--docking-definition arg            Run docking on the basis of this docking definition file (a .flexx
                                    file that can be exported from SeeSAR's Docking mode) - it contains
                                    the protein with predefined binding site and pharmacophore
                                    constraints.
                                    Note: Can't be used together with '--protein' or '--refligand'.
-d [ --docking-type ] arg (=0)      Determines the algorithm which is used for docking:
                                         0 [Standard-Docking]
                                         1 [Template-Docking] The algorithm for template docking mandates
                                           the simultaneous use of '--template'.
                                         2 [Covalent-Docking] The library input file must contain
                                           molecules with a warhead or exactly one linker atom.
-t [ --template ] arg               Template molecule file(s). Supported file types are *.mol2, *.mol and
                                    *.sdf.
                                    Note: The template molecule(s) must fulfill the same requirements as
                                    the reference ligand. Note also that library molecules must share a
                                    sizeable common substructure with a template otherwise they will be
                                    skipped.
                                    Multiple templates up to a maximum of 10 are supported. In this case,
                                    up to the maximum number of poses given by '--max-nof-conf' will be
                                    returned per template.

Configuration:
--allowed-6ring-confs arg (=0)      Conformations allowed for aliphatic six-membered rings:
                                         0 [Only chair conformations are allowed.]
                                         1 [Chair and twist-boat conformations are allowed.]
                                         2 [Chair, twist-boat and boat conformations are allowed.]
--clash-tolerance arg (=0)          Set clash tolerance for docking:
                                         0 [Standard]
                                         1 [Medium]
                                         2 [High]
--keep-input-sd-tags [=arg(=1)]     Keep input sd tags for the docking solutions.
--max-nof-conf arg (=10)            Maximum number of top-ranking poses for each docked molecule.
                                    Note: If a docking definition file is used, the value specified in the
                                    file is used.
--rigid-attachment-point [=arg(=1)] The attachment point of the covalent ligand at the protein remains
                                    rigid during covalent docking. Requires --docking-type=2.
--stereo-mode arg (=0)              Automatically flip stereo centers during docking:
                                         0 [Do not flip stereo centers]
                                         1 [Flip R/S stereo centers]
                                         2 [Flip E/Z stereo centers]
                                         3 [Flip R/S and E/Z stereo centers]

General options:
-h [ --help ]                       Print this help message.
--license-info                      Print license info.
--thread-count arg                  Maximum number of threads used for calculations. The default is to use
                                    all available cores.
--version                           Print version info
-v [ --verbosity ] arg (=2)         Set verbosity level
                                         0 [quiet]
                                         1 [error]
                                         2 [warning]
                                         3 [info]
                                         4 [steps]
```

Please note that the abbreviated, one-letter options are preceded with one dash – whereas the longer, named options are preceded with two dashes: `--`. If an option needs an argument (arg), you can include or omit the equals sign. Adapt the command line usage to your operating system and shell.

4.2 Program Options

–i [––input] arg Specify the ligand input file in `.sdf`, `.mol` or `.mol2` format. The input file should contain all the compounds which you want to dock. **All molecules must have 3D coordinates**, otherwise they are skipped during the docking run. Generation of 3D coordinates can be accomplished with SeeSAR. You can load your 2D molecule file into SeeSAR's Molecule Editor Mode and export the automatically generated 3D molecules as SD file. Alternatively, you can use our standalone command line tool Conformator to generate 3D conformations, especially recommended for large libraries with more than 50,000 molecules (`https://biosolveit.de/download/?product=conformator`).

Example:

```
flexx -i MyLigands.sdf
```

```
flexx -i MyFavorites.mol2
```

–o [––output] arg Specify a name for the output file (`.sdf` format) containing the docking solutions (poses). The output file will contain all successfully docked molecules from the input file with a maximum number of top-ranked poses as defined with the `--max-nof-conf` argument (see page 10).

Example:

```
flexx -o MyDockingPoses.sdf
```

–p [––protein] arg Specify a file containing the protein in `.pdb`, `.ent`, `.cif`, `.mcif` or `.mmcif` format.[1] If you do so, please make sure you also specify a reference ligand with the `-r` option (see below). The protein is obsolete if you have prepared a docking definition file (see `--docking-definition` option below).

Example:

```
flexx -p MyProtein.cif
```

–r [––refligand] arg Provide file(s) containing one or more reference ligand(s) in `.sdf`, `.mol` and `.mol2` format. The reference ligand will be used to determine the binding site for docking. Therefore, it is important you make sure the reference ligand has 3D coordinates and occupies a pocket of the protein which you specified via the `-p` option. To define the binding pocket, FlexX automatically detects all protein residues which have at least one heavy atom lying in 6.5 Å spheres defined around every reference ligand atom. This is very reasonable in most cases. However, if your reference ligand is very small (e.g. a fragment-like binder) the defined binding site may be to small to dock the larger compounds of your library. In this case you can manually extend the binding pocket by enlarging the fragment-like binder, e.g. by adding an aliphatic chain in SeeSAR's Molecule Editor Mode. The reference ligand is obsolete if you have prepared a docking definition file (see `--docking-definition` option).
NOTE: If you specify more than one reference ligand, the reference ligands must lie in subpockets that partially overlap. The overall binding site will be constructed from all overlapping subpockets. If you specify two reference ligands that bind, e.g., on opposite sides of the protein and their 6.5 Å binding

[1] In addition to protein structures, FlexX can also process DNA and RNA targets.

spheres (see above) do not overlap, it is not possible for FlexX to generate an overall connected binding site comprising both spheres. In such cases, you will receive an error message reporting that there is not sufficient overlap between the subpockets. To avoid confusion, we recommend to define a binding site for multiple reference ligands in SeeSAR's Binding Site Mode and export it as docking definition file (see Section 5.1 and Figure 1 and `--docking-definition` option below).

Example:

```
flexx -r MyReference.sdf

flexx -r MyReference.mol2

flexx -r MyReference1.sdf -r MyReference2.sdf

flexx -r MyCrystallizedFragments.sdf
```

––docking-definition arg As an alternative way to using a separate protein and reference ligand file (see above) you can run a docking on the basis of a docking definition file (`.flexx` file). The definition file must have been exported from SeeSAR's Docking Mode and contains the protein with a predefined binding site, optionally with additional pharmacophore constraints (see Figure1). These pharmacophores are used to guide the docking process and do not act as simple post filters.[2] Therefore, the docking calculations are accelerated and you generate only relevant docking solutions according to your constraints. Please note that the `--docking-definition` option cannot be used together with `-p` and `-r` options.

Example:

```
flexx --docking-definition MyReceptor.flexx
```

–d [––docking-type] arg(=0) Specify the docking type by giving an integer as argument:

0 **Standard Docking.** Non-covalent docking of ligands into the binding site. See Section 5.1 for more information.

1 **Template Docking.** Non-covalent docking of ligands into the binding site with the previous superposition of a common core onto one or more template ligand(s). Requires the specification of one or more template molecules with the `--template` option (see 4.2). See Section 5.2 for more information.

2 **Covalent Docking.** Covalent attachment of ligands with a suitable warhead to a compatible residue of the binding site. See Section 5.3 for more information.

Example:

```
flexx -d 2
```

–t [––template] arg Specify one or more template ligand file(s) (supported file types are `.sdf`, `.mol` and `.mol2`). FlexX can use ligands with a known binding mode as template. Template-based docking is performed by determining the Maximum Common Substructure (MCS) between the template and the compound-to-dock. Based on the MCS multiple overlays are generated to preserve the binding mode of the common core. You can use this for fast docking of congeneric compound series or for growing-like docking from an "anchor", e.g. a fragment-like ligand. The template ligands must have 3D coordinates and lie in the binding pocket of the protein that is defined with either the `--r` or `--docking-definition` option, otherwise they are skipped. Additionally, the input molecules must share a sizable common

substructure (five heavy atoms or more) with the template(s), otherwise they are skipped. An overall maximum of ten template molecules (either contained in a single file or in multiple separate files) can be specified at the same time.
NOTE: If you specify multiple template molecules, the maximum number of poses (specified with the option `--max-nof-conf`, see page 10) is returned for each template molecule, i.e., if you have a single input molecule and two template molecules and set the number of results to default (=10 results), you will get 20 results in total, ten for each template molecule.
NOTE: To run a template docking, the `-d` option must also be set to 1 (see above). See Section 5.2 for more information on template-based docking.

Example:

```
flexx -t MyTemplate.sdf

flexx -t MyTemplate1.sdf -t MyTemplate2.sdf

flexx -t MyCrystallizedFragments.sdf
```

4.3 Configuration

In this section, the default values for the respective argument are bracketed.

––allowed-6ring-confs arg(=0) Select the conformations allowed for six-membered aliphatic rings. By default, only low-energy chair conformations are tolerated. You can gradually adjust this behavior by allowing also the energetically less favorable twist-boat and boat conformations:

- 0 Only chair conformations are allowed. The default.
- 1 Chair and twist-boat conformations are allowed.
- 2 Chair, twist-boat and boat conformations are allowed.

Please note that if you provide a docking definition file the parameter will be read from the respective `.flexx` file and used as default. However, you can overwrite it with the `--allowed-6ring-confs` option.

Example:

```
flexx --allowed-6ring-confs 2
```

––clash-tolerance arg(=0) Set the clash tolerance for the docking run. Refers to the receptor-ligand overlap volume which is still tolerated for a docking pose to be considered valid. You may want to increase the tolerance if you use a very narrow binding site.

- 0 Standard. The default. Tolerance for the receptor-ligand overlap is rather low.
- 1 Medium.
- 2 High. High tolerance for the receptor-ligand overlap.

Example:

```
flexx --clash-tolerance 2
```

––keep-input-sd-tags For every molecule in the input SD file the associated SD tags are read and added to every docking pose generated for that molecule in the output SD file.
NOTE: Requires an input SD file.

Example:

```
flexx --keep-input-sd-tags
```

––max-nof-conf arg(=10) With this argument you can control the maximum number of generated docking solutions (conformations) per ligand. The default value is 10 conformations per compound. Please note that if you provide a docking definition file using the `--docking-definition` option the number of conformations will be read from the respective `.flexx` file and used as default. However, you can overwrite it with the `--max-nof-conf` option. For the virtual screening of large libraries, the default of 10 conformations will often be too high and produce large amounts of data, so that you may want to reduce this to 1-2 conformations that are usually sufficient to enrich promising binders. However, for smaller libraries and for re-docking of compounds it may be beneficial to increase the

maximum number of conformations up to 100 in order to generate more poses for the re-scoring with HYDE (see Section 6). After re-scoring, it is now more likely to find the near-native conformation among the top-ranked poses.

Example:

```
flexx --max-nof-conf 100
```

--rigid-attachment-point Keeps the attachment point of the covalent ligands at the protein rigid during docking. Requires the simultaneous use of `--docking-type=2`. You may want to keep the attachment point rigid if you want to a use a defined conformation of a protein, e.g. a snapshot from MD simulations, or if you want to perform a re-docking with the crystal structure ligand. See Section 5.3 for more information.

Example:

```
flexx --rigid-attachment-point
```

--stereo-mode arg(=0) This option allows you to control the treatment of stereo centers during the docking by giving an integer as argument. The default value is 0, e.g. stereo centers are maintained during docking. The following behavior can be selected:

- `0` Do not flip stereo centers. All stereo centers of the input molecules are maintained. This is the default value.
- `1` Flip R/S stereo centers.
- `2` Flip E/Z stereo centers.
- `3` Flip both R/S and E/Z stereo centers.

Flipping R/S and/or E/Z stereo centers may help to identify the isomer with the most promising binding geometry for your target. Keep in mind that flipping every stereo center can lead to a combinatorial explosion for compounds with multiple stereo centers! This may lead to extended runtimes for large docking libraries. Please note that if you provide a docking definition file the parameter will be read from the respective `.flexx` file and used as default. However, you can overwrite it with the `---stereo-mode` option.
NOTE: If you activate E/Z flips (stereo mode 2 and 3), the algorithm will try to flip cis amides to a trans conformation if it can (but not vice versa)!

Example:

```
flexx ---stereo-mode 3
```

4.4 General Options

–h [––help] Displays the command line help with short descriptions for every argument option. For more information see Section 4.1.

Example:

```
flexx --help
```

––license-info Shows command line information about the license setup you currently use. If you have any problems with your license, send an email to `mailto:support@biosolveit.com` and include this information. For more information see Section 2.2.

Example:

```
flexx --license-info
```

––thread-count arg Specify the maximum number of threads used for the docking calculations. By default, all available logical cores of your computer are used. You may want to reduce the number of threads if you want to run other computations on your computer at the same time, or if you share the compute resource.

Example:

```
flexx --thread-count 4
```

––version Displays information on the version of FlexX on the command line. In quoting FlexX, please mention this version number.

Example:

```
flexx --version
```

–v [––verbosity] arg(=2) You can set the verbosity level, e.g. the level of console output, by giving an integer as argument. The default value is 2. The following options are available:

- `0` Quiet. No messages will be displayed on the console. Errors will be ignored whenever possible.
- `1` Error. Only error messages will be displayed.
- `2` Warning. The default setting, warnings and error messages will be displayed.
- `3` Info. Additional information beyond errors and warnings are displayed.
- `4` Steps. In addition to the 'Info' option, the progress is displayed in detail.

Example:

```
flexx -v 0
```

5 Docking Types

5.1 Standard Docking

Standard Docking refers to the **non-covalent** docking of a ligand into the binding pocket of a receptor (`--docking-type=0`). This is the default behavior of FlexX, i.e., you do not have to specify the docking type explicitly. For standard docking, the ligand is treated flexible, whereas all residues of the binding site are kept rigid. A part of the ligand is flexibly placed in the binding pocket and the final docking pose is obtained by incremental construction of the complete ligand.[3]

To perform a standard docking, you need one or more input ligands with 3D coordinates and an appropriate definition of a binding site.

You can define a binding site with a protein (`.pdb` format) and a reference ligand that is bound to a certain part of the protein, e.g. load the biotin-streptavidin complex 1stp together with the biotin ligand (BTN, make sure to download the file with instance coordinates) from the PDB database (`https://www.rcsb.org/structure/1STP`) and use both files on the command line:

```
./flexx -i MyLigands.sdf -p 1stp.pdb -r 1stp_B_BTN.sdf -o MyDockingOutput.sdf
```

Alternatively, you may load the protein complex in SeeSAR and define the binding site (automatically or manually in the Binding Site Mode), then switch to the Docking Mode and optionally define pharmacophore constraints and adjust the docking parameters, then export a docking definition file (`.flexx` file, see Figure 1) and use it on the command line:

```
./flexx -i MyLigands.sdf --docking-definition 1stp.flexx -o MyDockingOutput.sdf
```

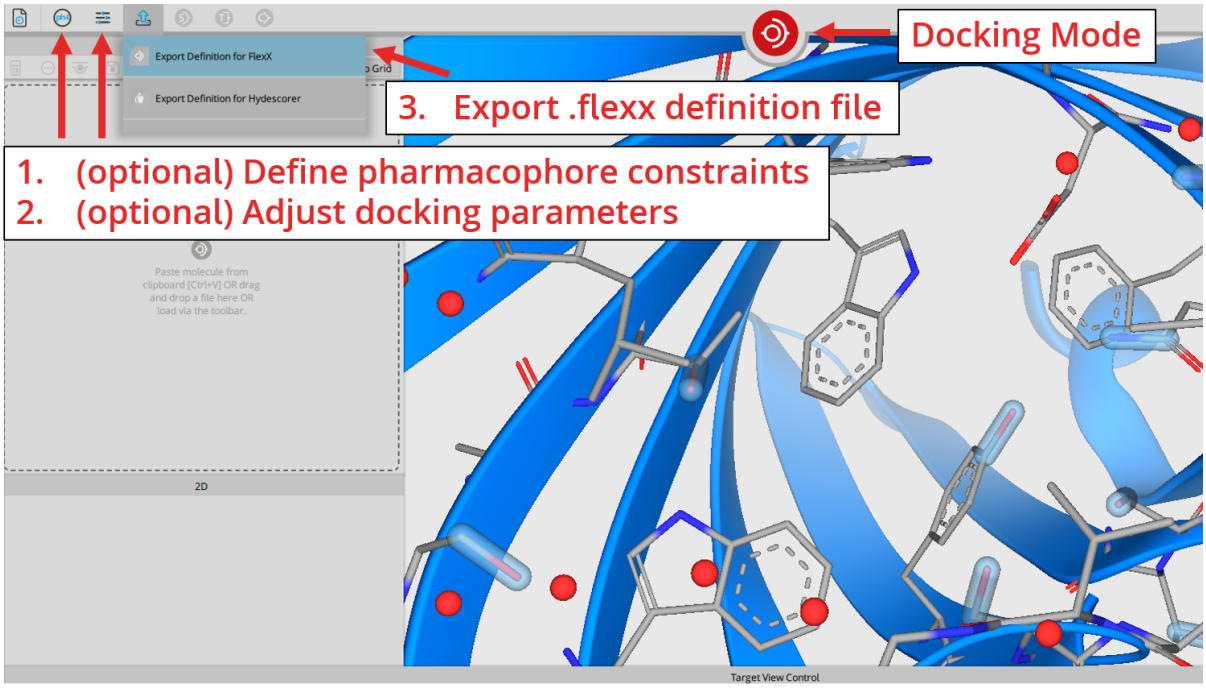


Figure 1: *Export a docking definition file from within SeeSAR's Docking Mode.*

5.2 Template Docking

Template docking is the superposition of a part of the input ligand onto a known conformation of a template ligand, then preserving the conformation of the superimposed part during generation of the docking poses (`--docking-type=1`). This is possible for ligands which you want to dock **non-covalently**. Typically, the template ligand is the bioactive conformation of a bound ligand from a crystal structure. The template ligand must have 3D coordinates and lie in the binding site of the protein (as defined either with a docking definition file (`--docking-definition`) or with one or more reference ligands(s) (`--reference`)) and must be specified with the `--template` option. The input ligands must share a significant common substructure (5 heavy atoms or more) with the template ligand, otherwise they cannot be superimposed. Template-based docking is performed by an adaptive matching procedure. First, the Maximum Common Substructure (MCS) between the template ligand and the input ligand is determined. Based on the MCS, multiple superpositions are generated to accurately preserve the binding mode of the common core and then the final solutions are obtained via the FlexX incremental construction procedure. You can use the template approach, e.g. for high-speed docking of congeneric compound series where the binding mode is known; for very fast mining of template-like structures from databases (non-suited molecules are skipped if they do not have a sizable MCS with the template); or for pseudo-growing of fragment-like binders into pockets.

To perform a template-based docking, an appropriate template ligand must be given via the `-t` option **and** the `-d` option must be set to 1. You can reuse the example 1stp (see Section 5.1), and use the crystal structure ligand (biotin) both as reference to define the binding site and as template ligand:

```
./flexx -i MyLigands.sdf -p 1stp.pdb -r 1stp_B_BTN.sdf -t 1stp_B_BTN.sdf -d 1
        -o MyDockingOutput.sdf
```

Alternatively, you can also use a docking definition file (see Figure 1 and Section 5.1) for template docking:

```
./flexx -i MyLigands.sdf --docking-definition 1stp.flexx -t 1stp_B_BTN.sdf -d 1
        -o MyDockingOutput.sdf
```

It is possible to specify multiple template ligands in a single docking run. Up to ten templates are supported at the same time within a single docking run. The template molecules can be contained either in a single file or in multiple files. You have to make sure that all template ligands lie inside the defined binding site (`-r` or `--docking-definition` option). All templates are treated individually, i.e., every input ligand is superposed onto all template molecules and suitable docking solutions (up to the desired maximum number of solutions, see `--max-nof-conf` on page 10) are generated independently for all templates. Consequently, e.g., if you dock a single input ligand with two templates and the default number of results (=10, see page 10), you will receive ten docking solutions per template molecule, i.e., a total of 20 overall docking results per input ligand.

5.3 Covalent Docking

FlexX can covalently dock molecules with a suitable warhead to an appropriate residue of the binding site (`--docking-type=2`). Typically, the ligand warhead is an electrophilic group which can react with a suitable nucleophilic residue in the binding site of a protein (typically a cysteine or serine residue). A reaction between a protein and a ligand normally consists of two steps: the first step involves the formation of a non-covalent complex between the ligand and the protein in which the electrophilic warhead comes close to the nucleophilic residue. Then, in the second step, the covalent bond formation takes place and the final protein-bound state (covalent complex) is obtained. With covalent docking, you can model this final covalent complex.

FlexX automatically detects ligands with suitable warheads and transforms them into the protein-bound state in the background, then they are docked to the appropriate residue of the binding site (see Figure 2). The most prominent warheads are currently supported (see Section 8 for a complete list) for automated transformation and docking. By default, the terminal bond of the protein residue (S-CH_2 bond in Figure 2, marked with a rotating arrow) is treated flexible during docking. You can keep this bond rigid if you set the `--rigid-attachment-point` option (see page 11). This is only recommended for special scenarios, e.g. if you want to dock to a selected conformation from a MD trajectory, or if you want to re-dock the crystal structure ligand.

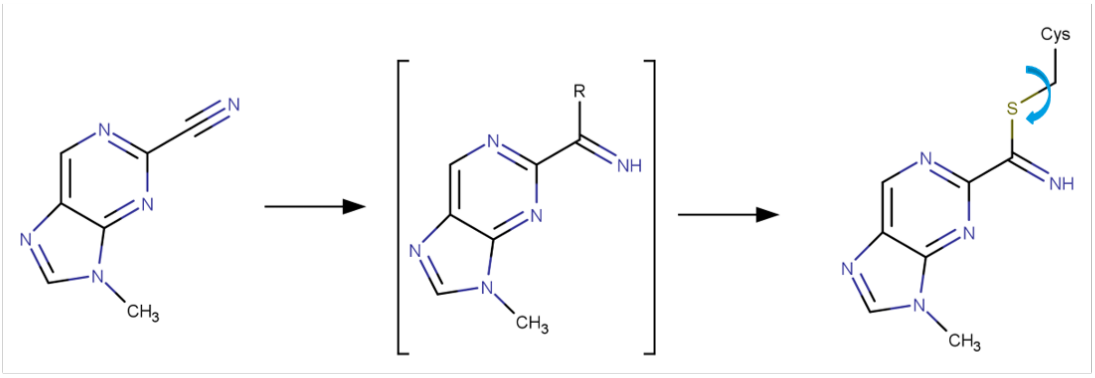


Figure 2: *Example for the automated transformation and docking of a ligand with a nitrile warhead.*

The recommended way to perform covalent docking on the command line with FlexX is by using a docking definition file. You can export the definition file from SeeSAR's Docking Mode (see Figure 3). As an example, load the cysteine protease Rhodesain in complex with covalently bound K11777 in SeeSAR (PDB 2p7u). Now switch to the Docking Mode. If you load a protein into SeeSAR which already contains a covalently bound ligand (covalent complex), then the residue to which the ligand is bound will be automatically defined as covalent attachment point (marked with a pink cylinder in SeeSAR's Docking Mode, see Figure 3). You can select a different covalent attachment point by simply clicking on the light-blue cylinder around the terminal atoms of the desired residue in the binding site. The desired attachment point is then highlighted in pink. Currently, seven protein residue types (cysteine, serine, lysine, threonine, tyrosine, glutamine, asparagine) can be selected for automatic preparation for covalent docking. If you want to dock to a different protein residue type, you have to add a linker atom to the protein residue you want to dock to. This can be done manually in SeeSAR's Protein Editor Mode (see Figure 6), the newly added linker can then be selected as attachment point in SeeSAR's Docking Mode as described before (Figure 3). After selection of the attachment point, you may define pharmacophore constraints and adjust the docking parameters (optional). Now you are ready to export a docking definition file (`.flexx` file) and use it on the command line for covalent docking:

```
./flexx -i MyWarheads.sdf --docking-definition 2p7u.flexx -d 2 -o MyDockingOutput.sdf
```

It is crucial to specify the covalent docking type (`-d=2`), otherwise the ligands are not transformed and docked non-covalently.

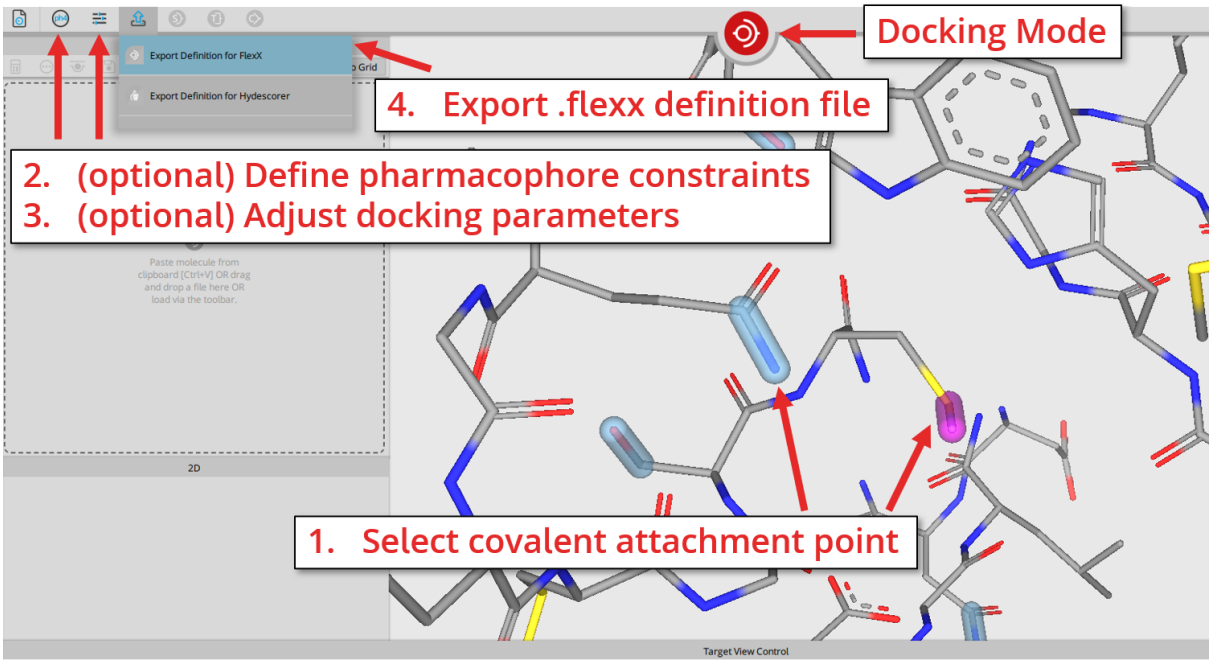


Figure 3: *Export a definition file for covalent docking from within SeeSAR's Docking Mode.*

It is also possible to perform a covalent docking with a separate protein and reference ligand to define the binding site. One possibility is to load a covalent protein-ligand complex and the respective reference ligand (which then is typically non-covalent if loaded from the PDB database) from the PDB database. For example, load the covalent cysteine protease complex 2p7u and the ligand (D1R, make sure to download the file with instance coordinates) from the PDB database (`https://www.rcsb.org/structure/2P7U`) and use it on the command line:

```
./flexx -i MyWarheads.sdf -p 2p7u.pdb -r 2p7u_B_D1R.sdf -d 2 -o MyDockingOutput.sdf
```

The covalent attachment point is now automatically detected as the residue to which the covalent crystal structure ligand is bound (in this case cysteine 25), and the binding site is defined with the help of the reference ligand.

Another option is to export a covalently bound ligand directly from SeeSAR and use it as reference ligand on the command line. This may be the crystal structure ligand as well as a covalent docking solution generated in SeeSAR. Covalent ligands exported from SeeSAR already carry a linker atom, i.e., the protein does not necessarily need to contain a covalently bound ligand to define the attachment point.

Automated transformation and docking is aware of new stereo centers generated during the transformation of the ligand to the protein-bound state (see Figure 4). A new stereo center created by the reaction with the protein residue at the attachment point is always enumerated, leading to two transformed isomeric ligands for the example in Figure 4. New stereo centers that are formed adjacent to the attachment point are not explicitly enumerated, but are *always* flipped during docking (i.e., both variants are considered for pose generation), independent of the selected stereo mode (`--stereo-mode` option, see page 11).

Automated covalent docking is currently possible for 36 different warheads (see Section 8). If you want to covalently dock ligands with different warheads, you have to prepare the ligands manually. If you want to dock compounds with a different warhead, you have to do the transformation of the ligand

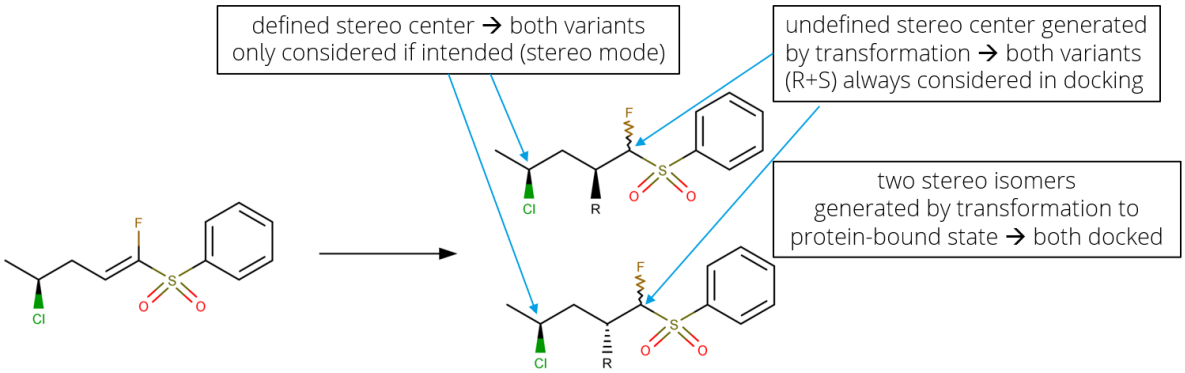


Figure 4: *Stereo isomers and undefined stereo centers generated by the transformation of a ligand to the protein-bound state. R = attachment point of the protein residue.*

to the protein-bound state manually, in such a way that the transformed ligand contains exactly one linker atom. You can add a linker manually to your ligands in SeeSAR's Molecule Editor Mode (for a single or a few compounds, see Figure 5) and save them as an SD file. For multiple compounds, it may be better to do the transformation with the ReactionSynthesizer from our CoLibri toolkit (`https://biosolveit.de/download/?product=colibri`, see Section 3.5 in the CoLibri user guide).

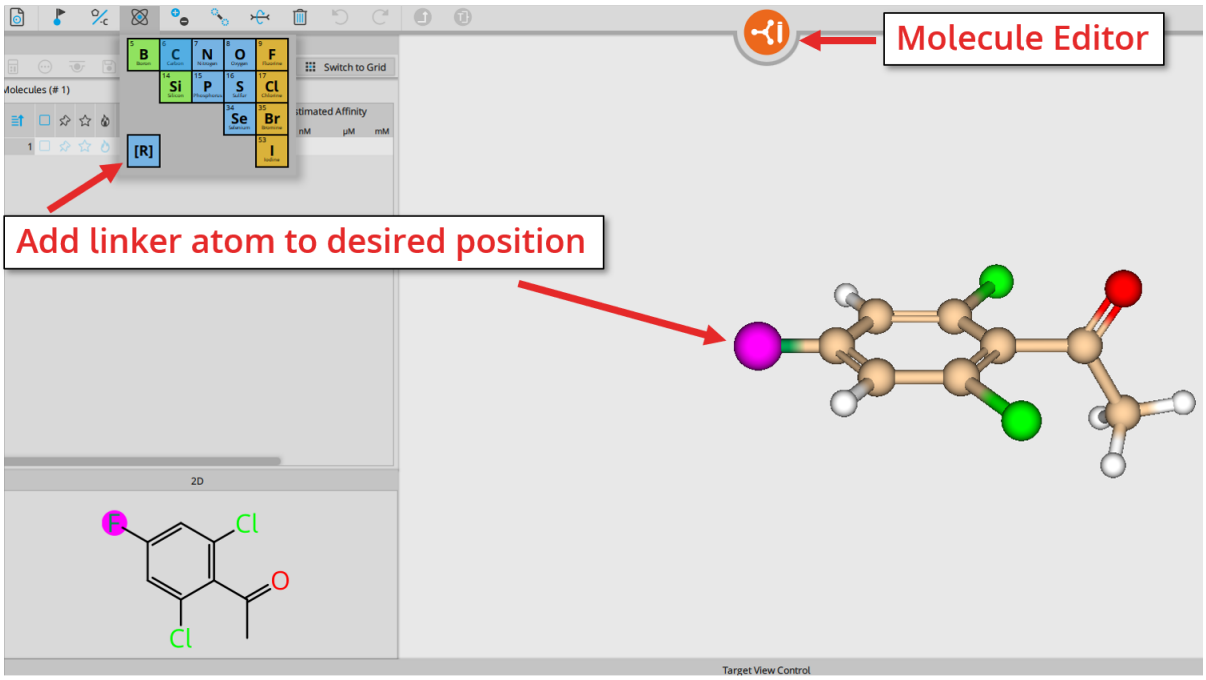


Figure 5: *Adding a linker atom manually to a ligand that cannot be transformed automatically in SeeSAR's Molecule Editor Mode.*

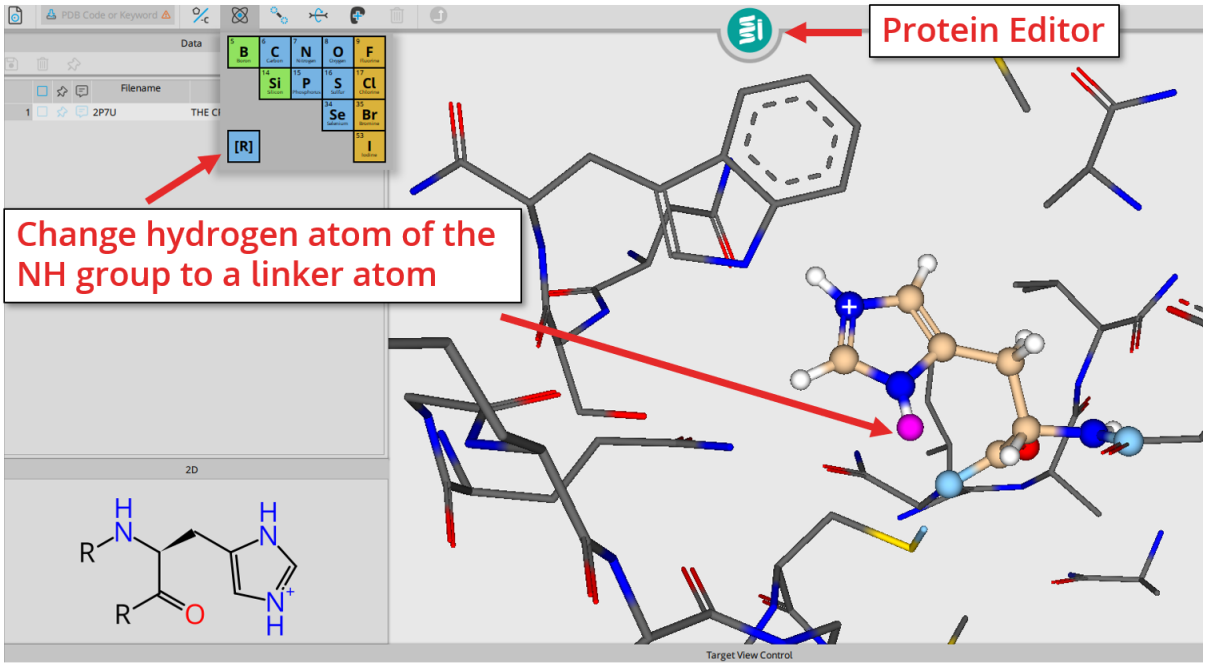


Figure 6: *Manual addition of a linker atom to a histidine residue in SeeSAR's Protein Editor Mode. In this example, PDB 2p7u was loaded into SeeSAR and transferred to the Protein Editor Mode. After selecting the appropriate histidine residue (HIS162), the hydrogen atom of the NH was replaced by a linker atom (R).*

6 Re-Scoring With HYDE

In any case, we recommend to optimize and re-score your docking solutions with our desolvation-aware HYDE scoring function to improve the ranking. You can either load your docking results file into SeeSAR or use the standalone HYDE command line tool (`https://www.biosolveit.de/products/#HYDE`). A major advantage of using SeeSAR is the visualization of the scoring results with the per-atom HYDE contributions. This enables you to easily rationalize the results and select compounds for further investigation.

7 Further Reading, References

The original ideas behind the FlexX incremental docking method have been cited hundreds of times; they are covered in the original publication by Matthias Rarey.[3] Additional information on performance and scoring can be found below in the references ([1] and[4]). Over the years, FlexX has undergone steady further developments and improvements, amongst them, for example, parallelization, extensions for covalent binders, fuzzy template-driven dockings and pharmacophore-guided dockings and lastly the automated transformation and covalent docking of ligands with a suitable warhead.

Additional information on the tool is available at

`https://biosolveit.de/products/#FlexX`.

Complementary tools, especially also the graphical platforms SeeSAR and infiniSee, can be obtained from the BioSolveIT website (`https://biosolveit.com`).

References

[1] Marcus Gastreich, Markus Lilienthal, Hans Briem, and Holger Claussen. Ultrafast de novo docking combining pharmacophores and combinatorics. *Journal of Computer-Aided Molecular Design*, 20:717–734, 12 2006.

[2] Sally A Hindle, Matthias Rarey, Christian Buning, and Thomas Lengauer. Flexible docking under pharmacophore type constraints. *Journal of Computer-Aided Molecular Design*, 16:129–149, 2002.

[3] Matthias Rarey, Bernd Kramer, Thomas Lengauer, and Gerhard Klebe. A fast flexible docking method using an incremental construction algorithm. *Journal of Molecular Biology*, 261(3):470–489, 1996.

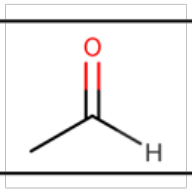
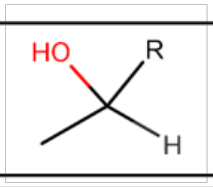
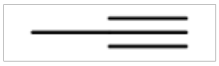
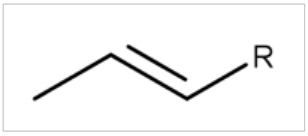
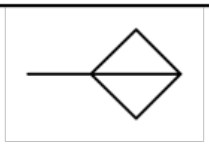
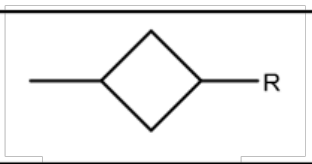
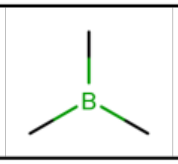
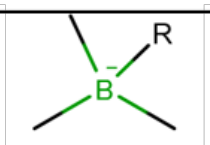
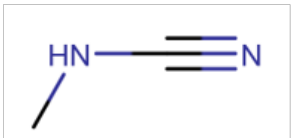
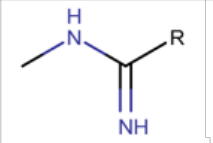
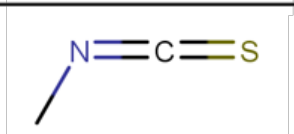
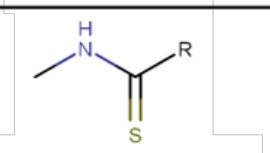
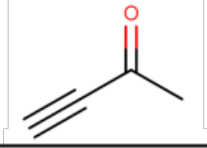
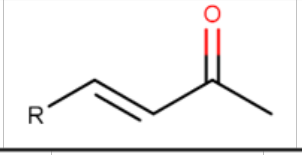
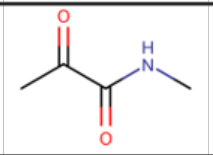
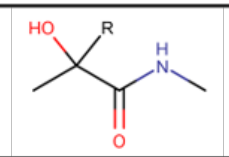
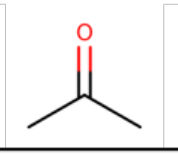
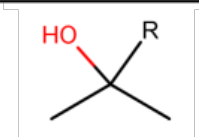
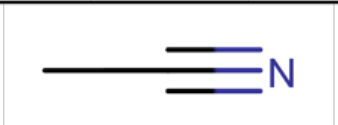
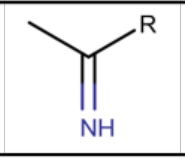
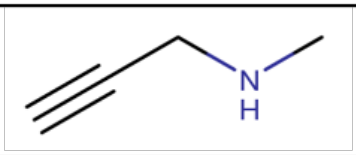
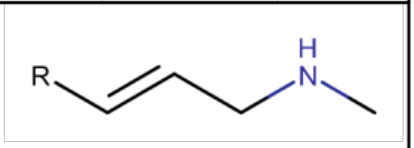
[4] Gregory L. Warren, C. Webster Andrews, Anna-Maria Capelli, Brian Clarke, Judith LaLonde, Millard H. Lambert, Mika Lindvall, Neysa Nevins, Simon F. Semus, Stefan Senger, Giovanna Tedesco, Ian D. Wall, James M. Woolven, Catherine E. Peishoff, and Martha S. Head. A critical assessment of docking programs and scoring functions. *Journal of Medicinal Chemistry*, 49(20):5912–5931, 2006. PMID: 17004707.

We wish you great success and much joy with FlexX!

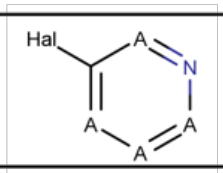
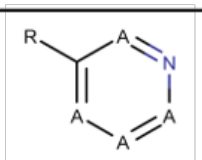
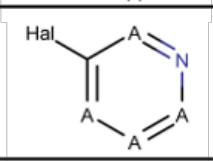
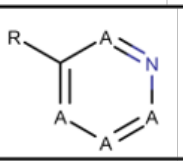
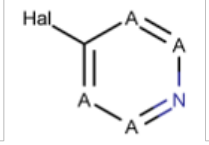
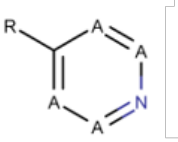
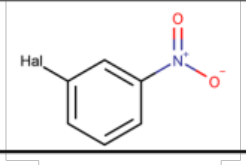
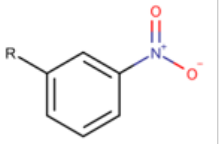
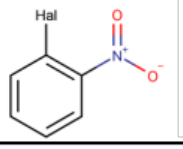
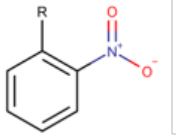
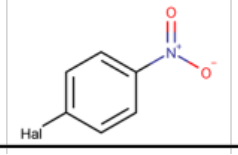
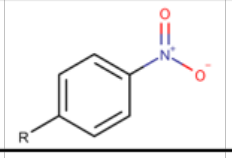
8 Supported Warheads for Automated Covalent Docking

This section contains all ligand warheads which will be automatically detected by FlexX, then transformed to the protein bound state and docked to an appropriate protein residue (`--docking-type=2`, see page 8 and Section 5.3 for more information).

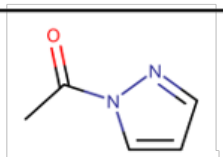
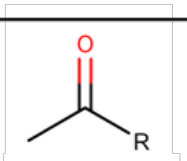
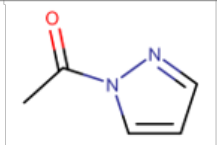
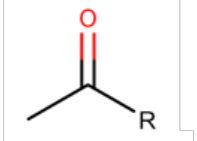
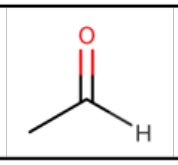
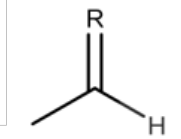
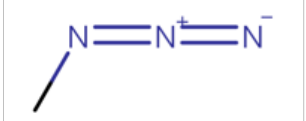
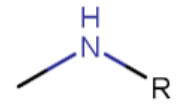
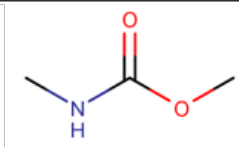
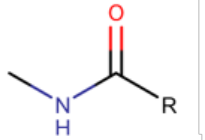
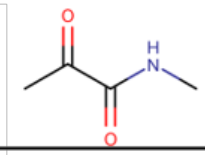
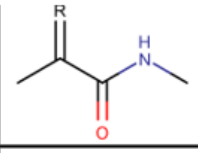
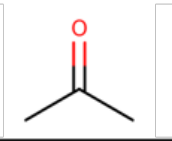
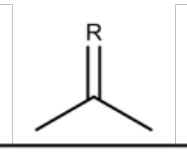
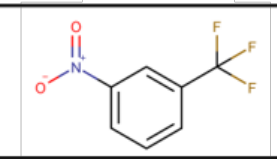
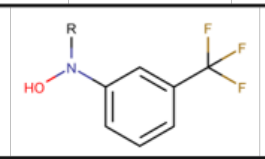
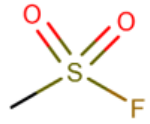
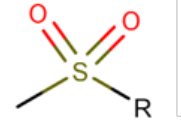
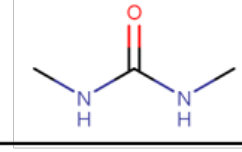
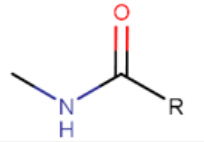
Addition reaction

warhead name	warhead structure	protein-bound state
Aldehyde		
Alkyne		
Bicyclobutane		
Boron		
Cyanamide		
Isothiocyanate		
Ketoalkyne		
Ketoamide		
Ketone		
Nitrile		
Propagylamine		

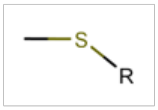
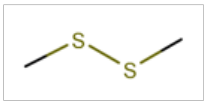
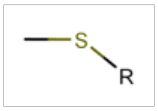
Aromatic substitution

warhead name	warhead structure	protein-bound state
m-HalogenHeterocycle		
o-HalogenHeterocycle		
p-HalogenHeterocycle		
m-Nitroarene		
o-Nitroarene		
p-Nitroarene		

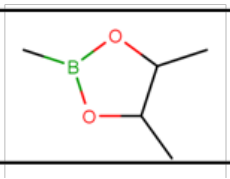
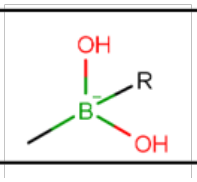
Condensation reaction

warhead name	warhead structure	protein-bound state
Acylimidazole		
Acylpyrazole		
Aldehyde		
Azide		
Carbamate		
Ketoamide		
Ketone		
Nitroarene		
Sulfonylfluoride		
Urea		

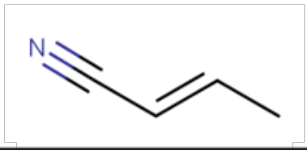
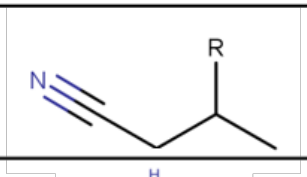
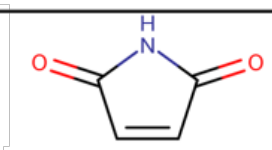
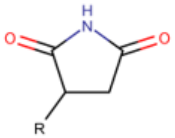
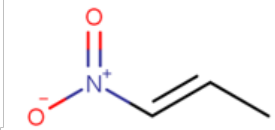
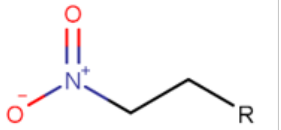
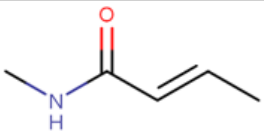
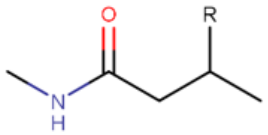
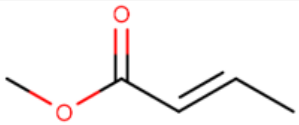
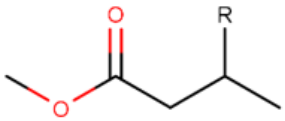
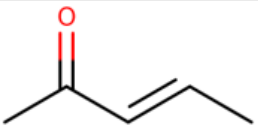
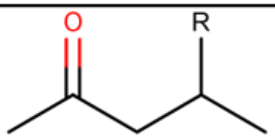
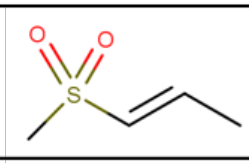
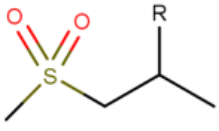
Disulfide formation

warhead name	warhead structure	protein-bound state
Thiole		
Disulfide		

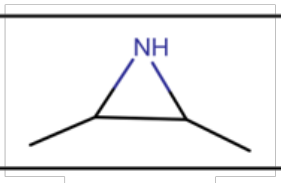
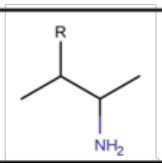
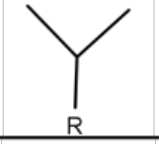
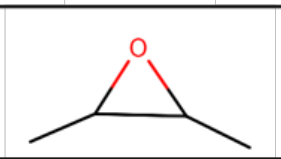
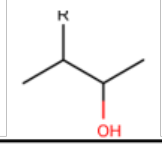
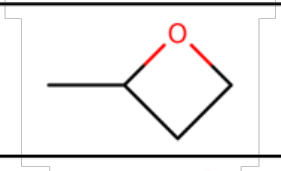
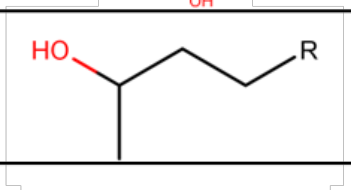
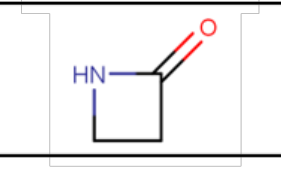
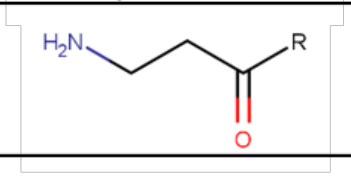
Addition - Elimination

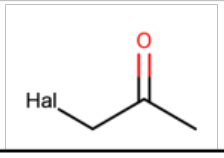
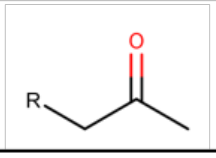
warhead name	warhead structure	protein-bound state
CyclicBoron		

Michael Addition

warhead name	warhead structure	protein-bound state
Acrylonitrile		
Maleimide		
Nitroalkene		
Vinylamide		
Vinylester		
Vinylketone		
Vinylsulfone		

Ring Opening

warhead name	warhead structure	protein-bound state
Aziridine		
Diazirine		
Epoxide		
Oxetane		
Propriolactame		

Substitution

warhead name	warhead structure	protein-bound state
Haloketone