Interactive Quantitative Structure Fragmentation

Introduction

The Interactive Quantitative Structure Fragmentation (IQFS) application is a Web-based tool for dissecting chemical structures into a set of non-overlapping tiles taken from a predefined set of small substructure-based fragments which cover the target structure completely. The identity of the found fragments plus their counts are the basis of a large number of property prediction methods for chemistry. In the most simple algorithms of this class, the fragments are directly associated with a specific group-contribution factor, which is summed up, averaged or otherwise combined to yield a predicted property value for a fragmented compound.

Open Set of Fragmentation Schemes

This program is not limited to a hard-coded, fixed set of fragmentation schemes. Rather, an potentially unlimited set of installed and user-defined fragmentation schemes can be invoked. Schemes are stored in specially formatted SMARTS-based scheme definition files. Adding a new fragmentation scheme to the application is simple: Simply copying a properly formatted schema file into the designated directory is all which is required. The system will automatically adapt to include the additional schemes. Two schemes are already contained in the default installation: The UNIFAC and Sedlbauer-Majer methods.

General Solver Procedure

The program will read chemical structures from various sources and try to find non-overlapping, completely covering tilings of the target structures using only substructures from the selected fragment sets. Depending on the submitted structures, it is both possible to fail to find a solution, and to arrive at more than one solution for a given structure. Failure is usually due to elements or functional groups present in the target structure(s) which are not covered by the fragment set. In some cases, it is also possible to fail to find a solution because the geometrical shape of the fragments does not allow a non-overlapping, completely covering tiling, in spite of all principle functional groups being present in the fragment set. In the case of more elaborate fragment sets, it can occur that more than one solution is feasible. The program will then apply an intelligent filtering mechanism in order to reduce the number of presented solutions.

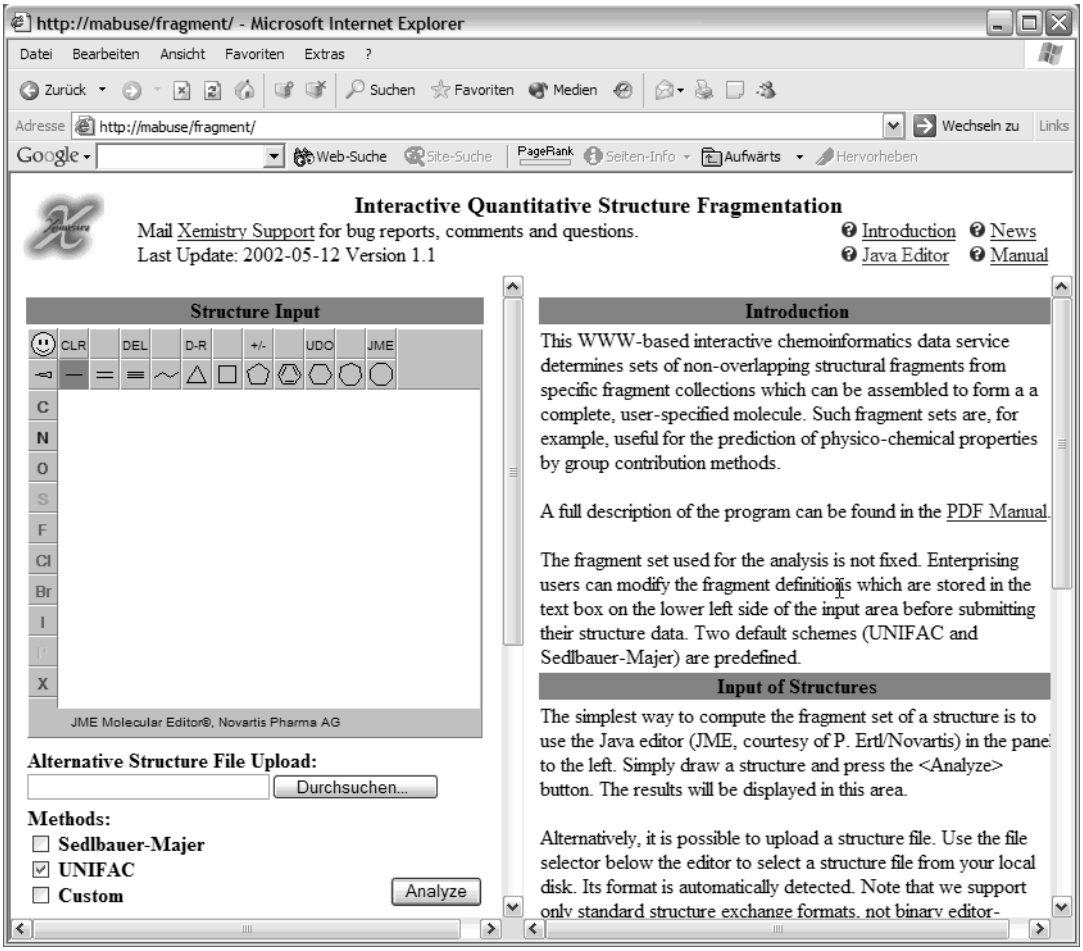
Result Set Filtering

Generally, solutions in which two or more smaller fragments can replace one larger fragment are discarded. Also, multiple solutions which are topologically equivalent (for example, a linear C5 chain represented by a C2 and a C3 fragment versus the inverse C3/C2 solution) are filtered out and presented only once. The remaining solutions are rated - those which use less and larger fragments are favored. Additionally, fragments with extra attributes (such as an explicit condition to match only on rings) receive a bonus relative to the same fragment without extra conditions. The rationale for the rating mechanism is that larger fragments are usually introduced with the specific aim of modelling the divergent behavior of more complex, larger atom groupings. For example, a single, specific COOH fragment is certainly a better solution than a combination of an C=O and an OH fragment for any fragmentation scheme of practical relevance. If the difference of the rating of the best solution to the next alternatives exceeds an empirically tuned threshold, such minor

solutions are withheld. Nevertheless, from time to time different solutions of comparable merit rating can be found. In this case, they are all displayed.

Program Operation Steps

As the first processing step, the solutions for the tiling of structures which were graphically sketched by the user or uploaded as files are presented on a Web page for initial inspection. If the results are satisfactory, they may be exported in different formats for further use in external software applications. The two standard Dupont transfer formats are directly written into text boxes on the Web page for convenient rapid *copy&paste* operation.



Since the fragmentation software is fully Web-based, clients using the program do not need to have any special software installed on their computers in order to be able to access the software. All which is required is a standard Web browser with Java and JavaScript support. Both Internet Explorer and Netscape (plus a variety of less well known alternative browsers) can be used.

Using the Software

After accessing the URL leading to the application, a start page similar to the one shown below is displayed:

The exact URL of the IQFS application can be set during installation. Please inquire of your system administrators about the details and URLs of the local set-up.

On the left side of the start page, a Java-based structure editor forms the upper part of the input zone. Help on various aspects of the program can be obtained by clicking on the help topics in the upper right corner of the Browser window. This manual is provided as an on-line PDF file via the (?) *Manual* button.

Structure Input

There are two methods to present structure data to the software. The first method is the direct, interactive drawing of molecules of interest in the editor. The operation of the editor is fully explained in the on-line manual accessible via the (?) *Java Editor* help button.

The second method is to upload one or more structures via the *File Upload* input field. In order to upload a file, either enter the file name (as seen on the client computer) in the input field, or use the standard file selector box which pops up when one clicks the button to the right of the name name entry field. The software will recognize all standard, non-proprietary structure exchange formats, for example MDL Molfiles, MDL SD-Files and SMILES. It will however **NOT** work with the binary application-specific file formats used by structure editors such as ChemDraw, ACD Draw, or ISIS, and it will not work with simple image files depicting chemical structures. All the standard structure editors provide a method to export the drawn structures as standard MDL Molfile or SMILES. These export files are compatible and may be uploaded to the fragmentation engine.

Multi-Record files are supported as uploaded data source. The maximum number of records in a multi-record file which will be processed as a batch can be configured by the system administrator. The default maximum are 200 records. The processing of larger files stops when the limit has been reached.

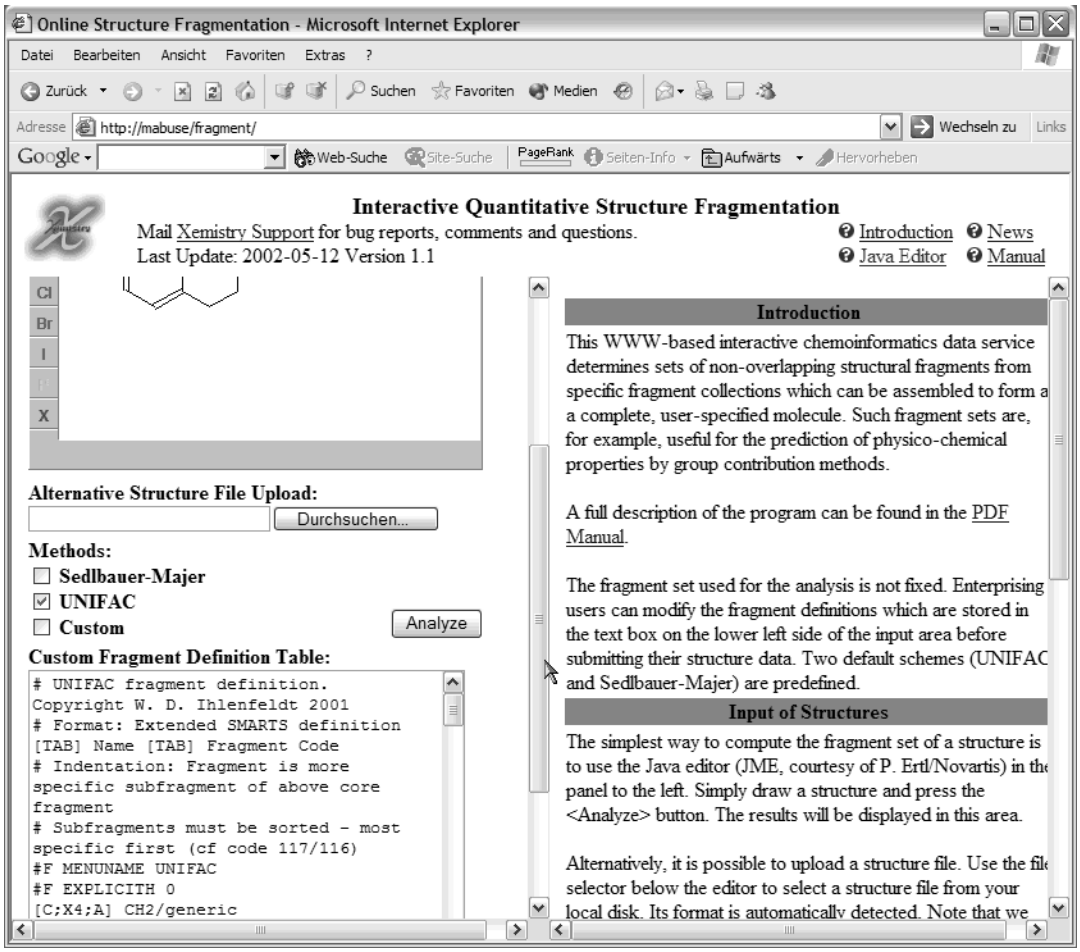
If a multi-record input file contains structures which cannot be solved, these are indicated and processing continues with the next record. Exported tables with the fragmentation results will, depending on the selected format, contain a column with a flag whether the structure was solved or not, or list a zero fragment result.

Switching between File Upload and Interactive Sketching

Note that the presence of a file name on the upload line has always precedence over whatever is drawn in the editor window. So, when a user wants to process structures drawn in the editor after an upload, the file name field needs to be cleared, or the upload file is processed again and the editor content ignored.

Selecting Fragmentation Schemes

The next step is the selection of the fragmentation schemes which should be applied to the input structures. After a little scrolling down in the input field, a menu of the installed fragmentation schemes is presented. This menu is dynamically adapted to show the full set of locally installed schemes and may thus be larger than in this screenshot.



Fragmentation schemes are selected or de-selected by toggling the selection buttons below the *Methods* heading. More than one scheme may be selected simultaneously. The last button labelled *Custom* has a special function: If it is active, the content of the text window below is used as scheme definition.

Custom Scheme Import

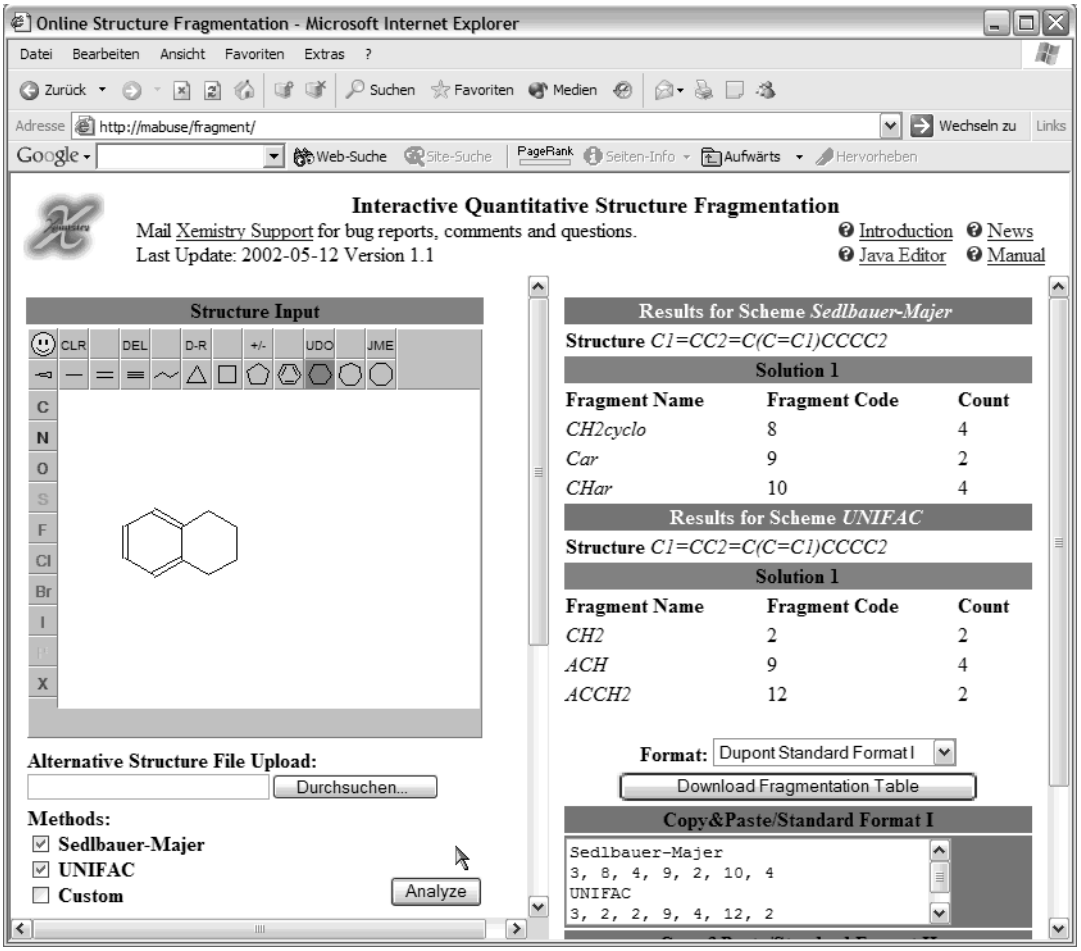
The custom scheme definition window is updated with the definition texts of the standard methods whenever their associated button changes state. So in order to take a look at the source text of the UNIFAC scheme, a user can click on the UNIFAC button. The text of the standard schemes may be copied from the window, and they may be used as a starting point for the development of a custom variant similar to a standard scheme.

It is not required to start with a standard scheme when experimenting with custom methods. Users may also clear the scheme window and paste their private definitions directly into the window. If the *Custom* checkbox is selected, their scheme will be used to process the current structure or upload set.

Note that also deselecting a standard scheme will transfer its definition into the custom definition text area. Please be sure to have completely specified the desired set of schemes before editing a custom definition.

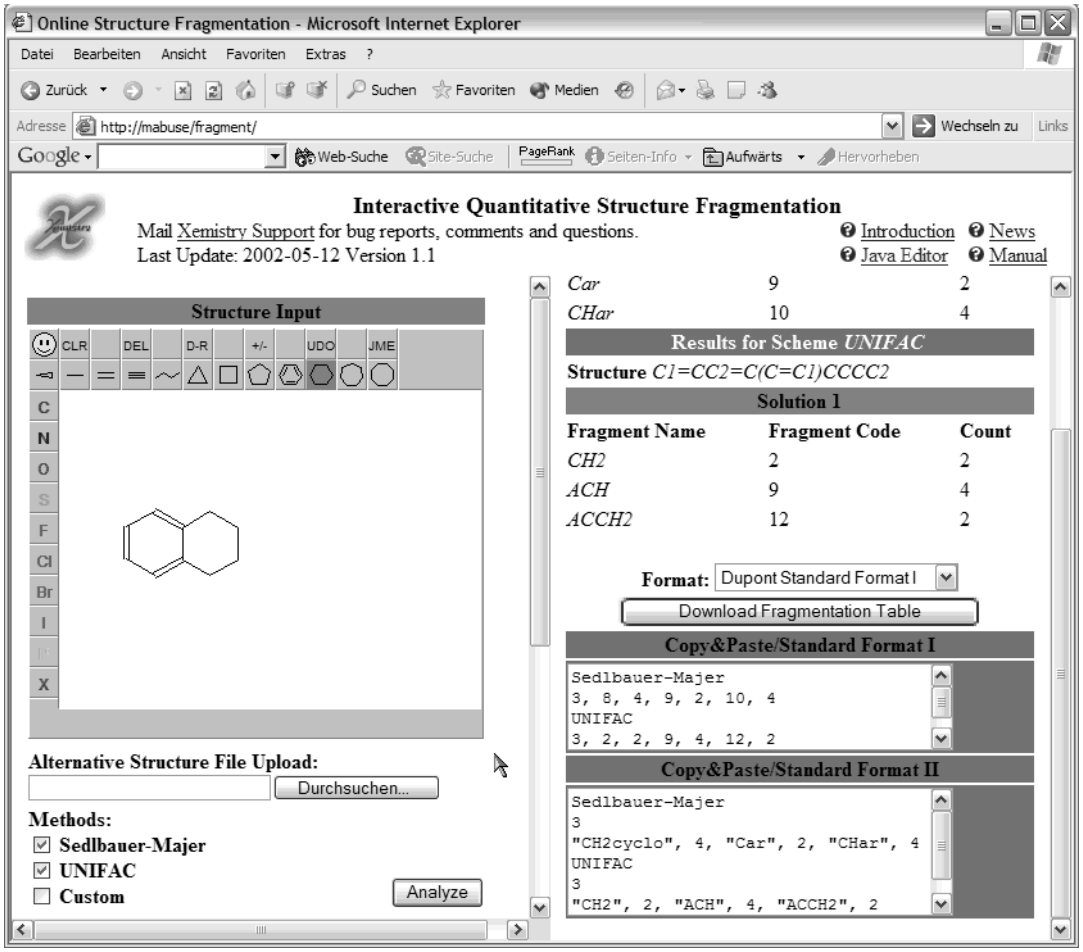
Running the Analysis

After providing structure data and selecting one or more schemes, the *Analyze* button invokes the actual structure analysis. The results will appear in the right half of the window, which initially holds a short introduction to this software.



The image above is a sample output for the analysis of a structure drawn in the Java molecule editor with two methods (Sedlbauer-Majer and UNIFAC). Here, both methods were successful, and both yielded only a single solution each.

In case of a multi-record file upload, the fragmentation schemes will be processed one after the other. The solutions for all structures for the first selected method are output as a block, followed by the blocks of results for additional selected methods.



Export of Results

A small collection of text boxes, menus and buttons is located at the bottom of the results frame.

The purple text boxes contain the fragmentation results in the Dupont standard formats I and II. This data is intended to be selected by the mouse and copied to external programs and files. It is a convenient method for rapid transfer of results of small datasets into other applications.

Larger result sets are exported in a more convenient fashion via a table export. The *Format* menu offers a variety of different table-oriented export formats. After selecting the desired format and pressing the *Download* button, the result set which has been temporarily stored on the server is converted into the selected format and sent again to the user, who will usually store it on disk.

The available table export format choices are dependent on wether only a single fragmentation scheme or more than one scheme has been chosen. Only the Dupont standard formats I and II and the native CACTVS binary table format allow the storage of the results from the application of more than one scheme in a safe and understandable manner. Standard table and spreadsheet formats are dangerously confusing if they contain rows where the meaning of the column data varies from row to row depending on the scheme the numbers were obtained from. Due to these considerations, simple table export formats such as tab- or comma-separated text, SQL or Excel are available only if exactly one fragmentation scheme was used for generating the results.

The intermediate, unformatted results are stored on the server only for a limited time, by default 24 hours. After that, the primary data analysis needs to be rerun in order to be able to retrieve the formatted results.

Custom Fragmentation Schemes

This software uses a very flexible, but not trivial encoding mechanism in order to support arbitrary user-defined fragmentation schemes.

Development of Custom Schemes

User-defined schemes can be developed and tested locally by pasting them into the scheme definition window on the input page. Once they have been tested, the definition files may be copied into the *schemes* directory on the server. All syntactically correct fragmentation schemes which are present in that directory are automatically made available to all users of the software. No further changes in the set-up are required. Upon deletion of fragmentation scheme definition files from the server, their schemes are automatically delisted from the input pages again without need for any changes in the set-up.

Fragmentation scheme definition files are based on simple SMILES/SMARTS definitions. A full description of the SMILES and SMARTS structure encoding standards can be found on the Website of Daylight (www.daylight.com), the original developer of these formats. This software supports the full feature set of Recursive SMARTS for fragment definition, plus a few extensions.

Format of a Definition File

Definition files are simple ASCII text files and may be edited with any text editor, even *notepad*. For the encoding of SMILES fragments, it can be helpful to have access to a standard molecule editor such as ChemDraw which supports the placement of SMILES strings onto the clipboard for subsequent text pasting. Unfortunately, we are not aware of any structure editor which allows the encoding of the full SMARTS feature set by graphical input. Therefore, the developer of custom fragmentation definition files is currently required to understand the SMARTS syntax in detail.

File lines which begin with a hash (#) character or contain only whitespace are considered comment lines and are not read as structure data. They may appear anywhere in the file.

Global File Properties

Among the comment lines, those which begin with the three characters “#F “ (with a blank after the F) are treated in a special way. These lines encode file property data, which describe the file as a data object. The syntax for these lines is “#F *propertyname value*”. The current implementation of this software uses two such file properties.

```
#F MENUNAME Sedlbauer-Majer
```

is a sample definition of the name of a fragmentation scheme. This name is displayed in various places in the generated output pages and files and also used on the method selection page as the name of the scheme. Names may contain blanks. The remainder of the line after the `#F` marker is read, and everything except leading and trailing whitespace is taken as the name of the scheme.

```
#F EXPLICITH 1
```

is an instruction that hydrogen atoms should be treated explicitly. This flag must be set if the fragment set contains explicit hydrogen atoms. In case of the Sedlbauer-Majer scheme, this is true since a dedicated fragment which identifies hydrogen on π-centers exists. Usually, it is more effective to deal with hydrogens in an implicit fashion, i.e. to assume that fragments only need to cover the heavy-atom part of the target structure. The UNIFAC scheme is encoded in this fashion. Note that fragments can still check the hydrogen count of their matched structure counterparts - it means just that hydrogen is not taken into account when not explicitly tested. The default value for this flag is zero. So its specification in the UNIFAC file is strictly speaking redundant.

The file properties must be present before the first SMARTS fragment definition. Every fragmentation definition file must provide a menu name - if it does not, it will be rejected. The explicit hydrogen treatment flag is optional.

Fragment Definitions

The next lines of a definition file are the actual fragment definitions, optionally augmented by comment lines. Fragments are specified as SMARTS strings (see below).

Every fragment definition line must consist of three **tab**-separated parts. The first part is the SMARTS string, followed by the name of the fragment, and terminated by the fragment code. The fragments need not be listed in numerical order. However, the fragment codes must form a complete list starting with 1, without gaps in it. The maximum number of fragments is essentially unlimited, although the processing time may, depending on the similarity of the fragments, increase exponentially with the number of fragments. A few dozen to a few hundred fragments should not pose any performance problems. The use of a **tab** character as field separator is mandatory. It cannot be replaced by spaces. As a benefit of this syntax, fragment names may contain blanks without disrupting the file structure.

A few sample lines from the Sedlbauer-Majer file demonstrate the concepts explained so far:

```
# Fragment definition for the Sedlbauer-Majer group contribution method
# Copyright W. D. Ihlenfeldt 2001
# Format: extended SMARTS definition [TAB] Name [TAB] Fragment Code
#F MENUNAME Sedlbauer-Majer
#F EXPLICITH 1
[C;A;H0;X4;R0] C        1
[CH;A;X4;R0]   CH       2
[CH2;A;X4;R0]  CH2      3
[CH3;A;X4;R0]  CH3      4
```

Fragment Groups

A mechanism for increasing performance and structuring the definition file in a logical way when working with more complex schemes is the use of fragment groups. A fragment group definition is initiated by a standard, rather generic SMARTS substructure definition which must match all the more specific variants represented by the fragments in the group, followed (**tab**-separator) by a generic name for the group. Since this introductory line does not yet define a final fragment, a fragment code is **not** used in this case. By convention, the string */generic* is appended to the group name.

The specific fragments which form a group are entered immediately below the group header, indented with **one tab** character. The format of these lines is otherwise the same as for a simple, direct definition. All indented fragment definition lines following a group header are added to the current group.

When the substructure of a fragment group header matches a portion of a target structure, the match is made specific by checking the group member substructure definitions in exactly the order they were defined below the group header. The first group member which matches is selected as specific representative of the group. Note that this scheme requires that the more specific group members are always listed first if the substructures are encoded in a way where a more generic group member variant matches also substructures which are the domain of a more specialized group member. For example, a definition which requires two certain neighbor atoms around an atom must be listed before a similar definition which requires only one of the neighbor atoms, and that one before the generic version which does not take the neighborhood into account.

Fragment definition files may combine fragment groups for some fragment types and direct definitions for others.

Examples of Fragment Groups

An example from the UNIFAC fragment definition file will demonstrate some of these advanced concepts:

```
COC(=O)OC               Carbonates/generic
       [CH3]OC(=O)O[CH3] (CH3O)2CO    115
       [CH3]OC(=O)O[CH2] CH3OCH2OCO   117
       [CH2]OC(=O)O[CH2] (CH2O)2CO    116
       [CH]OC(=O)O[CH2]  CHOCH2COCO   120
```

This group matches carbonates. A generic version of the carbonate pattern is used in the group definition pattern. It is specialized in the definitions of the group members. Four variants of the carbonate motif are known to the scheme: with two -OCH_3 substituents, one -OCH_3 and one open -OCH_2- part, with two open -OCH_2- parts, and with one -OCH_2- and one -OCH- group. In the hypothetical case that a structure is found where the generic pattern is matched, but none of the specialized instances, an error will be raised.

```
C(Cl)(F)[Cl,F,#1]       CClF/generic
       [CH](Cl)(Cl)F    HCCl2F       88
       C(Cl)(Cl)F       CCl2F        87
       [CH](Cl)(F)F     HClF2        91
       C(Cl)(F)F        CClF2        90
       [CH](Cl)F        HClF         89
```

This is an example of a group which uses an alternative atom list in the group header filter. In this case, the very specific definition of $CHClF_2$ (91) must occur before the definition of -$CClF_2$ (90) with an open valence, since the latter would also match structure 91. Alternatively, fragment 90 could have been coded in a way requiring that there are no hydrogens around the carbon atom, such as [CH0](Cl)(F)F. In that case, the order would have been irrelevant.

```
[CH{1-3}][S;H0]         CH2S/generic
       [CH3]S           CH3S         102
       [CH2]S           CH2S         103
```

```
       [CH]S              CHS           104
C1CNCCO1                  MORPH         105
```

This excerpt shows the use of a simple direct fragment definition without subclasses for the *morpholine* fragment. It is used side by side with the group-based definition of thioethers without any special precautions.

```
[c]1[c][s;X2][c][c]1 Thiophene/generic
       [cH]1[cH][s;X2][cH][cH]1 C4H4S 106
       [c]1[cH][s;X2][cH][cH]1  C4H3S 107
       [cH]1[c][s;X2][cH][cH]1  C4H3S 107
       [c]1[c][s;X2][cH][cH]1   C4H2S 108
       [c]1[cH][s;X2][cH][c]1   C4H2S 108
       [cH]1[c][s;X2][c][cH]1   C4H2S 108
       [cH]1[c][s;X2][cH][c]1   C4H2S 108
```

This last example shows that it is possible to map several topologically different fragments to the same fragment code. Here, the two possible thiophene substructures with three hydrogens on the ring (and one open site, either *alpha* or *beta* to the sulphur atom) are spelled out and assigned the same number 107. The following four lines repeat this procedure for four different forms of thiophene rings with two hydrogens. This approach is necessary since there is no way to specify a global, position-independent hydrogen count in SMARTS.

Substructure Definition Syntax

Substructure fragments are defined in an extended Recursive SMARTS syntax. The full scope of standard Daylight SMARTS is supported. Several minor additional syntactic enhancements were added in order to provide the degree of expressiveness that was required by the fragment schemes which were encoded so far.

The following short introduction into SMARTS will only cover those specification language elements which were actually used in the fragmentation schemes implemented so far. Nevertheless, any syntactically valid SMILES/SMARTS construct even if never mentioned in this manual may be used if desired. Please refer to the Daylight website (www.daylight.com) for additional documentation on the full capabilities of SMARTS.

A Short SMARTS Course

A SMARTS specification is a single-word string which does not contain any white space.

The most important building blocks for a SMARTS substructure are atoms and bonds. Atoms are encoded either by their element symbol, or a hash character followed immediately by the periodic system sequence number. So, ’C’ is carbon, and ’#1’ is hydrogen. Elements in the organic subset (B,C,N,O,P,S,F,Cl,Br,I) may be simply written as their element symbol, if no other attributes are specified. All other elements must be enclosed in square brackets, i.e. ’[Se]’. For elements from the organic subset, bracketing is optional if no additional attributes are set, but if additional options are specified, the atom expression must be enclosed in brackets. Case is important because lowercase first characters of element symbols convey a special meaning (see below).

Hydrogen is a special case. The symbol H does not represent an hydrogen atom, but a hydrogen count. An explicit hydrogen atom therefore must be expressed as ’[#1]’. An element symbol may

be directly followed by a hydrogen count, if the atom is bracketed, as in [NH2]. If no explicit count is entered after the H, its default value is one.

Explicit hydrogen atoms and those following *immediately*, without any separator, after an element symbol are those which are tracked in the tiling process when using explicit hydrogen fragmentation schemes. So, a hypothetical fragment set consisting only of the fragments [CH3] and [CH2] could be used to find a solution for ethane and propane, but not for methane or isobutane.

Note that this special convention for the treatment of directly linked hydrogen is a program-specific SMARTS extension. In this software, *when processing a scheme with explicit hydrogens*, the precise meaning of [CH3] and [C;H3], which is identical in standard SMARTS, is not equivalent. The first fragment directly contributes three hydrogen atoms to the tiling. The second fragment just requests three neighbor hydrogen atoms for a match, but does only provide a single carbon atom as tile. For a full coverage of the target structure, which is required for a valid solution with explicit hydrogens, these hydrogen atoms must then be provided by other fragments such as [#1] if the second carbon fragment is matched anywhere.

Bond and Ring Specification

Bonds between atoms are expressed by the characters -, = and # for bond orders between one and three. A bond order of one is the default and the - character may thus be omitted. So, CC is just another fully equivalent encoding of C-C (ethane), and CC=CC represents 2-butene.

Branches in the structure tree are encoded by round parentheses. All atoms and bonds in the section extending up to the complementary closing parenthesis comprise a branch. The next simple atom or further branch segment following after the closing parenthesis is linked again to the atom to which the first atom of previous branch was attached - not to any atom of the closed branch. Branches may be nested to an arbitrary depth. Examples: CC(C)C(=O)O is a hydrogen-depleted fragment for isobutyric acid, and C(F)(F)F is an CF_3 fragment. The terminal O of the first fragment is attached via a single bond to the C to which also the just closed (=O) branch was bonded, forming a carboxyl group.

The order of the atoms in the SMARTS string is not important. This software uses a graph-matching procedure to determine the possible matches of any fragment. So, instead of above CF_3 definition an alternative representation FC(F)F could be used and would yield exactly the same results.

Ring closures are marked by digits. When the SMARTS string is interpreted from left to right, every matching pair of digits introduces a ring bond. After the closing digit has been encountered, it may be re-used for another ring closure atom pair. So, C1CC1 is cyclopropane. Ring closures can be combined with bond orders, so C1CCC=1 is one possible encoding of cyclobutene.

Aromaticity

One method to specify an aromatic ring is to simply provide a Kekulé representation. C1=CC=CC=C1 is thus a valid phenyl fragment. Its aromaticity is automatically detected, and when the bonds are matched, the bond order of the aromatic rings is not directly compared, just their aromaticity status. Aromatic bonds in fragments which do not form a full ring (a prerequisite for automatic aromaticity detection) can be specified with the ':' character. Such aromatic bonds in a fragment will only match aromatic bonds in the target structures.

The aromaticity of atoms may be explicitly declared by using a lowercase first letter in their element symbol. Also, such atoms do not need alternating ring bonds or explicit aromatic bonds for bond classification in rings. c1ccccc1 is thus another and shorter way of defining a phenyl fragment. Atoms written with an uppercase first letter will, if no additional attributes are specified, match both aromatic and aliphatic atoms in the target structure.

Atom Attributes

Atoms (and bonds, but this is not discussed here) can be tagged with additional match conditions which make them more selective for matching purposes. A simple example for an additional match condition is the hydrogen count, which was already mentioned above. Additional match attributes can be combined by the logical operators ',' (*or*, medium precedence), '&' (*and*, high precedence), ';' (*and*, low precedence), '!' (*not*, very high precedence). If no logical operator is specified, the default linking of the match attributes is a strong *and* ('&'). So, the exact meaning of [CH3] is 'carbon *and* three neighbor hydrogens', and [Cl,Br,I] is a list of atom alternatives "chlorine *or* bromine *or* iodine".

Important attributes are R (ring membership count), a (aromatic), A (aliphatic), * (any atom wildcard), v (valence) and X (connectivity). Standard SMARTS defines more then this small list of attributes, providing for example also match attributes for charges, chirality and isotope labelling. Please refer to the Daylight documentation if you need to use these attributes in custom fragmentation schemes - they are all supported, but have not been used in any schemes which were encoded so far.

Attribute Counts

Some of the attributes have default counts associated with them: R: atom is member in one or more rings, a: atom is in one or more aromatic systems, H: atom has exactly one hydrogen neighbor. The attributes a, H and R *may*, and the attributes v and X *must* be followed by a count. So [C;X4] matches any carbon atom with four neighbors, and [C;R;X3] is a carbon atom which is a member in at least one ring (of the SSSR set[1]) and has three neighbors (implying a double bond if the possibility of atomic charges is excluded), and [!C;a] is an aromatic hetero atom.

It can be useful to use 0 as an attribute count. For example, [CH0R0] is a carbon atom explicitly without hydrogen neighbors and not a ring atom, while a simple [C] does not require any specific hydrogen neighborhood or ring membership for a match. The reader should also be aware that there is no requirement to use any atom symbol at all in an atom expression. [X4] will match any atom with four neighbors, and [A] any aliphatic atom.

SMARTS Syntax Extensions and Match Condition Modifications

In the sample fragmentation schemes, you will notice that many atoms are qualified as explicitly aliphatic, such as in [CH2;A;X4;R]. This is required because the structure match engine of this program runs in a mode which will match atoms which are not explicitly aliphatic both onto aromatic and aliphatic targets. In contrast to standard SMARTS, a qualification with the A attribute is required to limit the match to aliphatic targets. Writing the atom symbol with an uppercase initial letter is **not** sufficient.

1. The Smallest Set of Smallest Rings, a ring set which excludes envelope rings. For example, in naphthalene, the SSSR contains two six-membered rings, but not the 10-membered envelope ring.

In an extension to standard SMARTS, the a attribute may also be qualified with a count which indicates the number of SSSR aromatic rings the atom participates in. For example, only the two central carbon atoms in naphthalene match an *a2* condition.

Another convenient extension are ranges which can be used anywhere where a simple count is possible. Instead of writing [C;H1,H2,H3] (carbon with one to three hydrogens, note the use of the lower precedence *and* operator - [C&H1,H2,H3] means a *carbon with one hydrogen*, or *any atom* with two or three hydrogens) this expression can be abbreviated as [CH{1-3}]. Similarly, [c;a{2-}] is the definition for the *ccond* (carbon in condensed aromatic rings) fragment in the Sedlbauer-Majer scheme. This match condition cannot be expressed as standard SMARTS. Ranges can be fully qualified as in {1-3} or open on the left or right side as in {2-} and {-3}. In these cases, the range implicitly begins at 0, or extends to infinity, respectively.

Recursive SMARTS

The sum of all matching atoms of a fragment substructure defines the tile which is used to cover a part of the target structure. The special procedures for dealing with hydrogens explicitly or implicitly have already been discussed. However, the tiling mechanism by direct extraction of the matched atoms has a notable disadvantage: It does not allow the check of non-participating neighborhood atoms by structural characteristics. Fragment definitions such as “hydrogen atom bonded to π system” (without including the π system atoms in the fragment) or “chlorine attached to aliphatic C=C” (without using the carbon atoms in the fragment tile) are not possible with the syntax discussed so far.

The solution for this problem is the use of Recursive SMARTS. A recursive SMARTS element is an arbitrary SMARTS expression enclosed by $(..). The part inside the parentheses may again contain Recursive SMARTS components. The recursive SMARTS elements must match their parts of the target structure to allow the overall global fragment match to succeed, but its matching atoms are not considered part of the fragment tile covering the target structure. Also, the atoms and bonds in the Recursive SMARTS components do not interact with the rest of the substructure, so they may match atoms and bonds which are already matched by other fragment parts. However, if multiple Recursive SMARTS components are attached to a common atom, they must all begin their match on different neighbors of that atom.

Recursive SMARTS Examples

So, the above mentioned chlorine atom bonded to an aliphatic C=C part can be encoded as Cl[$([C;A]=[C;A])]. The chlorine atom outside the Recursive SMARTS part is the only atom in the final tile, so this tile will cover only a single chlorine atom in the target structure - which however must be located in a specific neighborhood. The Recursive SMARTS definition is encoded as a normal neighbor atom bonded to the chlorine (in [] brackets). Within the pseudo atom brackets, it is defined to be an aliphatic carbon bonded to another aliphatic carbon. The first carbon atom will match an atom in the direct neighbor sphere of the chlorine atom, and the second atom will be one more bond distant. Neither of these carbon atoms are part of the final tile.

A more complex definition used in the Sedlbauer-Majer scheme is a “hydrogen atom bonded to a π system which can be either a simple C=C substructure, or a derivative of formic acid”. A simple implementation of this constraint is [#1][$([C;A]=[C;A]),$([C;A](=O)[O,N])]. Here, the Recursive SMARTS part is encoded in the form of two alternative branches (separated by the comma *or*

operator). The first alternative is a simple aliphatic C=C group. The second alternative is an aliphatic carbon atom with a double bond to an oxygen atom, and a single bond to either oxygen or nitrogen. This definition covers formic acid, formic acid esters and formic acid amides - but not yet, for example, formic acid chlorides.

The definition of a carbonyl fragment which does not have any aromatic neighbors is a little bit more complicated than it seems at first. The straightforward approach `O=C[$(!a)]` is not sufficient. This construct will not match benzophenone as required, but will still match acetophenone, since the aliphatic atom can be assigned to the methyl group. One possible improvement is to use `O=C([$(A)])[$(A)]` instead. Here, both neighbors (assuming that the valence of carbon is always four in the datasets of interest) must be aliphatic. Multiple Recursive SMARTS definitions around a common atom must match their first atom onto different neighbor atoms of the joint central atom, and thus the fragment is not found present in acetophenone any longer, but acetone still matches as intended.

Recursive SMARTS is also an important tool for the specification of bonding environments. For example, the definitions [CH0;A;X2;R0]=[$(*)] and [CH0;A;X2;R0]#[$(*)] can be used to distinguish between an aliphatic, non-ring carbon atom in the middle of an allene-type system (first example, assuming that having two neighbors and one double bond to one of the neighbors implies that the second bond is also a double bond) and a carbon atom which is bonded via one triple bond and one single bond to its neighbors. These fragments both define only a single atom - the dummy neighbor atoms which are matched via the Recursive SMARTS part do not count for the tiling - but the bonds which link them must still match the structure which is being processed.

Stereochemistry

The program is fully aware of standard stereochemistry (cis/trans double bonds, including those which involve free electron pairs, tetrahedral stereochemistry, also with free electrons, and allenes). Fragments with stereochemical descriptors will only match structures with corresponding stereo features, and neither match the opposite stereochemistry, nor undefined stereochemistry. Fragments without stereo features will match corresponding groups with and without stereochemistry in the target structures. In case both a stereo-specific fragment and an unspecific fragment match onto the same target atoms, the stereo version has precedence.

An Example:

```
[#1]            -H         1
C(/I)=C(/I)     -trans-I2  2
C(/I)=C(\I)     -cis-I2    3
C(I)=C(I)       -undef-I2  4
```

Using this fragment set, cis-diiodoethene will me tiled with fragment 3 and the trans-component with fragment 4. In case no stereochemistry is present, fragment 4 will be used. Since internally topological stereo descriptors are used, and no 2D atom locations, it is not required to specify fragments in alternative drawing styles. So, the *cis* fragment may be either specified with two *up* bonds, or with *two* down bonds. For processing purposes, these forms are fully equivalent.

Note that the Java editor always generates compounds with defined double bond stereochemistry. The presence of atom stereochemistry in editor structures is determined by the presence or absence of wedge bonds. The rules to check whether an uploaded file contains stereochemical information or not are more complex. Generally, if the file contains 2D- or 3D-coordinates,

explicit stereo descriptors for atoms or bonds, or wedge bonds, the file is assumed to contain stereo information, and the internal stereo descriptors are derived from whatever information is available. Explicitly undefined stereogenic double bonds can be encoded by a ligand with an angle to the double bond close to 180 degrees, or by *wavy-bond* attributes. It is legal to have defined and undefined stereo centers, double bonds and allene systems in the same structure.

Bond-Only Fragments

Some fragmentation schemes contain factors which are bond-specific and do not contain any atom reference. In the simplest case, standard fragments without bonding information are used to tile the core framework of the structure, and extra correction terms are added for double and triple bonds.

A simple fragment set of above type for alkanes and alkenes could for example be specified as

```
[CH3}              CH3       1
[CH2]              CH2       2
[CH]               CH        3
[$(*)]=[$(*)]      DB        4
```

Fragments which consist only of two recursive atom fragments which are both of type *any* and have a bond between them (defining a simple bond order, or possibly a more complex SMARTS bond specification) are treated specially in the fragmentation algorithm. These fragments are not used for normal tiling. Rather, their counts are established after a basic solution has been achieved.

Both ends of such bond fragments must be specified as Recursive SMARTS dummy atoms. Encodings such as [*]=[*] or [$([*]=[*])] are **not** equivalent.

Note that no warning is produced when bond types not covered by extra bond-only fragments are encountered - such as a triple bond in above example.

The Single-Atom Dummy Fragment

An isolated dummy fragment atom [*] is treated slightly different from a normal one-atom fragments. This fragment has implicitly an infinitely low merit value and will only be used when no other alternatives are available to cover an atom. It will never interfere with the matching of other, more specific fragments. On the other hand, it will not cover any atom which could be matched by other fragments, but for which due to geometrical constraints no solution could be found. Such compounds raise an error.

So a simple scheme

```
[#1]               H         1
[C,c]              C         2
[N,n]              N         3
[O,o]              O         4
[*]                X         5
```

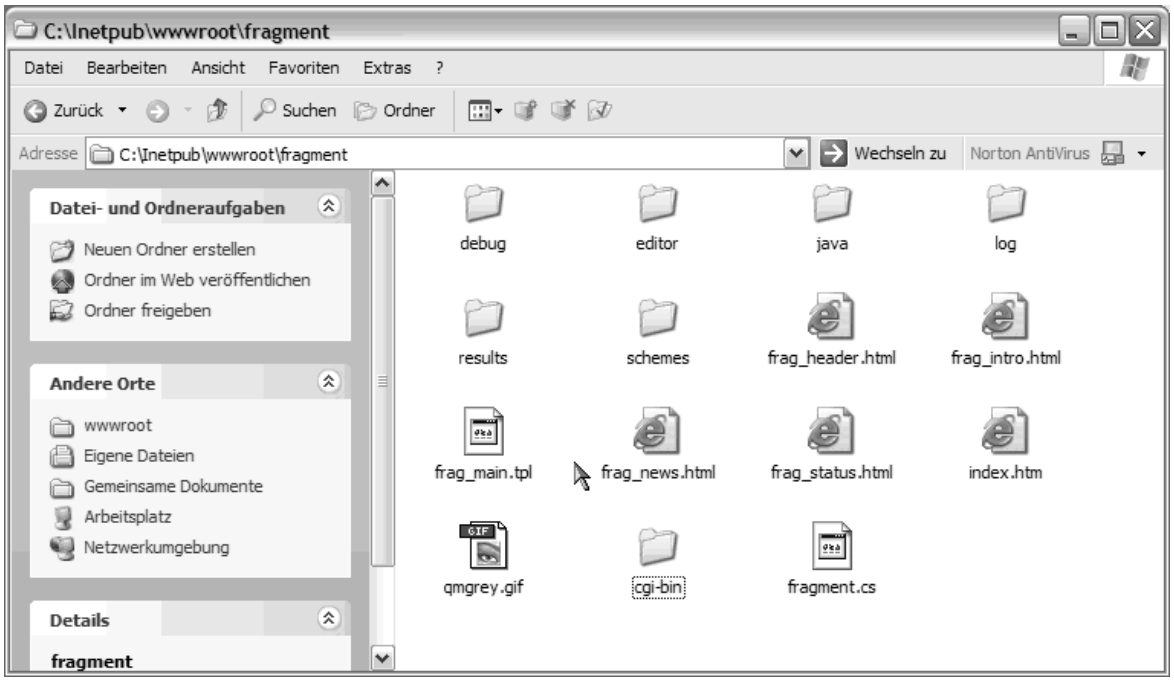
will cover every structure imaginable, replacing all undefined elements with fragment 5.

Program Installation

Installation for MS IIS 5

The program and all supporting components are delivered as a single zip file. The software can be installed anywhere in the directory hierarchy of the Web server. The internal references and links within the software are relative and do not need to be adapted. It is strongly suggested that the program is installed in a dedicated directory. In this installation guide, the sample directory *fragment* on the server root is used.

After unzipping in a suitable server directory, a set of files similar to the one displayed below will be present:



Directory Structure

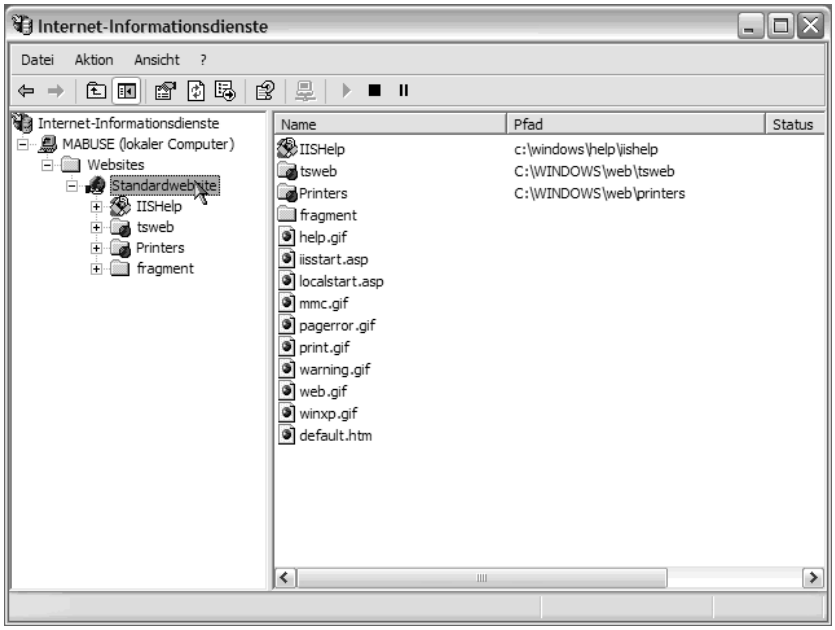
The function of the most important directories and files is as follows:

Debug:	The directory into which a debug trace is written if the debug flag is set. Normally, this directory is empty. It needs to be writable by IIS if debugging is enabled.
Editor:	Help and data files for the Java molecule editor which is distributed with the system.
Java:	Java classes of the molecule editor.
Log:	This directory contains a compact log with the submitted structures and their analysis results if the logging script option is active. Only a few lines are added for each processed structure. Nevertheless, the log file should be

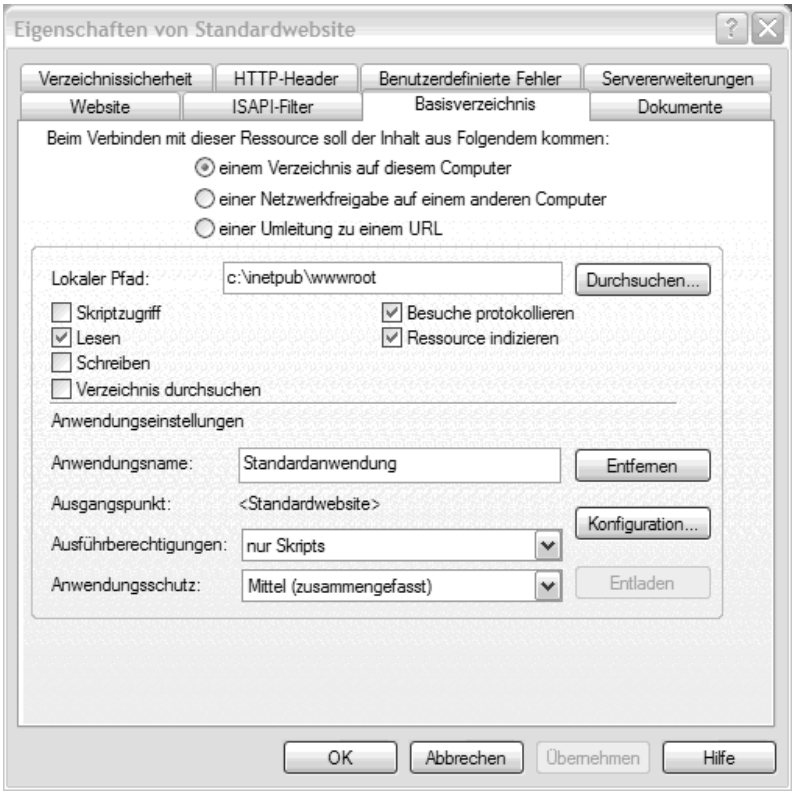
	periodically deleted if it has grown too big. This directory needs to be writable by IIS if logging is active.
Results:	Temporary storage area for results. Intermediate files which contain the raw results are kept here. This directory needs to be writable by IIS. The maintenance of the intermediate result directory is automatic. Files which are outdated (older than 24 hours) are automatically removed by later runs of the application.
Schemes:	A directory to store the files which encode the definitions of the various fragmentation schemes offered by the software. All files stored here are expected to contain a fragmentation scheme definition as described in this manual. The directory contents are scanned by a template expansion routine of the main script. It will automatically advertise on the input page all methods contained in the files placed here. IIS needs only read access to the data in this directory, but authorized expert users may wish to be able to deposit their own schemes here. The suffix for the fragment definition files is *.smi* for SMILES, a standard format for chemical structure data. These are simple text files, **not** *Synchronized Media Integration Language (SMIL)* files. Do not attempt to execute these by multimedia player software such as RealOne or the Windows Media Player even if the file icon seems to imply that these programs are the standard openers.
Cgi-bin:	This directory contains the main interpreter *csweb.exe* which drives the application.
index.htm:	The start page, which is just a frame set.
frag_main.tpl:	The template for the main input page. It will be dynamically modified by the application script to allow the automatic inclusion of additional fragmentation schemes and software installation path independence.
fragment.cs:	The top-level application script for the Web-based service.
batch.cs:	Application script for simple batch processing.

Script Interpreter Association

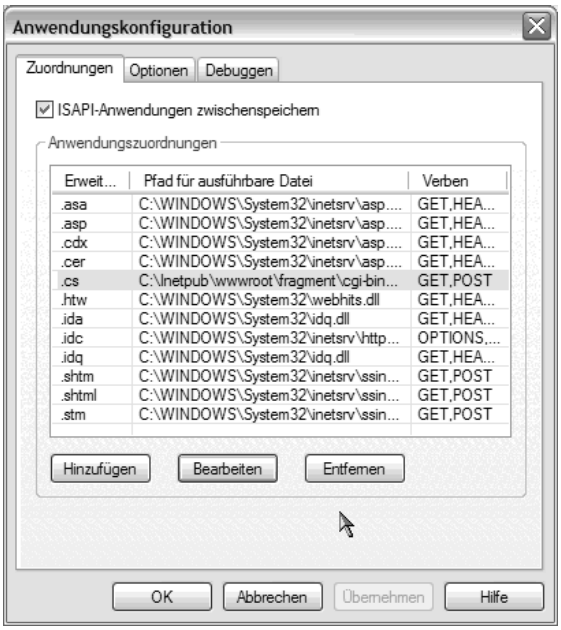
The file *fragment.cs* is the main script file which drives the application. This script file is interpreted by the general-purpose chemical structure processing system *csweb.exe*, which is located in the *cgi-bin* directory. IIS needs to be told that files ending in *.cs* are to be interpreted by *csweb.exe*. This is done by means of the IIS administration console *InetMgr*.



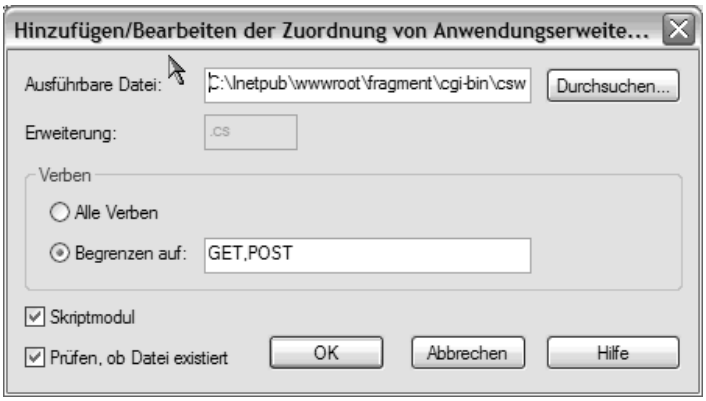
After selecting the *Properties* menu on the standard Website (or the *Fragment* directory), the following panel is displayed:



Clicking on the *Configuration...* button opens the file name suffix association panel:



The file suffix *.cs* needs to be associated to the *csweb.exe* application:



In order to set this up, enter the suffix *.cs* in the extension input field and the full path name for *csweb.exe* into the executable name panel.

IMPORTANT: After selecting *csweb.exe* as executable, one must add, after the path to the program, the string ' -f %s %s' (without the quotes). The full line (which most likely is too long to fit into the input field without scrolling) should look similar to

C:\Inetpub\wwwroot\fragment\csweb.exe -f %s %s

Site-Specific Script Editing

The application script *fragment.cs* is an editable text file. It can be opened and edited by any text editor, such as *notepad.exe*. For the standard configuration of the system, only the base directory

as seen by the CGI application (not as seen by the client) must be configured in this script. This is done be editing line 6

set BASEDIR /inetpub/wwwroot/fragment

to let the variable BASEDIR point to the local installation directory, if is it not set to the pre-configured default. Directories in the path need to be separated by '/', not backslashes.

Now the *index.htm* main page of the fragmentation service can be accessed for the first time. If everything went right, the input form with the Java molecule editor and the dynamically updated list of available fragmentation modules should come up. The input page template is already processed by the script interpreter, so if it is not set up correctly, the start page will fail to show up correctly.

The script contains in the first further few lines a number of additional, commented variables which control time-outs, maximum number of compounds processed in a batch, etc. These may be changed according to the local needs.

Client Computer Configuration

The demands on client computers which access this software system are modest. The application is browser-neutral, both Internet Explorer and Netscape may be used. Furthermore, the client platform (PC, Mac, Unix workstation) does not matter, as long as a standard Web browser with Java and JavaScript support can be run.

In order to be able to use the integrated Java-based molecule editor for interactive input of chemical structures for processing, both Java and JavaScript must be enabled. The version of the Java virtual machine is not critical, both Microsoft and Sun implementations will work.

Batch Processing

The normal mode to operate the software is via the Web interface. However, the core functionality of the software is also accessible as a DOS application for batch processing and testing. The same chemical data processing interpreter *csweb.exe* is used, but instead of interpreting the WWW-enabled *fragment.cs* application script, it is run with the simpler *batchfrag.cs* script.

DOS Application Invocation

The batch version of the software is started as

```
csweb -f batchfrag.cs -- fragfile inputfile format >outfile
```

All arguments representing file references (*csweb*, *batchfrag.cs*, *fragfile*, *inputfile*) must be specified with absolute or relative directory paths if they are not present in the current directory, or, for *csweb* only, in the executable search path.

The arguments are:

batchfrag.cs	The simple batch processing script (found in the base directory of the software installation)
fragfile	The name of a fragment definition file. Two sample definition files for the UNIFAC and Sedlbauer-Majer schemes are part of the standard distribution and stored in the *Schemes* directory.

inputfile	A single- or multi-record input file in a standard chemistry exchange format such as SMILES or SD-file.
format	The desired result table output format. In the batch version, this must be one of *dupont1*, *dupont2*, *excel*, *sylk*, or *tab*.

The results are written to the standard output channel and may be redirected into a file with the > mechanism.