The InfoChem

Fast Search Engine

(ICFSE)

V.2.10

InfoChem GmbH

Version 03/2016

InfoChem GmbH
Dr. Troll-Str. 14
82194 Gröbenzell
Tel: + 89 58 30 02
Fax:+ 89 58 03 839

InfoChem GmbH
Landsberger Str. 408
81241 München
Tel: + 89 58 30 02
Fax:+ 89 58 03 839

General Description

The InfoChem Fast Search Engine (ICFSE) offers the functionality for fast creating, updating and searching chemical structures and reactions.

Based on the open technical concept of InfoChem ICFSE can be integrated in various platforms and in software environments like web application and data base systems.
For example the ICFSE is the structure and reaction search engine for the web based chemistry database SpresiWeb and dealing with more than 8 million structures and 4 million reactions.
The ICFSE is also integrated in the InfoChem Chemistry Cartridge for Oracle IC*Cartridge* and provides the high performance chemistry search. In future concepts alternative database systems can be extended by chemical structure and reaction handling using the external ICFSE service.

For proprietary developments at the customer site a convenient Java component is provided allowing to control ICFSE completely from Java.
The Java Wrapper was successfully used to integrate the chemistry search functionality of ICFSE into the text search system *Microsoft FAST Search Server 2010* in SharePoint environment.

Technical Background

ICFSE is a background service listening to incoming TCP/IP requests, performing the requested tasks in optimized search indices and resending the results.

The communication to ICFSE is based on TCP/IP sockets, the exchange format is XML, see samples below.

The Java wrapper implements an interface to allow a convenient access to ICFSE. See examples below.

The current implementations use C++ ANSII standard. Sockets and multithreading techniques are restricted to Windows platforms and operating systems based on POSIX standard.
Currently available POSIX Implementations: Sun-Solaris, Linux.

ICFSE is fully scaleable. The service can be operated on distributed hardware (addressed via different IPs) and large indices can be split for parallel search without limitations.

For an overview of the technical concept see illustration on the next page.

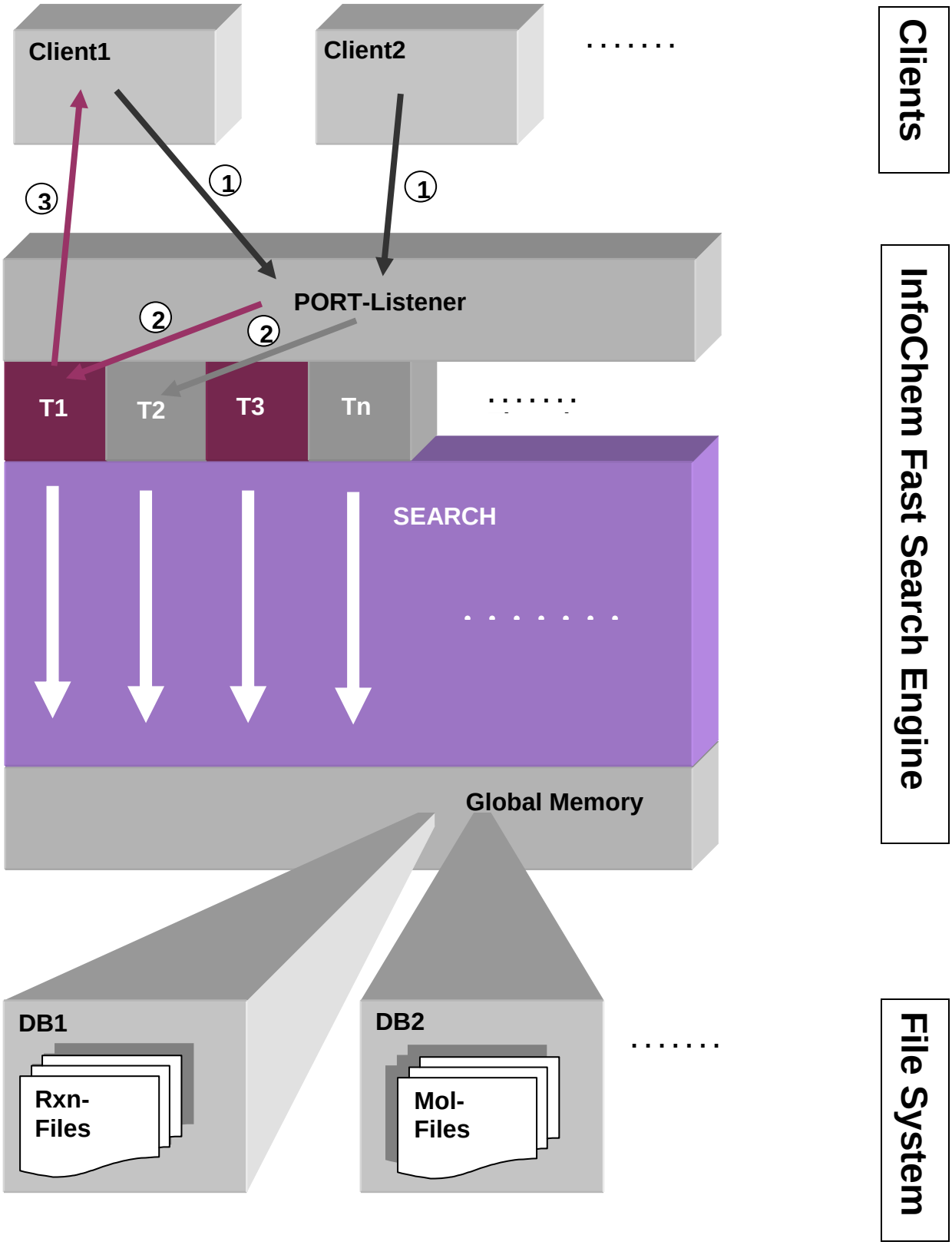


Figure 1: Technical concept ICFSE

ICFSE Integration

ICFSE is integrated in the InfoChem SPRESI Web application (http://www.spresi.de) as read-only search service and in the InfoChem Chemistry Cartridge for Oracle with read/write functionality. For illustration the integration in the Cartridge is shown below.

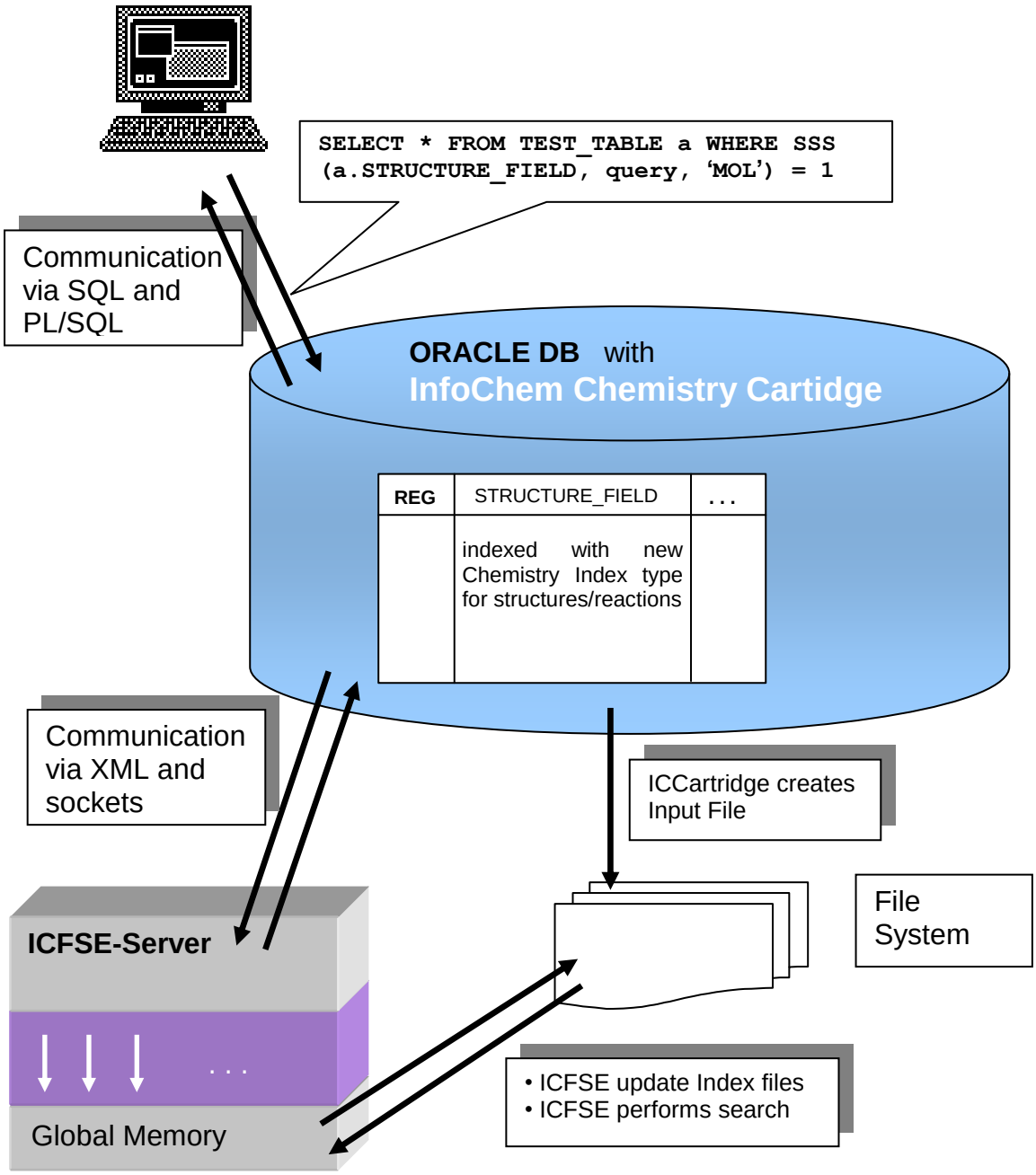


Figure 2: Integration concept ICFSE in Oracle Chemistry Cartridge ICCartridge

ICFSE Functionality

- Chemical search operation:

Reactions:

- **RSA** (“All-in-one” reaction search)
- **RSS** (reaction substructure search)
- **XRS**, **XRSSUB**, **XRSRCT**, **XRSPRD**, **IXRS**, **IXRSSUB, IXRSRCT**, **IXRSPRD** (several exact reaction search operators, with possibility to include isotopes or subsets of the reaction in the exact search)
- **RCTNOTPRD**, **PRDNOTRCT** (Reaction role search operators for reactants and products)
- **RSS_RT_[SPHERE]** (Combined RSS/RTS Search)
- **RSS_RT_ALL** (Combined RSS/RTS Search for all three spheres)
- **RTS_[SPHERE]** (Reaction Type Search Broad/Medium/Narrow)
- **RTS_ALL** (Reaction Type Search for all three spheres)
- **RXCCLUSTER_[SPHERE]** (Reaction Center Cluster Search)

Structures:

- **SSA** (“All-in-one” reaction search)
- **SSS** (substructure search)
- **ISSS** (inverse substructure search)
- **XSSEZ**, **XSS**, **RXSS**, **IXSS**, **TXSS**, **PXSS**, **CompXSS**, **FlexXSS** (several exact structure search operators, with possibility to include isotopes, tautomeres, salts, etc. in the exact search)
- **MSIMFP** (similarity search)
- **SFS**, **XSFS** (sum formula and exact sum formula search)

- Update functionality**:**

NEW	Create a new fast search index. Input formats: • MDL MOL/SD file for molecules, RXN/RD File for reactions. • InfoChem Mol/Rxn Rosdal Files. • SMILES Files. The structure/reaction record identifier must be delivered with every single structure/reaction record.
DELETE	Delete fast search index.
APPEND	Append records to a fast search index. This command includes the functionality of deleting individual records. High performance update for large databases: 1,000 new records to a 4 Mio. database: 10 Min.
EDIT	Implemented as delete individual record and append new updated record.
RENAME	Rename a fast search index.
FEED	Append single Records for Java Wrapper.

- Normalization and registration of chemical structures.
- Calculation of unique structure Identifiers (HashCodes).
- Highlighting of substructure in given structure.

Data and query input

Structure formats:	MDL standard formats: Input from SDfile, RDFile, Molfile, RxnFile and InfoChem specific Rosdal format, SMILES Format Next version: InChI
Query features:	all MDL query features like any atom, list, any bond etc. supported Additional Search operators are available (e.g. RSS_RT) and additional Query features under development

System Requirements

Hardware:

Windows/Linux:	Pentium III or higher; > 1 GHz recommended
SUN Solaris:	SPARC > 700 MHz recommended

up to one million structures:	512 MB additional RAM
for every further million structures:	512 MB additional RAM
up to one million reactions:	800 MB additional RAM
for every further million reactions:	800 MB additional RAM
up to one million structures/reactions:	4 GB free disk space
for every further million structures/reactions:	4 GB additional disk space

Software:

Operating system:

Windows 2000
Windows XP
Windows Server 2003
Windows Server 2008
SUN Solaris 8 or higher, 32 / 64 Bit Version
Linux

Performance

System for reaction and structure search performance tests:

ICFSE in ICCartridge environment

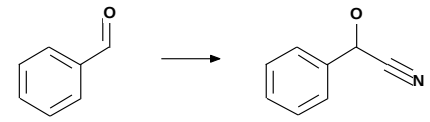
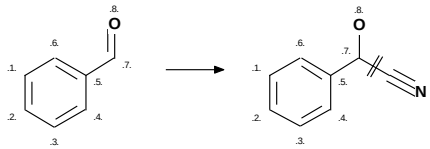
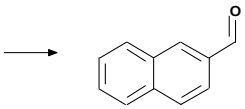
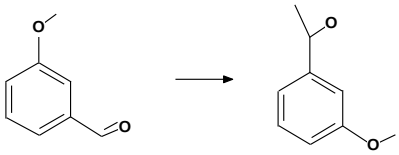
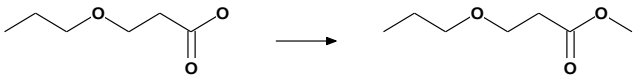
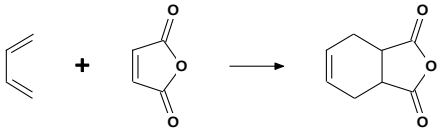
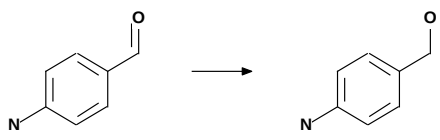
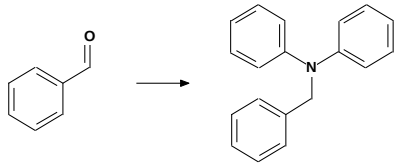
AMD Opteron processor 146, 2.01 GHz, 4.00 GB RAM

Windows 2003 Server

Oracle 10.1.0.4.0

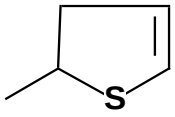
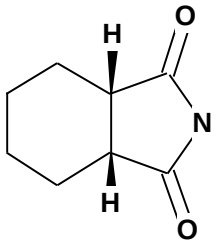
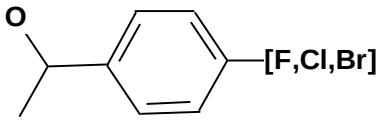
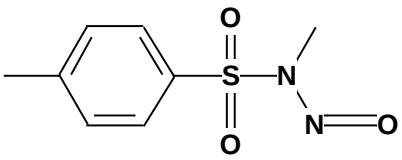
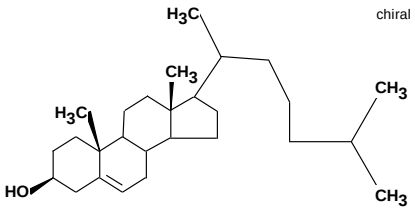
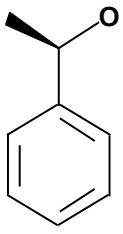
Reactions

Data set: Table with 1.061.207 records containing reactions (using ICCartridge reaction index type)

query	hits	time [sec]
	360	1.7
	223	1.2
	4334	7.8
	889	0.8
	656	0.4
	374	0.2
	4436	2.1
	187	4.3

Molecules

Data Set: Table with 1.057.341 records containing molecules (using ICCartridge structure index type)

query	hits	time [sec]
	211	0.23
	247	0.1
	4736	3.66
	22	0.1
	565	0.4
	12846	4.6

XML Command Examples for ICFSE

Create NEW Molecule Fast Search Index

```
<XML><UPDATE>
      <DB_NORM_STATUS>NORMALIZED</DB_NORM_STATUS>
      <CMD>
        <NAME>NEW</NAME>
        <NORMALIZE>YES</NORMALIZE>
        <INDEX_TYPE>MOL</INDEX_TYPE>
        <INDEX_DIRECTORY>
           i:\cartridge\idx_dir\
        </INDEX_DIRECTORY>
        <INPUT_FILE>
           i:\cartridge\idx_dir\index_input_file.ls
        </INPUT_FILE>
      </CMD>
</UPDATE></XML>
```

The specified INPUT_FILE can be provided in Rosdal or MDL-Molfile format. A **unique identifier** has to be delivered with every structure record. For more information please order the detailed documentation of ICFSE.

SEARCH Molecule Fast Search Index

```
<XML><MOL>
      <SD_FILE>E:\Spresi\Balsar\query.mol</SD_FILE>
      <CMD>
        <NAME>SSS</NAME>
        <ARG2>i:\cartridge\idx_dir\</ARG2>
      </CMD>
</MOL></XML>
```

The molecule query specified in the MDL-Molfile query.mol is launched to search in the fast search index created above. The results are sent in a specified format via TCP/IP containing the user defined structure identifiers. For more information please order the detailed documentation of ICFSE.

APPEND Molecule Fast Search Index

```
<XML><UPDATE>
      <CMD>
        <NAME>APPEND</NAME>
        <INDEX_DIRECTORY>
           i:\cartridge\idx_dir\
        </INDEX_DIRECTORY>
        <INPUT_FILE>
           i:\cartridge\idx_dir\index_app_file.ls
        </INPUT_FILE>
        <DELETE_RECORD_FILE>
           i:\cartridge\idx_dir\index_cleanid_file.ls
        </DELETE_RECORD_FILE>
      </CMD>
</UPDATE></XML>
```

The specified INPUT_FILE index_app_file.ls contains the structure information and the unique identifier for the appended records. The DELETE_RECORD_FILE contains the unique identifier for the deleted records. For more information please order the detailed documentation of ICFSE.

Create NEW Reaction Fast Search Index

```
<XML><UPDATE>
      <DB_NORM_STATUS>NORMALIZED</DB_NORM_STATUS>
      <CMD>
        <NAME>NEW</NAME>
        <NORMALIZE>YES</NORMALIZE>
        <INDEX_TYPE>RXN</INDEX_TYPE>
        <INDEX_DIRECTORY>
           i:\cartridge\idx_dir\
        </INDEX_DIRECTORY>
        <INPUT_FILE>
           i:\cartridge\idx_dir\index_input_file.ls
        </INPUT_FILE>
      </CMD>
</UPDATE></XML>
```

The specified INPUT_FILE can be provided in Rosdal or MDL-Rxnfile format. A unique identifier has to be delivered with every structure record. For more information please order the detailed documentation of ICFSE.

SEARCH Reaction Fast Search Index

```
<XML><MOL>
      <RXN_FILE>E:\Spresi\Balsar\query.rxn</RXN_FILE>
      <CMD>
        <NAME>RSS</NAME>
        <ARG2>i:\cartridge\idx_dir\</ARG2>
      </CMD>
</MOL></XML>
```

The reaction query specified in the MDL-Rxnfile query.rxn is launched to search in the fast search index created above. The results are sent in a specified format via TCP/IP containing the user defined structure identifiers. For more information please order the detailed documentation of ICFSE.

Examples for Java API

feed - Add new record to ICFSE chemistry index

Javadoc:

public java.lang.String **feed**(java.lang.String molBuffer)

throws ICStructureServiceException

Adds a new structure to the InfoChem FastSearch Engine (ICFSE). This method checks if the structure exists already in the index (registration) and generates a unique index (structure and sorted hashcodes).

Parameters:
molBuffer - normalized (if iccheck is used also unnormalized) Input Mol connection-table stored in a String buffer
Returns:
string containing the 18 digit unique hashcode
Throws:
ICStructureServiceException

Code Sample:

```
public void feed_test() {
try {
       structure = "\n  -ISIS-  04080511262D\n\n"
       + "  9  9  0  0  0  0  0  0  0  0999 V2000\n"
       + "    4.1402   -1.1500    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.1391   -1.9773    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.8539   -2.3902    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    5.5703   -1.9769    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    5.5675   -1.1463    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.8521   -0.7372    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.8496    0.0878    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.1339    0.4982    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    5.5629    0.5024    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "  4  5  1  0  0  0  0\n"
       + "  2  3  1  0  0  0  0\n"
       + "  5  6  2  0  0  0  0\n"
       + "  6  1  1  0  0  0  0\n"
       + "  1  2  2  0  0  0  0\n"
       + "  6  7  1  0  0  0  0\n"
       + "  3  4  2  0  0  0  0\n"
       + "  7  8  2  0  0  0  0\n"
       + "  7  9  2  0  0  0  0\n"
       + "M  END\n";

ICStructureService structureService = new ICStructureService("localhost", 10001);
String hashkey1 = structureService.feed(structure1);
System.out.println("Hashkey is: " + hashkey1);

String hashkey2 = structureService.getHashFromMol(structure1);
System.out.println("Hashkey is: " + hashkey2);
}
catch (Exception e) {
}
}
```

expandStructure – Search query in ICFSE chemistry index

Javadoc:

public java.util.Iterator **expandStructure**(java.lang.String molBuffer,
int searchOperator,
int similarityThreshhold,
int maxHitCount)
throws ICStructureServiceException

Forwards the structure query to ICFSE and returns a list of structure hits.

Parameters:
molBuffer - normalized Input Mol connection-table of the structure query stored in a String buffer
searchOperator - constant value for the ICFSE search operator (parameter must be defined): ICStructureService.SEARCH_OP_SSS, ...
similarityThreshhold - similarity threshold for similarity search in percent from 0 to 100
maxHitCount - maximum number of hits to be returned
Returns:
Iterator over the elements of a list of hits
Throws:
ICStructureServiceException

Code Sample:

```
public void expand_test() {
try {
       structure = "\n  -ISIS-  04080511262D\n\n"
       + "  9  9  0  0  0  0  0  0  0  0  0999 V2000\n"
       + "    4.1402   -1.1500    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.1391   -1.9773    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.8539   -2.3902    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    5.5703   -1.9769    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    5.5675   -1.1463    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.8521   -0.7372    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.8496    0.0878    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    4.1339    0.4982    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "    5.5629    0.5024    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0\n"
       + "  4  5  1  0  0  0  0\n"
       + "  2  3  1  0  0  0  0\n"
       + "  5  6  2  0  0  0  0\n"
       + "  6  1  1  0  0  0  0\n"
       + "  1  2  2  0  0  0  0\n"
       + "  6  7  1  0  0  0  0\n"
       + "  3  4  2  0  0  0  0\n"
       + "  7  8  2  0  0  0  0\n"
       + "  7  9  2  0  0  0  0\n"
       + "M  END\n";

ICStructureService structureService = new ICStructureService("localhost", 10001);
Iterator iter = structureService.expandStructure(structure,
ICStructureService.SEARCH_OP_SSS, 50, 5000);

while (iter.hasNext()) {
       FWStructureHit hit = (FWStructureHit)iter.next();
       System.out.println(hit.getKey());
       }
}
catch (Exception e) {
}
}
```