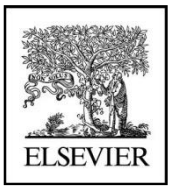

Reaxys API Manual

Reaxys Application Programming Interface

Introduction, Syntax & Examples

Version: 1.3

reaxys.com

CONTENTS

TABLES

1. INTRODUCTION

Reaxys supports search and retrieval of chemistry reaction and substance data with citations through a web-based application implemented for HTML browsers on several platforms. The Reaxys API provides a different way of accessing the Reaxys database and search engine, allowing programmers to write their own direct interface to the native Reaxys data in XML format.

This document describes the use of the Reaxys API (later called “API” in this document).

The Reaxys API functionality is available from the following products: Reaxys Expert API and Reaxys Medicinal Chemistry Expert API.

1.1 Glossary

The glossary defines the meaning of some expressions in the specific context of this document.

Table 1: Glossary

Term	Description
API	Application programmers interface; in the context of this document the API of Reaxys
Caller	Name of the client application, which uses the Reaxys API
JSESSIONID	Session identifier for the state-full API
Persistent cookie	Cookie, which ensures that an existing session is returned to the correct server in the pool, which holds the opened session (load balancer influence)
Reaxys	Elsevier’s reactions, substances and citations database including the web browser client
stationID	Identifier of a physical unit at client side

2. LOGICAL DESCRIPTION

2.1 Application Caller Name (“API Key”)

Each application, which is built on the Reaxys API, must identify itself with a unique caller name. Reaxys will be configured to accept applications with known caller names. An application caller name must not be reused for another application.

2.2 State-full Workflow

The use of the API is state-full. This means that any caller must:

- Open a session using his credentials
- Store the session parameters
- Reuse them for all operations on the API
- And close the session when all operations have been completed.

The API is a web-service with the usual restrictions:

- An idle anonymous session is closed by time-out after 30 minutes
- An idle named session is closed after 6 hours
- The time between sending a request and waiting for the response cannot exceed 4 minutes.

When a session is opened, the API returns 4 important session parameters, which must be kept and reused for each further request during the same session:

- JSESSIONID: unique key as part of the response header identifying the API session, e.g. like “JSESSIONID=CF334B142654E32594138F957DC5508E”
- sessionid: unique key as part of the response identifying the server session, e.g. like “7166096181282156194”
- stationid: unique key as part of the response header identifying the station, e.g. like “stationed= 418FD88183F92391F3BB692137F74E511293028790045_d956597522807e343 eeb4a061e8488”
- persistence “X-RE-Persist”: identifier routing further requests to the right server in the multi-server environment, e.g. “3hU4k4y3aSdPyb/SdShDtzq1ZMUF0yAg5ZR/PhLXIEylv0ilg4AJ1 zpRqqRaEQnjfZ+AAWrCbIpPcEOPrAYZ6HexvaJTWi+eFCoGzrMeEfg7jbRT5Rl4Ghy6nB8M3 t+flQ==”

The next chapters will provide examples how to use.

2.3 Searches and Data Retrieval

The initial task is the creation of a "hitset" by a search – this is a named list of objects, which match the search parameters. If the result of a search does not contain a hit, the hitset is not generated nor receives a name. The possible objects are:

1. Reactions (R)
2. Substances (S)
3. Citations (C)
4. Bioactivities (DPI)
5. Targets (TGI)

Reactions can only be search and retrieved with a Reaxys Expert API license. Bioactivities and targets can only be searched and retrieved with a Reaxys Medicinal Chemistry Expert API license. With a combined Reaxys and Reaxys Medicinal Chemistry Expert API license, all objects can be searched and retrieved.

It is important to understand that your searches must be precise as possible: A search for a melting point (e.g. "MP.MP = 20-21") will retrieve a list of all substances having at least one melting point fulfilling this condition. These substances may also have other melting points. The hit (the melting point, which caused the substance being member of the hitset) is highlighted and can be identified in the returning XML data.

With any search the caller must specify the object type he is searching for. A search for an author in reactions (R) provides a hitset of reactions having at least one reaction detail, which references at least one citation with the matching author name.

In a second step the data for the hits can be retrieved. The caller defines which parts of the object are needed. When sending a retrieval request, the caller specifies:

- Name of the hitset
- Items of this hitset
- Facts, which have to be retrieved
- Number of instances per fact

If necessary, a search and retrieval of data can be combined to one request.

2.4 Performance and Optimization

In order to prevent too much traffic and unnecessary searches on a server, the following conditions must be respected:

- WORKER: This option must always be used to reduce the load on Reaxys.
- NO_CORESULT: When a search is performed on substances or reactions, another search automatically provides a de-duplicated list of corresponding citations. This is typically not needed for API calls; so the use of this option will increase the performance dramatically (depending on the size of the suppressed co-hitset).
- Track existing hitsets and reuse them as often as possible instead of recreating them by repeating the same search.

3. TECHNICAL IMPLEMENTATION

All requests are sent with HTTPS POST requests to the server. The payload – request and response – are exchanged in XML format using the native Reaxys data structure which comprises Reaxys and Reaxys Medicinal Chemistry fields. The data structure is described in rx.xsd and/or rx.dtd. Fields are listed in Reaxys_database_fields.xlsx.

3.1 Caller Identification

The application, which sends requests to the Reaxys API, must include its name into each single request. It's an attribute to the request node:

```
<request caller="$CALLER">
```

3.2 Login Request Types and Responses

Named Login Request (contains all known parameters)

```
<?xml version="1.0"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request caller="$CALLER">
      <statement command="connect" stationid="$STID"
        ip address="$IP" username="$UNAME" password="$UPW"/>
    </request>
  </xf>
```

Anonymous Login Request

```
<?xml version="1.0"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request caller="$CALLER">
      <statement command="connect" stationid=""
        ip_address="$IP" username="" password=""/>
    </request>
  </xf>
```

Login Response (similar for both cases)

Header

```
...
Set-Cookie: JSESSIONID=$JSID; Path=/reaxys
Set-Cookie: stationid=$STID; Expires=Thu, 22-Dec-2011 15:40:52 GMT; Path=/
...
```

Payload

```
<?xml version="1.0"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
```

```
  <request sessionid="$SID">
    <statement command="login" stationid="" username="" password=""/>
  </request>
  <response version="1.0.158">
    <status>OK</status>
    <message>1.130 sec</message>
    <sessions>
      <session>
        <sessionid>$SID</sessionid>
        <username/>
        <licensegroup>$GNAME</licensegroup>
        <full_username/>
        <companyname>$CNAME</companyname>
        <email/>
        <ip_address>$IP</ip_address>
        <peer_address>$PIP</peer_address>
        <stationid>$STID</stationid>
        <starttime>2010-12-22:10:40:53.516</starttime>
        <expirationtime>2010-12-22:11:40:53.516</expirationtime>
      </session>
    </sessions>
  </response>
</xf>
```

Notes

- The `status` node must signal `OK`; if the login fails, the reason is tried to describe in the message node.

  Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request>
      <statement command="connect" licensegroup=""
        username="" password="" ip_address="$IP"/>
    </request>
    <response version="1.0.174">
      <status>ERROR</status>
      <errnum>50</errnum>
      <message>Your username or password are wrong,
        or access isn't allowed from your IP address.;2.426 sec</message>
      <messages>
        <entry code="12" component="BAS" level="ERROR">
          <timestamp>2011-04-28T14:42:11,695</timestamp>
          <longtext>
            You have entered an invalid username/password combination.
            Please re-enter your username and password and click the GO button.
          </longtext>
        </entry>
      </messages>
    </response>
  </xf>
```

- `Username` and `password` may be absent; then an anonymous session is created. The transmitted IP as part of the request must be identical with the physical IP of the communication; the latter is used for checking access credentials.

  Important The request returns two cookies – JSESSIONID and X-RE-Persistant. Both cookies must be stored during the whole lifetime of the session and added to each request which is sent to the server.

- The `stationID` is assigned by the API. Any client implementation must store the `stationID` as soon as it is assigned, and it is best to store the `stationID` if it is to be reused later for new sessions.

  Note Only API `stationIDs` will be accepted in the near future; it is not possible that the API-using application creates its own one. This means that the `stationIDs` returned from the server must be kept and treated carefully.

- A customer's license determines whether or not an anonymous request works or not. Most customers access Reaxys via a defined IP range. Nevertheless, a named session (with username and password) should be used whenever possible.

3.3 Logout Request

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request caller="$CALLER">
      <statement command="disconnect" sessionid="$SID"/>
    </request>
  </xf>
```

Response

```
<?xml version="1.0"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request sessionid="$SID">
      <statement command="disconnect"/>
    </request>
    <response version="1.0.158">
      <status>OK</status>
      <message>session ended;0.013 sec</message>
    </response>
  </xf>
```

Note Sessions must be closed, because the API does not allow numerous sessions per account. Any application must ensure that sessions are treated carefully, reused, and closed when no longer needed. Orphaned sessions are closed after an idle time-out.

3.4 Basic Search Requests

The following section provides basic search request and response examples for using the Reaxys API. Additional request and response examples can be found in section Request nodes.

3.4.1 Search for a Substance Request

```
<?xml version="1.0"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request caller="$CALLER" sessionid="$SID">
```

```
      <statement command="select"/>
      <select_list>
        <select_item/>
      </select_list>
      <from_clause dbname="RX" context="S"></from_clause>
      <where_clause>
        structure('
          Marvin  11031004132D
        HDR
          0  0  0     0  0            999 V3000
        M  V30 BEGIN CTAB
        M  V30 COUNTS 8 8 0 0 0
        M  V30 BEGIN ATOM
        M  V30 1 C 5.8641 -1.502 0 0
        M  V30 2 C 5.8641 -2.498 0 0
        M  V30 3 C 5.0058 -3.001 0 0
        M  V30 4 C 4.1359 -2.498 0 0
        M  V30 5 C 4.1359 -1.502 0 0
        M  V30 6 C 4.9941 -0.999 0 0
        M  V30 7 O 6.7333 -0.999 0 0
        M  V30 8 O 3.2667 -3.001 0 0
        M  V30 END ATOM
        M  V30 BEGIN BOND
        M  V30 1 2 1 2
        M  V30 2 1 1 6
        M  V30 3 1 1 7
        M  V30 4 1 2 3
        M  V30 5 2 3 4
        M  V30 6 1 4 5
        M  V30 7 1 4 8
        M  V30 8 2 5 6
        M  V30 END BOND
        M  V30 END CTAB
        M  END','compound,exact,isotopes,stereo_absolute,salts,mixtures,charges,radicals')
      </where_clause>
      <options>WORKER,NO_CORESULT</options>
    </request>
  </xf>
```

3.4.2 Search for a Substance Response

Note The response contains the name of the hitset (here H001_123), which is used later when retrieving data. The size (here 622) is the number of found substances (see context node).

```
<?xml version="1.0"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request caller="$CALLER" sessionid="$SID">
      <statement command="select"/>
      <select_list>
        <select_item/>
      </select_list>
      <from_clause context="S" dbname="RX">
        <where_clause>
          structure('Molfile repeated here',
          'compound,exact,isotopes,stereo_absolute,salts,mixtures,charges,radicals')
        </where_clause>
        <options>WORKER,NO_CORESULT</options>
      </from_clause>
    </request>
    <response version="1.0.158">
      <status>OK</status>
      <message>ignored: compound;;1.399 sec</message>
      <results>
        <result>
          <resultname>H001_123</resultname>
          <resultsize>622</resultsize>
          <dbname>RX101000RX</dbname>
          <context>substances</context>
          <created>2010-12-23:06:06:58.604</created>
          <where_clause>
```

```
            structure('Molfile repeated here',
            'compound,exact,isotopes,stereo_absolute,salts,mixtures,charges,radicals')
          </where_clause>
        </result>
      </results>
    </response>
  </xf>
```

3.4.3 Search for Specific Substance Request (FA)

```
<?xml version="1.0"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request caller="$CALLER" sessionid="$SID">
      <statement command="select"/>
      <select_list>
        <select item>IDE</select item>
        <select item>FA</select_item>
      </select list>
      <from_clause resultname="H001_123" first_item="20" last_item="25"/>
    </request>
  </xf>
```

3.4.4 Search for Specific Substance Response (FA)

Contains the detail data for item 20 to 25, e.g. for item no 20 (only partly displayed):

```
...
<items>
  <item index="20">
    <facts>
      <fact name="RX" type="fact" parent="CHE" display="Reaction">1</fact>
      <fact name="YY" type="fact" parent="ID" display="Structure">1</fact>
      <fact name="EXTID" type="fact" parent="ID" display="External Identifiers">2</fact>
      <fact name="MP" type="fact" parent="PHY" display="Melting Point">1</fact>
      <fact name="SLB" type="fact" parent="PHY" display="Solubility (MCS)">1</fact>
      <fact name="RCT" type="fact" parent="CHE" display="Presence as Reactant">1</fact>
      <fact name="BEH" type="fact" parent="CHE"
        display="Chemical Behavior Presence">1</fact>
      <fact name="CNR" type="fact" parent="BIB" display="Citation Number">1</fact>
      <fact name="ID" type="title" display="Identification">3</fact>
      <fact name="BIB" type="title" display="Bibliographic Data">1</fact>
      <fact name="CHE" type="title" display="Reaction Data">3</fact>
      <fact name="PHY" type="title" display="Physical Data">2</fact>
    </facts>
  </item>
</items>
...
<substances>
  <substance index="1">
    <IDE>
      <IDE.XRN highlight="true">3722272</IDE.XRN>
      <IDE.CN>piperidine; salt of hydroquinone</IDE.CN>
      <IDE.CN>Piperidin; Salz des Hydrochinons</IDE.CN>
      <IDE.AUN>Benzene-1,4-diol; compound with piperidine</IDE.AUN>
      <IDE.LSF>
        C<sub>6</sub>
        H<sub>6</sub>
        O<sub>2</sub>
        *C<sub>5</sub>
        H<sub>11</sub>
        N</IDE.LSF>
      <IDE01>
        <IDE.FMF>
          C<sub>5</sub>
          H<sub>11</sub>
          N</IDE.FMF>
        <IDE.FXRN>102438</IDE.FXRN>
```

```
      </IDE01>
      <IDE01>
        <IDE.FMF>
          C<sub>6</sub>
          H<sub>6</sub>
          O<sub>2</sub>
        </IDE.FMF>
        <IDE.FXRN>605970</IDE.FXRN>
      </IDE01>
      <IDE.MF>
        C<sub>5</sub>
        H<sub>11</sub>
        N*C<sub>6</sub>
        H<sub>6</sub>
        O<sub>2</sub>
      </IDE.MF>
      <IDE.CHA>0</IDE.CHA>
      <IDE.ELC>
        C<sub>11</sub>
      </IDE.ELC>
      <IDE.ELC>
        H<sub>17</sub>
      </IDE.ELC>
      <IDE.ELC>N</IDE.ELC>
      <IDE.ELC>
        O<sub>2</sub>
      </IDE.ELC>
      <IDE.NA>31</IDE.NA>
      <IDE.NE>4</IDE.NE>
      <IDE.NF>2</IDE.NF>
      <IDE.MW>195.261</IDE.MW>
      <IDE.STYPE>heterocyclic</IDE.STYPE>
      <IDE.INCHI>VFFSQRZZXZBCQH-UHFFFAOYSA-N</IDE.INCHI>
      <IDE.INCHA>VFFSQRZZXZBCQH-UHFFFAOYAN</IDE.INCHA>
      <IDE.AVAIL>0</IDE.AVAIL>
      <IDE.MAXPUB>1898</IDE.MAXPUB>
      <IDE.NUMREF>1</IDE.NUMREF>
      <IDE.MARKREF>0</IDE.MARKREF>
      <IDE.MARKREF>0</IDE.MARKREF>
      <IDE.MARKREF>0</IDE.MARKREF>
      <IDE.MARKREF>0</IDE.MARKREF>
      <IDE.MARKREF>0</IDE.MARKREF>
      <IDE.MARKREF>0</IDE.MARKREF>
      <IDE.MARKREF>0</IDE.MARKREF>
      <IDE.ED>1991/02/26</IDE.ED>
      <IDE.UPD>2008/02/19</IDE.UPD>
    </IDE>
    <FA>
      <FA.FA>RX(1),YY(1),EXTID(2),MP(1),SLB(1),RCT(1),BEH(1),CNR(1)</FA.FA>
    </FA>
  </substance>
<substance index="2">
...
```

3.4.5 Retrieving Melting Points for Specific Items Request

Example for retrieving the melting points for item no 20 (note the syntax for repeated melting point facts):

```
<?xml version="1.0"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request caller="$CALLER" sessionid="$SID">
      <statement command="select"/>
      <select_list>
        <select item>IDE</select item>
        <select_item>MP(1,10)</select_item>
      </select_list>
      <from clause resultname="H001_123" first_item="20" last_item="20"/>
    </request>
  </xf>
```

3.4.6 Retrieving Melting Points for Specific Items Response

Extraction of melting points for item no 20:

```
...
</IDE>
  <MP>
    <MP.L>966300</MP.L>
    <MP.MP>102 - 104</MP.MP>
    <MP.ED>2007/11/05</MP.ED>
       <citations>
         <citation index="1">
           <CNR>
             <CNR.CNR>966300</CNR.CNR>
             <CNR.CED>2007/10/04</CNR.CED>
             <CNR.CUPD>2008/01/25</CNR.CUPD>
           </CNR>
           <CIT>
             <CIT.DT>Journal</CIT.DT>
             <CIT.AU>Rosenheim; Schidrowitz</CIT.AU>
             <CIT01>
               <CIT.CO>JCSOA9</CIT.CO>
               <CIT.JT>Journal of the Chemical Society</CIT.JT>
               <CIT.JTS>J. Chem. Soc.</CIT.JTS>
               <CIT.VL>73</CIT.VL>
               <CIT.PY>1898</CIT.PY>
               <CIT.PAG>141</CIT.PAG>
               <CIT.ISSN>0368-1769</CIT.ISSN>
             </CIT01>
           </CIT>
       </citation>
       ...
```

Note The de-duplicated citations are repeated at the end of the XML response.

3.5 Restrictions

The API implements restrictions of the user access in order protecting the stability of the product. At the point of writing this document, it is:

- The number of retrieved objects per request (number of items) is restricted (typically configured to 100). If more data are needed, several requests must be sent step by step.
- The number of retrieved instances of one fact is restricted (50). If more data are needed, several requests must be sent step by step.
- Not all commands in “Table 2: Command table” (page 15) are allowed. Please consider the return message, when the value is different from “OK”.

Further restrictions will be applied and published soon. Please regard that any implementation of the API must carefully use the API and protect Reaxys according to best programming practice.

4. REQUEST NODES

The XML will contain a <response> node for the status of the request plus 0 to 1 data nodes <substances>, <reactions>, and <citations>, in the order shown. In addition, the request XML is included.

4.1 Request XML Nodes

Direct subnodes of <request> are:

```
statement
select_list
cluster_list
into_clause
from_clause
where_clause
group_by_clause
order_by_clause
```

The request node has the attribute sessionid specifying the ID (a long random number) of an existing session. It is required except when a new session is being established or when a new password is requested outside a session.

A further attribute to <request> is commandid: It is interpreted for select requests representing searches and for cancel requests.

The statement node is required. A command and possibly its parameters are given as node attributes:

Table 2: Command table

Field	Definition
command	One of connect, disconnect, login, getpassword, cancel, save, delete, sessions, expand, select.
licensegroup	For command = connect: Name representing an organization holding a license. Can be absent or empty if the ip_address parameter is enough for identification.
username	For commands = connect, login, getpassword: The user name. Can be absent or empty if the ip_address parameter is enough for identification (connect only).
password	For commands = connect, login: The user's password. Can be absent or empty if the ip_address parameter is enough for identification. (connect only).
ip_address	For command = connect: The organization's IP address. Optional.

Field	Definition
stationid	For command = connect: Identifier for the user's workstation. It should consist of ASCII capital or small letters, digits, and underscores. The maximum length is 125, reasonably e.g. 32.
shibboleth_cookie	For command = connect: An identifier required if Shibboleth authentication is desired, otherwise absent.
firstname	For command = login, getpassword: Given name of the user, used in registration.
lastname	For command = login, getpassword: Surname of the user, used in registration.
Title	For command = login, getpassword: Title of the user, e.g. "Mr."
Email	For command = login, getpassword: Email address of the user, needs to be present and unique.
Jobtitle	For command = login: Job title of a user being registered, optional.
institution	For command = login: Institution of a user being registered, optional.
Location	For command = login: Location of a user being registered, optional.
want_info	For command = login: Indicator for a user being registered, optional, user wants to receive customer care information.
resultname	For command = save or delete: Name of an existing result set.
New_resultname	For command = save: Name of a result set to be created.
comment	For command = save: User comment to be attached to the saved result. Free format.
Query	For command = save: User query to be attached to the saved result. Free format.

The sessions command has no parameters and is requesting a list of all current session on the XML server.

The expand command has its parameters in the from_clause and where_clause subnodes. The index on a database field is listed.

select statements are requesting the following types of server interactions:

- search: Run a query creating a "result set".
- retrieveData: Return selected items out of a result set.
- retrieveClusters: Create statistics based on a result set, return selected items out of the statistics.

There is a way to merge an initial search with retrieving the first portions of data and clusters:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request sessionid="$SID">
      <statement command="select"/>
      <select_list>
        <select_item>IDE</select_item>
        <select item>MP(1,3)</select_item>
      </select_list>
      <from_clause dbname="rx" context="s" first_item="1"></from_clause>
      <where_clause>mp.mp=100-105</where_clause>
    </request>
    <response version="1.0.174">
      <status>OK</status>
      <message>2.394 sec</message>
      <results>
        <result>
          <resultname>H009_345</resultname>
          <resultsize>214908</resultsize>
          <citationset>H010 789</citationset>
          <citationcount>713002</citationcount>
          <dbname>RX110300RX</dbname>
          <context>substances</context>
          <created>2011-04-28:16:20:17.047</created>
          <comment>saved 2011-04-28:16:20:17.047</comment>
          <where clause>mp.mp=100-105</where_clause>
        </result>
      </results>
    </response>
    <substances>
      <substance index="1">
        <IDE>
          <IDE.XRN>1033</IDE.XRN>
          ...
          (more nodes)
          ...
        </IDE>
        <MP>
          <MP.L>583891</MP.L>
          <MP.MP highlight="true">105</MP.MP>
          <MP.SOL>H2O</MP.SOL>
          <MP.ED>2007/10/07</MP.ED>
          <citations>
            <citation index="1">
              <CNR>
                <CNR.CNR>583891</CNR.CNR>
                <CNR.CED>2007/10/07</CNR.CED>
                <CNR.CUPD>2008/01/25</CNR.CUPD>
              </CNR>
              <CIT>
                <CIT.DT>Journal</CIT.DT>
                <CIT.AU>Vogel; Debowska-Kurnicka</CIT.AU>
                <CIT.IMPACT>1.65</CIT.IMPACT>
                <CIT01>
                  <CIT.CO>HCACAV</CIT.CO>
                  <CIT.JT>Helvetica Chimica Acta</CIT.JT>
                  <CIT.JTS>Helv. Chim. Acta</CIT.JTS>
                  <CIT.VL>11</CIT.VL>
                  <CIT.PY>1928</CIT.PY>
                  <CIT.PAG>910,914</CIT.PAG>
                  <CIT.DOI>10.1002/hlca.192801101108</CIT.DOI>
                  <CIT.ISSN>0018-019X</CIT.ISSN>
                </CIT01>
              </CIT>
            </citation>
          </citations>
        </MP>
      </substance>
    </substances>
    <citations>
      <citation index="1">
        <CNR>
          <CNR.CNR>583891</CNR.CNR>
          <CNR.CED>2007/10/07</CNR.CED>
          <CNR.CUPD>2008/01/25</CNR.CUPD>
        </CNR>
        <CIT>
          <CIT.DT>Journal</CIT.DT>
```

```
            ...
            (more nodes)
            ...
          </CIT>
        </citation>
      </citations>
    </xf>
```

- `listDatabases`: All available
- `listResults`: All created in a session or saved permanently.

4.1.1 Connect Request

Note In all examples, the XML response is shown and contains the originating request within `<request>...</request>`.

Table 3: Connect Types

Example 1	Specify `licensegroup` (usually empty), `ip_address`, `user`, and `password` (both possibly empty).
Example 2	Specify `shibboleth_cookie` and `ip_address`. A positive response either says that a session was created (returning a `session_token`) or that the user should select among multiple "paths" (returning no `sessionid`, but a `<paths>` node plus a `session_token`).
Example 3	If multiple paths were returned, a second connect request based on the user's selection has to go out with these attributes **Attribute** / **From node within <session>:** `session_token` / `session_token` `path` / `Number`- The one out of several that the user has selected.

Example 1 – [licensegroup] + ip_address + user + password:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "http://host:port/xfserv/bx.dtd">
  <xf>
    <request>
      <statement command="connect" licensegroup="$GNAME"
        username="$UNAME" password="$UPW"/>
    </request>
    <response>
      <status>OK</status>
      <sessions>
        <session>
          <sessionid>$SID</sessionid>
          <username>$UNAME</username>
          <licensegroup>$GNAME</licensegroup>
          <starttime>2008-02-01:10:03:37.565</starttime>
          <expirationtime>2008-02-01:11:03:37.565</expirationtime>
        </session>
      </sessions>
    </response>
  </xf>
```

Example 2 – shibboleth_cookie, single choice:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "http://host:port/xfserv/bx.dtd">
  <xf>
    <request>
      <statement command="connect" shibboleth_cookie="123"/>
    </request>
    <response>
      <status>OK</status>
      <sessions>
        <session>
          <sessionid>$SID</sessionid>
          <username>$UNAME</username>
          <licensegroup>$GNAME</licensegroup>
          <starttime>2008-02-01:10:03:37.565</starttime>
          <expirationtime>2008-02-01:11:03:37.565</expirationtime>
          <session_token>234</session_token>
        </session>
      </sessions>
    </response>
  </xf>
```

Example 3 – shibboleth_cookie, multiple choices:

Initial Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
   <xf>
     <request>
       <statement command="connect" licensegroup="" username=""
         password="" ip_address="$IP" stationid="" shibboleth_cookie="123"/>
     </request>
     <response version="1.0.119">
       <status>OK</status>
       <message>2.258 sec</message>
       <sessions>
         <session_token>234</session_token>
         <paths>
           <path>
             <number>624410</number>
             <description>Reaxys Test Acct 1, Reaxys Test Dept A</description>
           </path>
           <path>
             <number>624608</number>
             <description>Reaxys Test Acct 3, Reaxys Test Dept 3A</description>
           </path>
         </paths>
       </sessions>
     </response>
   </xf>
```

Subsequent Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
   <xf>
     <request>
       <statement command="connect" licensegroup="" username="" password=""
         ip_address="$IP" stationid="" shibboleth_cookie=""
         session_token="234" path="624608"/>
```

```
  </request>
  <response version="1.0.119">
    <status>OK</status>
    <message>1.523 sec</message>
    <sessions>
      <session>
        <sessionid>$SID</sessionid>
        <username>$UNAME</username>
        <licensegroup>$GNAME</licensegroup>
        <full_username>N/A AnonShibboleth AnonShibboleth</full_username>
        <companyname>$CNAME</companyname>
        <email>null</email>
        <ip_address>$IP</ip_address>
        <peer_address>$PIP</peer_address>
        <starttime>2010-02-10:17:11:20.393</starttime>
        <expirationtime>2010-02-10:18:11:20.393</expirationtime>
        <session_token>234</session_token>
      </session>
    </sessions>
  </response>
</xf>
```

Table 4: Connect parameters

Field	Definition
licensegroup	Abbreviated name of the company or organization holding a license on the Xfire server addressed. Allowed characters are ASCII letters, digits, underscores. Maximum length is 32. If IP licensing is in effect and if an ip_address is given, this parameter should be absent or empty.
Username	Same maximum length and allowed characters. Can be absent or empty for a non-empty ip_address: anonymous login.
Password	Allowed characters are ASCII 32 to 126. Maximum length is 32 (to be checked). May be anything for username = 'anonymous'. Enclose in single quotes if blanks are present. Can be absent or empty for a non-empty ip_address: anonymous login.
ip_address	IP address of the peer connected to the application server, dot-connected.
shibboleth_cookie	An identifier creating a valid session, no further parameters required.
session_token	An optional identifier of the newly created session, provided by Authentication, see example 2. To be specified in subsequent requests, see example 3.
Path	Also required in a subsequent request (example 3). The value is one of the "number" nodes in the initial request's response.
Number	Identifier for one of the "departments" a user could work in.
description	Accompanying text – department name.

The session ID is contained in the response. It is a random Java Long, and is passed to all succeeding method calls.

Note Sessions are expiring at the time returned unless there is an intervening activity by the user. Requests to expired sessions give a “no such session” message. To continue, a new session has to be made.

Table 5: Error codes and descriptions

In case of an error response to a connect request, there are 2 possibilities to be distinguished by error or warning code:

Field	Definition
50	ERROR: The combination of username, password, and `ip_address` does not pass authentication, i.e. one or more of the 3 parameters do not have an allowed value. A session is not established.
76	WARNING: Authentication didn’t fail, but the session should not be actually used for the specific reason that an anonymous session (i.e. username == password == “”) is not allowed for the license group identified by `ip_address`, due to a specific setting for that license group. Based on code 76, the receiver of this response could display a specific message or page. A session is established.

4.1.2 Disconnect Request

To terminate a session

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "http://host:port/xfserv/bx.dtd">
  <xf>
    <request sessionid="$SID">
      <statement command="disconnect"/>
    </request>
    <response>
      <status>OK</status>
      <message>session ended</message>
    </response>
  </xf>
```

4.1.3 Validate Request

The validate request needs one or both pairs of non-empty parameters:

- `ip_address` plus `licensegroup`: it is checked if the IP address is known and if it belongs to the license group passed.
- `username` and `email`: the user needs to be known and have the e-mail address specified.

Example 1: IP check passed:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM “https://host:port/xfserv/rx.dtd”>
```

```
<xf>
  <request>
    <statement command="validate" ip_address="$IP" licensegroup="$GNAME"/>
  </request>
  <response version="1.0.157">
    <status>OK</status>
    <message>
      IP $IP not found in A&E. IP $IP found in BAS with correct group name.;2.001 sec
    </message>
  </response>
</xf>
```

Example 2: IP check failed:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request>
      <statement command="validate" ip_address="$IP" licensegroup="$GNAME"/>
    </request>
    <response version="1.0.157">
      <status>ERROR</status>
      <errnum>0</errnum>
      <message>
        IP $IP not found in A&E. IP $IP not found in BAS.;2.213 sec
      </message>
    </response>
  </xf>
```

4.1.4 Progress Request

Return information about the session status, esp. what is happening in a task currently running.

Note This command may (and should) be issued before the response to a previous command has come back.

Request plus Response example:

```
<xf>
  <request sessionid="$SID">
    <statement command="progress"/>
  </request>
  <response version="1.0.16">
    <status>OK</status>
    <message>searching for 5.815 sec
      Searching...
      Calling screening
      1 selected start-atoms:
         Atom:    1  Rank0: 0.290456    Rank: 0.233784
      /!S;NL=Screening;
      /!S;OL=Screening Set 0 Level 0...;
      /!S;OL=Screening Set 0 Level 1...;
      /!S;OL=Screening Set 0 Level 2...;
      /!S;OL=Screening Set 0 Level 3...;
    </message>
    <task>searching for 5.815 sec</task>
    <action>Screening Set 0 Level 3...</action>
    <percentage>12</percentage>
  </response>
</xf>
```

Table 6: Information sub-nodes and descriptions

Information is returned in these subnodes of `<response>`:

Field	Definition
`Status`	Should be always `OK`.
`Message`	Full text of server progress messages, probably not shown to the user.
`Task`	`none` or name of the current task plus time it took so far.
`Action`	Optional: current action within the task, possibly containing a completion percentage estimate.
`percentage`	Estimated percentage of structure search completion.
`Hits`	Estimated number of substances or reactions found (not in the above example).

4.1.5 Cancel Request

Example 1:

```
<xf>
  <request sessionid="$SID">
    <statement command="cancel"/>
  </request>
  <response version="1.0.16">
    <status>OK</status>
    <message>action cancelled;0.220 sec</message>
  </response>
</xf>
```

Example 2:

```
<xf>
  <request sessionid="$SID" commandid="$CID">
    <statement command="cancel"/>
  </request>
  <response version="1.0.16">
    <status>OK</status>
    <message>action cancelled;0.220 sec</message>
  </response>
</xf>
```

Cancel the command currently running; no action if there is no such command. The cancel command is returning immediately; the interrupted (other) command after some delay.

A cancelled search command could still return a result set most likely being incomplete.

Cancel can be directed against a "`commandid`" specified in a previous select request. If such a request is waiting for a resource, it is aborted. If it is busy, it is cancelled. If `commandid` is unknown, nothing happens.

4.1.6 Expand Request

There are 2 types:

- Return a portion of the index of a field in a database starting at a specific field value.
- Same content starting at a specific position.

Table 7: Expand Parameters (by value)

Field	Definition
dbname (from_clause)	Name of the database to address.
first_item (from_clause)	Should be 1.
last_item (from_clause)	Should be >= 1, determines the number of items returned.
where_clause	*fieldname = initial_value*. The value needs to be quoted unless it is numeric. Two single quotes stand for the index start.

```
<xf>
  <request sessionid="$SID">
    <statement command="expand"/>
    <from_clause dbname="BS085000AE" first_item="1" last_item="10">
    </from_clause>
    <where_clause>MP.MP=''</where_clause>
  </request>
  <response>
    <status>OK</status>
    <expands field="MP.MP" position="1" size="1122">
      <expand frequency="1">-35.16 - -35.16</expand>
      <expand frequency="1">17 - 17</expand>
      <expand frequency="1">24.84 - 24.84</expand>
      <expand frequency="1">28 - 29</expand>
      <expand frequency="1">28 - 30</expand>
      <expand frequency="1">29 - 29</expand>
      <expand frequency="2">29 - 30</expand>
      <expand frequency="1">31 - 33</expand>
      <expand frequency="3">32 - 33</expand>
      <expand frequency="1">32 - 34</expand>
    </expands>
  </response>
</xf>
```

Table 8: Expand parameters (by position)

Field	Definition
dbname (from_clause)	Name of the database to address.
first_item (from_clause)	Should be >= 1, starting position.

Field	Definition
last_item (from_clause)	Should be >= first_item, last position.
where_clause	*fieldname*. Identifying the database field.

```
<xf>
  <request sessionid="$SID">
    <statement command="expand"/>
    <from_clause dbname="BS085000AE" first_item="2" last_item="11">
    </from_clause>
    <where_clause>MP.MP</where_clause>
  </request>
  <response>
    <status>OK</status>
    <expands field="MP.MP" position="2" size="1122">
      <expand frequency="1">17 - 17</expand>
      <expand frequency="1">24.84 - 24.84</expand>
      <expand frequency="1">28 - 29</expand>
      <expand frequency="1">28 - 30</expand>
      <expand frequency="1">29 - 29</expand>
      <expand frequency="2">29 - 30</expand>
      <expand frequency="1">31 - 33</expand>
      <expand frequency="3">32 - 33</expand>
      <expand frequency="1">32 - 34</expand>
      <expand frequency="2">33 - 34</expand>
    </expands>
  </response>
</xf>
```

Table 9: Expand Output Sub-nodes

Field	Definition
expands.field	Field being expanded.
expands.position	Position of the first expand item in the index.
expands.size	Total index size.
expand.frequency	Number of items (substances, ...) having the value given.
expand(content)	Index value.

4.1.7 Search Request

A resultname, unless specified within <into_clause>, is generated automatically and needs to be specified in retrieve... calls.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "http://host:port/xfserv/bx.dtd">
  <xf>
    <request sessionid="$SID" commandid="$CID">
      <statement command="select"/>
      <select_list>
        <select_item>
        </select_item>
      </select_list>
      <from_clause dbname="BS085000AE" context="substances"></from_clause>
      <where_clause>MP.MP between 100 and 110</where_clause>
    </request>
    <response>
```

```
    <status>OK</status>
    <results>
      <result>
        <resultname>H001_123</resultname>
        <resultsize>209</resultsize>
        <citationcount>123</citationcount>
        <dbname>BS085000AE</dbname>
        <context>substances</context>
        <created>2008-02-01:11:15:34.081</created>
        <where_clause>MP.MP between 100 and 110</where_clause>
      </result>
    </results>
  </response>
</xf>
```

Table 10: Search parameters

Field	Definition
`commandid`	Optional ID for the request, possibly specified in a later cancel request.
`Dbname`	Raw name of the database to search.
`Context`	One of `Substances`, `Reactions`, `Citations`. `Items` retrieved will come from the section named.
`where_clause`	Non-empty boolean expression, more in the chapter about its syntax. A special case is contained('resultname') for reordering or regrouping the results.
`into_clause` (not in the example)	Name of the new result to be created by the search. The name must consist of letters, digits, and underscores, starting with a letter. Its length must be at most 28 characters. In addition: • The initial letter must not be a capital or small Q or H. • The name must end with an underscore plus a string derived from the session ID where an initial minus sign has been replaced by an underscore. Examples: X001_123 for a SID of 123 X002__123 for a SID of -123 Unlike resultsets created by "save" requests, the ones specified here are deleted on session termination.
`group_by_clause` (not in the example)	Comma-separated list of (see a later chpater for details), or empty: *`fieldname`* `[(asc|desc)] [(value|size)]`
`order_by_clause` (not in the example)	Comma-separated list of (see a later chpater for details), or empty: *`fieldname`* `[(asc|desc)]`
`options` (not in the example)	Comma separated list of options either like KEYWORD or KEYWORD=value Options for searching are: NO_CORESULT Do not create a (citation) co-

Field	Definition
	resultset. CHECKONLY — Do not run the search, only check its validity. WORKER — Do not create any extra results automatically. USE_PARTS=(true|false) — Create multiple intermediate results, one per boolean query component. The default is configured for the server. Search options applicable to a substance search giving related reactions. For each reaction in the final result, one or more of the substances that were found initially occur as: starting_material — Substances found appear as reactants. product — ... as products. reagent catalyst solvent reagent_or_catalyst — ... as either reagents or catalysts. **Note** Options applicable to retrieval may be present and are ignored.

4.1.8 Retrieve... Requests

There are two types of Retrieve Requests

- retrieveData
- retrieveCluster

4.1.8.1 retrieveData request

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request sessionid="$SID">
      <statement command="select"/>
      <select_list>
        <select_item>IDE</select_item>
        <select_item>FA</select_item>
        <select_item>YY</select_item>
      </select_list>
      <from_clause resultname="H001_123" first_item="1" last_item="3"></from_clause>
    </request>
    <response>
      <status>OK</status>
      <results>
        <result>
          <resultname>H001_123</resultname>
          <resultsize>209</resultsize>
          <citationcount>123</citationcount>
          <dbname>BS085000AE</dbname>
          <context>substances</context>
```

```
      <created>2008-02-01:11:27:54.891</created>
      <where_clause>MP.MP between 100 and 110</where_clause>
        <items>
          <item index="1">
            <facts>
              <fact name="YY">1</fact>
              <fact name="MP">1</fact>
              <fact name="NMR">2</fact>
              <fact name="IR">1</fact>
              <fact name="RX">2</fact>
            </facts>
          </item>
          <item index="2">
            <facts>
              <fact name="YY">1</fact>
              <fact name="INP">2</fact>
              <fact name="MP">2</fact>
              <fact name="NMR">2</fact>
              <fact name="MS">1</fact>
              <fact name="CNR">2</fact>
            </facts>
          </item>
          <item index="3">
            <facts>
              <fact name="YY">1</fact>
              <fact name="MP">1</fact>
              <fact name="RX">2</fact>
              <fact name="CNR">1</fact>
            </facts>
          </item>
        </items>
    </result>
  </results>
</response>
<substances>
  <substance index="1">
    <IDE>
      <IDE.BRN>48</IDE.BRN>
      <IDE.CN>1,2,3,4,5,6,7,8-octahydrophenazine</IDE.CN>
      <IDE.LSF>C<sub>12</sub>H<sub>16</sub>N<sub>2</sub>
      </IDE.LSF>
      <IDE01>
        <IDE.FMF>C....12H....16N.....2</IDE.FMF>
      </IDE01>
      <IDE.MF>C<sub>12</sub>H<sub>16</sub>N<sub>2</sub>
      </IDE.MF>
      <IDE.CHA>0</IDE.CHA>
      <IDE.ELC>C....12</IDE.ELC>
      <IDE.ELC>H....16</IDE.ELC>
      <IDE.ELC>N.....2</IDE.ELC>
      <IDE.NA>30</IDE.NA>
      <IDE.NE>3</IDE.NE>
      <IDE.NF>1</IDE.NF>
      <IDE.MW>188.272</IDE.MW>
      <IDE.ED>2007/10/25</IDE.ED>
      <IDE.UPD>2007/10/25</IDE.UPD>
    </IDE>
    <FA>
      <FA.FA>YY(1),MP(1),NMR(2),IR(1),RX(2)</FA.FA>
    </FA>
    <YY>
      <YY.STR>1641797258:eJyllE1qxDAMhfeB3EEnMJIl/2jdGdpNZzGL3v8olRPMBKwuqjEmiOfnjxfZeN/27e
      v23DcgAaoA6E5VhZ+MiGa0wUlqy6PCVJDrWdkywgf8xbjOF0YODBlGehCTU0WiuVnKBfP4B0YS25ibu4bTFK3
      trCwNxtPkxhNTSjANpcbzfApi9KRKIlWemI5vpDkzjAOnaItHGtGJadF7g6nXdm1sHMM0T4rimFcG0ZrfxpBh
      grfY/iYf30XmY2WRxXGbrzhuE6rjNnBzZAN3RzawOnKF0f5Fbod7SdJgvBqLu/sQBXJ6YgRiR87jPV1l9uRvg
      Of9E6Qf5f1x27dfYdvWig==</YY.STR>
    </YY>
  </substance>
  <substance index="2">
    <IDE>
      <IDE.BRN>249</IDE.BRN>
      <IDE.CN>isoimperatorin</IDE.CN>
      <IDE.LSF>C<sub>16</sub>H<sub>14</sub>O<sub>4</sub>
      </IDE.LSF>
      <IDE01>
        <IDE.FMF>C....16H....14O.....4</IDE.FMF>
      </IDE01>
```

```
          <IDE.MF>C<sub>16</sub>H<sub>14</sub>O<sub>4</sub>
          </IDE.MF>
          <IDE.CHA>0</IDE.CHA>
          <IDE.ELC>C....16</IDE.ELC>
          <IDE.ELC>H....14</IDE.ELC>
          <IDE.ELC>O.....4</IDE.ELC>
          <IDE.NA>34</IDE.NA>
          <IDE.NE>3</IDE.NE>
          <IDE.NF>1</IDE.NF>
          <IDE.MW>270.285</IDE.MW>
          <IDE.ED>2007/10/25</IDE.ED>
          <IDE.UPD>2007/10/25</IDE.UPD>
        </IDE>
        <FA>
          <FA.FA>YY(1),INP(2),MP(2),NMR(2),MS(1),CNR(2)</FA.FA>
        </FA>
        <YY>
          <YY.STR>367406120:eJydVUFqxDAMvAfyB7/ASLJlS+fu0l7awh76/6dUsb0QGvWwWsxiJtJkZuQk+7ZvH7f
          HviWCRJQSuEtV0w8BgBXar+RaEI4dZkYZO8h2GdJb+o/jvCYNZW7CcwfEFKQxNQR9NjN2DtLUTMplqYGuQRrM
          TRVXNqVE1Vg2UnHagwJyovl+jaZqW83UWzybritiQOagGs7I7XlubOLxiEVWIhZx9PhB7lX6JBSG+MA7ypoUF
          8QgjWVTdTUzcI2bKnVZoUJnU69MCmxS8+xihtqjpiybMpurZcPRSTVr0bOVqClpWmfYUOazHor4qcFMkUbPza
          FmPgxmrwRfW5YtjXWBy/i/wNWBrZ19uDmwEXfnlgaLU233U6ea0/H98GB0SFpCcuA+SC6wJPTMqwsbAzqZmAz
          kq0CTgU4mRoxOJkaMTrDGgOJUm251quX4NP+FP1N63N8TVR37+9dt334BFeYsKA==</YY.STR>
        </YY>
      </substance>
      <substance index="3">
        <IDE>
          <IDE.BRN>506</IDE.BRN>
          <IDE.CN>tert-butyl 2-(2-hydroxyethyl)-3-methoxyphenylcarbamate</IDE.CN>
          <IDE.LSF>C<sub>14</sub>H<sub>21</sub>NO<sub>4</sub>
          </IDE.LSF>
          <IDE01>
            <IDE.FMF>C....14H....21N.....1O.....4</IDE.FMF>
          </IDE01>
          <IDE.MF>C<sub>14</sub>H<sub>21</sub>NO<sub>4</sub>
          </IDE.MF>
          <IDE.CHA>0</IDE.CHA>
          <IDE.ELC>C....14</IDE.ELC>
          <IDE.ELC>H....21</IDE.ELC>
          <IDE.ELC>N.....1</IDE.ELC>
          <IDE.ELC>O.....4</IDE.ELC>
          <IDE.NA>40</IDE.NA>
          <IDE.NE>4</IDE.NE>
          <IDE.NF>1</IDE.NF>
          <IDE.MW>267.325</IDE.MW>
          <IDE.ED>2007/10/25</IDE.ED>
          <IDE.UPD>2007/10/25</IDE.UPD>
        </IDE>
        <FA>
          <FA.FA>YY(1),MP(1),RX(2),CNR(1)</FA.FA>
        </FA>
        <YY>
          <YY.STR>-
          1269950608:eJyllUtuwzAMRPcGfAedQOBH1GfdBO2mKZBF73+UUraCGBC7MG0IhjCmnjkjOVmXdfm6PdclYO
          sjgDlaa+GXAEAL9aKYges+q8jQZxD1MYSP8B/jOHYMx1T2xRQZkZ2Ydzes3VQvBmOhmkY3VOSCqYI7kKvQAfM
          4gUmRU+JXxHIlYoQBJPJiNBsmGRgGPGB+TmAg1txepgS9Efce0gi2cfZu+DtiPTcEXkw3Vejy8RtL9uMn9bhT
          ZyKWSCmP45cSeDGaDXIbizGTE5OjUupISbfKmU2JkEoepsD9aWo32I6Lvd0g1GGFpfm60S2i7T7JvD2Z5GRUa
          53YcjYgCi5GtcrVkPV9zZAl9J+TSc4B0ZDLJk+d1ICWef0H4lnW96Fhvssys7UUs1GtDRrmu2yY7/Js/juE5/
          0zCORtfn/c1uUPtE4XcA==</YY.STR>
        </YY>
      </substance>
    </substances>
</xf>
```

Table 11: Retrieve parameters

Field	Definition
`select_item`	A single select item is one of these strings: *viewname* *factname* or *factname*(*m*) or *factname*(*m*,*n*) Viewname is the name of a predefined view, factname is a direct child node to substance/reaction/citation, m and n define the repetition limits to return, default is (1,1).
`Resultname`	Name returned by a previous search.
`first_item, last_item`	Items (i.e. particular substances/reactions/citation) in the result set to retrieve data for.
`Options`	Comma-separated list of isolated keywords or key/value pairs. Keywords are in capital, (optional) values do not have enclosing quotes. Example: ISSUE_RXN=true HITONLY — This option is restricting the facts returned to such ones containing a highlight. ISSUE_RXN=(true|false) — For reaction structures (fact RY), issue a single V2000 or V3000 rxnfile in field RY.STR. Default underlined. ISSUE_RCT=(true|false) — For reaction structures (fact RY), issue multiple V2000 or V3000 molfiles in fields RY.RCT (for the reactants) and RY.PRO (for the products). Default underlined. COMPRESS=(true|false) — Compress all structures, see in the chapter about content below for more. EXPORT=(true|false) — Use a specific format for the printing service. ISSUE_ZCO=(true|false) — Omit Z coordinates from output structures. OMIT_MAPS=(true|false) — Omit mappings from output reactions.
View names defined: MARKUSH	Return the expanded Markush structure for Markush substances. In other cases, the return is identical to the normal structure without highlights or empty

4.1.8.2 retrieveClusters Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xf SYSTEM "https://host:port/xfserv/rx.dtd">
  <xf>
    <request sessionid="$SID">
      <statement command="select"/>
      <select_list>
        <select_item/>
      </select_list>
      <from_clause resultname="H001_123" grouplist="1,2" first_item="1" last_item="3">
      </from_clause>
      <where_clause/>
```

```
    <group_by_clause>IDE.MW desc size, IDE.MF asc value</group_by_clause>
  </request>
  <response version="1.0.174">
    <status>OK</status>
    <message>0.012 sec</message>
    <results>
      <result>
        <resultname>H001_123</resultname>
        <resultsize>214908</resultsize>
        <citationset>H002_234</citationset>
        <citationcount>713002</citationcount>
        <dbname>RX110300RX</dbname>
        <context>substances</context>
        <created>2011-04-28:16:19:34.498</created>
        <comment>saved 2011-04-28:16:19:34.498</comment>
        <where_clause>mp.mp=100-105</where_clause>
        <groups field="IDE.MW desc size">
          <groupcount>284</groupcount>
          <group index="1">
            <groupkey>&gt;276 - 288</groupkey>
            <groupsize>10189</groupsize>
          </group>
          <group index="2">
            <groupkey>&gt;264 - 276</groupkey>
            <groupsize>9997</groupsize>
          </group>
          <group index="3">
            <groupkey>&gt;288 - 300</groupkey>
            <groupsize>9928</groupsize>
          </group>
        </groups>
        <groups field="IDE.MF asc value">
          <groupcount>91715</groupcount>
          <group index="1">
            <groupkey>Ag*AsF<sub>6</sub>*2CF<sub>2</sub>N<sub>2</sub>S</groupkey>
            <groupsize>1</groupsize>
          </group>
          <group index="2">
            <groupkey>Ag*AsF<sub>6</sub>*2F<sub>2</sub>Xe</groupkey>
            <groupsize>1</groupsize>
          </group>
          <group index="3">
            <groupkey>Ag*BF<sub>4</sub>*2C<sub>7</sub>H<sub>6</sub>O*2C
              <sub>18</sub>H<sub>15</sub>P</groupkey>
            <groupsize>1</groupsize>
          </group>
        </groups>
      </result>
    </results>
  </response>
</xf>
```

Table 12: Group parameters

Field	Definition
`group_by_clause`	Each specifier looks like *`fieldname`* `[(asc|desc)] [(value|size)]` i.e. like in the groupByClause above. Alternately, use `<cluster_list>` and `<cluster_item>` nodes.
`Grouplist`	Each int must be 1 to the number of cluster items, requesting data for some of the clusters only.
`first_item, last_item`	Request the same range of items for all clusters specified in '`grouplist`' (int values).

5. RESPONSE NODE

An example of the request and response nodes is given in req.xml. The node has an optional attribute "version" indicating the XML server version producing it.

5.1 <response> subnodes

The `<response>` subnodes are described below. They represent the part of a response that is not database content and is status information.

5.1.1 <status> subnode

The content is `OK` or `ERROR`, no subnodes.

5.1.2 <errnum> subnode

For a `<status>` content other than `OK`, the number of the error or warning message following.

5.1.3 <message> subnode

A possible error or warning message plus, at the end, the total turnaround time for the request.

5.1.4 <messages> subnode

This node provides information about events occurring for a query or another kind of request that is to be presented to the end user:

```
<messages>
  <entry component="XML" level="ERROR" code="19">
    <timestamp>2009-06-05T17:19:26,811</timestamp>
    <longtext>The query ends with a field name or another unexpected word, in query: ide.xrn
    Please modify your query and try again. If the problem persists, then please contact our
    Customer Care team.</longtext>
  </entry>
</messages>
```

Table 13: <messages> subnode

Subnodes and attributes:

Field	Definition
`<entry>`	Subnode containing all parts of a single message. Possibly repeating.

Field	Definition
component=	Name of the message originator, “XML” for the XML server.
level=	FATAL Fatal error, a new session is needed. ERROR Error, the last action must be repeated in a different way. WARNING Warning (the last action’s outcome is possibly unexpected), possibly change and repeat. INFO Information, the outcome is OK.
code=	Error code, unique number per component, 0 == OK.
<timestamp>	Time of the event, in ISO format: *YYYY-MM-DDThh:mm:ss,sss*
<shorttext>	Short version of the message text.
<longtext>	Long version.

5.1.5 sessions> subnode

In responses to the “connect” and “sessions” commands, session information in these are the subnodes of <session>.

Table 14: <sessions> subnode

Field	Definition
Sessionid	The ID of the session, a long random number assigned on connect.
Username	Name (login ID) of the user.
licensegroup	Shorthand name representing the user's organization.
full_username	Full name of the user, like "Mr. A. Jones"
companyname	Actual name of the organization.
ip_address	IP address of the customer’s workstation, proxy, or firewall, as visible to the application server and specified in the connect request.
peer_address	IP address of peer having sent the request to the XML server.
Starttime	Session creation time.
expiriationtime	Time the session will expire in case of no intervening search or retrieval command.

5.1.6 <expands> subnode

Response to the expand command. <expands> is enclosing one or more <expand> nodes containing an index value and its frequency in data given as a like-named attribute.

Table 15: <expands> subnode

Attributes of <expands> are:

Field	Name of the field being expanded.
Position	Position of the first <expand> item following, with respect to the start of the field's index.
Size	Total number of index entries, i.e. not the number of <expand> items succeeding.

5.1.7 <results> subnode

Global data about the results of a search given as subnodes of <result> for one or more result sets:

Table 16: <results> sub-node

Field	Definition
Resultname	Name of the result set.
Resultsize	Size of the result set.
citationcount	Number of all citations referenced by all items in the result set. Absent if the items already are citations.
Dbname	Name of the database searched.
Context	Context, i.e. type of the items found: substances, reactions, or citations.
Sortmode	Currently not used.
Created	Creation time stamp of the result set.
Cancelled	Present with a value of “true” if the result came from a cancelled search.
Comment	User comment. Free format.
Query	User query. Free format.
into_clause	Currently not used.
from_clause	Currently no meaningful content used.
where_clause	Query leading to the results.
group_by_clause	Conditions controlling how the items are divided into groups.
order_by_clause	Conditions controlling item order based on specific field values.
query_parts	Returns partial results if the query was split into components, enclosing 2 or more <query_part> subnodes.

5.1.8 <query_part> subnodes

For certain queries

```
field1=value1 and field2=value2 ...
```

or

```
structure(...) and field=value ...
```

Query components separated by `and` are run individually first giving partial results later combined to a final result. Partial results are reported in a single subnode to node `<result>`, e.g.:

```
<query_parts>
  <query_part>
    <resultname>...</resultname>
    <resultsize>...</resultsize>
    <where_clause>...</where_clause>
  </query_part>
  <query_part>
    <resultname>...</resultname>
    <resultsize>...</resultsize>
    <where_clause>...</where_clause>
  </query_part>
</query_parts>
```

The 3 nodes within each `<query_part>` have the same meaning as above. A user could view a table or partial results and e.g. view a particular `resultset`.

The server can be configured to deliver the nodes by default, or not to do so. The USE_PARTS=(true|false) search option explicitly controls the behavior.

5.1.9 <groups> subnode

If cluster information was requested, `<result>` contains multiple `<groups field="...">` subnodes naming the group-by field and enclosing the data requested.

Table 17: <groups> subnode

Field	Definition
`<groupcount>`	Total number of groups for the field.
`<group index="...">`	Group at the position "index". Subnodes are: `<groupsize>` Number of items in the group. `<groupkey [type="type" ] [name="name" ] [parent="parent"]>` Characteristic value for the group.

The optional attributes type, name, and parent on the `groupkey` node are only present for the Property Hierarchy field.

Table 18: Attribute Types

Field	Definition
`Type`	"`fact`" or "`title`". A title is a common name for a specific set of facts, e.g. "`Melting Point`" and "`Boiling Point`" both belong to "`Physical Properties`". Titles themselves can appear under superordinate titles.
`Name`	Short name of the current fact or title.
`Parent`	Short name of the superordinate title to a fact or title. Empty or missing if the fact or title is on top.

<groups> Example:

```
<groups field="PH.PH desc size">
  <groupcount>135</groupcount>
  <group index="1">
    <groupkey type="title" name="ID" parent="">Substance Data</groupkey>
    <groupsize>127722</groupsize>
  </group>
  <group index="2">
    <groupkey type="fact" name="YY" parent="ID">Structure</groupkey>
    <groupsize>122100</groupsize>
  </group>
  <group index="3">
    <groupkey type="fact" name="RX" parent="CHE">Reaction</groupkey>
    <groupsize>89477</groupsize>
  </group>
  <group index="4">
    <groupkey type="fact" name="PRE" parent="CHE">Preparation Presence</groupkey>
    <groupsize>88145</groupsize>
  </group>
  <group index="5">
    <groupkey type="fact" name="PRO" parent="CHE">Presence as Product</groupkey>
    <groupsize>76817</groupsize>
  </group>
  <group index="6">
    <groupkey type="fact" name="PSD" parent="ID">Patent-Specific Data</groupkey>
    <groupsize>63036</groupsize>
  </group>
  <group index="7">
    <groupkey type="fact" name="DET" parent="CHE">Detailed Reaction Presence</groupkey>
    <groupsize>50046</groupsize>
  </group>
  <group index="8">
    <groupkey type="title" name="SPE" parent="">Spectroscopic Information</groupkey>
    <groupsize>46123</groupsize>
  </group>
  <group index="9">
    <groupkey type="fact" name="LB" parent="ID">Substance Label</groupkey>
    <groupsize>39853</groupsize>
  </group>
</groups>
```

5.1.10 <items> subnode

If field availabilities were requested, they are given in multiple <item index="..."> subnodes, one for each item requested, at position "index" in the result set.

Table 19: <items> subnode

Field	Definition
<groupno>	For a grouped result: number of the group the item is contained in. "index" is the position within the group in this case.
<groupkey>	For a grouped result: characteristic value of the current group. Currently not provided.
<groupsize>	For a grouped result: size of the current group. Currently not provided.
<facts>	Encloses multiple <fact name="..." type="..." parent="..." display="..."> nodes: name — Name of the fact (or title). type — Type of the node: "fact" or "title". parent — Name of the superordinate title or empty. display — Long name of the fact. Content — Number of occurrences of the fact within the item. In a special HITONLY retrieval mode, two counts are given: *restricted_count(total_count)* Where *total_count* would appear when the mode is not set.

6. CONTENT NODES AND XSD/DTD FILES

Direct subnodes to <xf> apart from <request> and <response> are zero to one nodes <substances>, <reactions>, or <citations>.

A pair of XSD/DTD files are available for the Reaxys database: rx.dtd and rx.xsd.

6.1 DTD content

Each DTD starts with these sections common to all database types:

- Entity definitions for greek letter symbols, e.g.

```
<!ENTITY alpha   "&#x03B1;">
```

- A %-entity “text” controlling subtags allowed within data field nodes that represent non-numeric “textual” database content:

```
<!ENTITY % text  "(#PCDATA|sub|sup|i|hi)*">
<!ELEMENT sub %text;>
<!ELEMENT sup %text;>
<!ELEMENT i %text;>
<!ELEMENT hi %text;>
```

- A text node may contain subtags sub(script), sup(erscript), i(talic), and hi(ghlighted) for the specific text markups named.
- The subnodes of <MARKUSH>, an inner node for expanded Markush structures.
- The subnodes of <request> and <response>.
- Structurally identical descriptions for the subnodes to substances, reactions, and citations, where e.g.:
    - substances contains one or more nodes substance. Same for reactions and citations.
    - substance contains subnodes representing “facts”. Each “fact” node is carrying a name of capital letters. Same for reaction and citation.
    - <!ELEMENT substance (IDE?,FA*,PH*,FLST*,BI*,YY*,...
    - A “fact” node is enclosing “field” nodes in 3 ways:
    - directly
    - indirectly at the first level via intermediate “stage” or “group” nodes.
    - indirectly at the second level via “stage” nodes containing “group” nodes containing fields.
    - Stage and group node names are capital letters plus digits.

- Fields are named according to *factname.fieldname* again using capital letters plus a dot.

DTD Content Example:

```
<!ELEMENT                                                                                    RXD
(RXD.L*,RXD.DID?,RXD.CL*,RXD.STP?,RXD.MID?,RXD.MTEXT?,RXD01*,RXD.SNR?,RXDS01*,citations?)>
```

Fact `RXD`, apart from fields `RXD.fieldname`, contains stage `RXDS01` and group `RXD01`. `RXDS01` contains fields and group `RXD02`. `RXD01` and `RXD02` contain fields.

```
<!ELEMENT                                                                                    RXDS01
(RXD.STG?,RXD02*,RXD.RGT*,RXD.CAT*,RXD.SOL*,RXD.RCS?,RXD.TIM?,RXD.T?,RXD.P?,RXD.PH?,RXD.COND
*,RXD.TYP*,
RXD.SUB*,RXD.PRT*,RXD.NAME?,RXD.DED?,RXD.COM*)>
<!ELEMENT RXD01 (RXD.YBRN?,RXD.YPRO?,RXD.YD?,RXD.NYD?)>
<!ELEMENT RXD02 (RXD.SRBN?,RXD.SRCT?)>
```

Any fact may contain a `citations` node for the bibliography of the articles or patents it references.

6.2 XSD content

An XSD for a database is adding this information to what is available from a DTD.

Table 20: schema types in Reaxys

Field	Definition
`intType`	Integers with an optional attribute on the field tag indicating that the value has been a search hit and so is highlighted. **Example**: `<MP.MP highlight="true">100</MP.MP>`
`realType`	Floating point format including optional exponents and highlighting: 1.3E-4
`rangeType`	*`lower_limit`* `[ - ` *`upper_limit`* `]` with real limits.
`textType`	Text with optional markups (`sub`, `sup`, `i`, `hi`) as defined above and containing entities according to the DTD. Hightlighting indicated by `<hi>` tags as well as by a `hightlight="true"` attribute.
`markushType`	Inner XML under a root `<MARKUSH>` for expanded Markush structures.

Attributes for facts and fields: Their location is in a pipe-separated string to be found in nested nodes element - annotation - appinfo. The pipe-separated components optionally start with *keyword*= and have these meanings.

Table 21: fact and field attributes (schema)

Field	Definition
`no_keyword`	Long name of the fact or field. These names could appear on display pages.
`xf:code=`	Internal field code used by the XML server towards the Xfire server, no external usage.
`xf:display=`	Values are: true or false. It is a hint if the current fact or field should be displayed.
`xf:fedit=`	Internal formatting instructions for the XML server.
`xf:format=`	Same.
`xf:ranks=`	Same.
`xf:refer=`	Values: `nothing`, `primekey`, `substances`, `reactions`, `citations`. The current field is: • The primary key of the current item • It contains a primary key value of another item in the section named or It has no such role
`xf:search=`	Values: `none` The current field cannot be searched using a relational expression in the where clause. Substance and reaction structures, however, can be searched in a structure() function. `exists` The current fact can be searched in an exists() function. `number` The current field is searchable using *fieldname* = *numeric_value*, *fieldname relation numeric_value*, or *fieldname* between *lower_numeric* and *upper_numeric* `phrase` Same expressions possible as for "number". Field values should be enclosed in single quotes. They may contain blanks or other separators plus these special characters in any position: ? stands for any character * stands for any string `word` Same as for "phrase" except for the difference that values should not contain blanks or other separators.
`xf:shortname=`	Internal field name used by the XML server towards the Xfire server, no external usage.
`xf:sortcode=`	If specified for a field, its name is allowed to appear in group-by or order-by clauses.
`xf:unit=`	The common physical unit for all values of this field, to be used in displaying. The values may contain numeric XML entities and tags ^{...}

Field	Definition
	for superscripts. **Note:** characters “<>&” are XML encoded.
xf:link=	The value is the name of another field containing primary key values, or the keyword is absent. If a hyperlink based on the current field is clicked, a search on the linked field should be triggered, using its value under a parent node common to the current field.
xf:presented=	The value is true, or the keyword is absent. The current field should be presented to the use as a searchable field.
xf:layout=(list|table)	Specify list or table (which is the default) format for the layout of a fact. For absence or a value of “table”, the display should look as before version 36 of this spec.

Hierarchy information: the way facts appear under “titles” (superordinate terms). This information is located in these nested XSD nodes:

```
<xsd:element name="xf" type="xfType">
  <xsd:annotation>
    <xsd:appinfo>
      <hierarchy>
      ...
      </hierarchy>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
```

Nodes within <hierarchy> are <title> (may have <title> and <fact> child nodes) and <fact> (no child nodes). The common attributes of <title> and <fact> are.

Table 22: hierarchy expressed in the schema

All short names are the same as their counterparts in <group> nodes.

Field	Definition
name	Short name of the item. For facts, identical to a fact’s node name.
parent	Short name of the parent title, missing or empty if the item is on top.
display	Long name of the item.

6.3 Nodes containing structures

Reaxys supports structures, reactions and Markush structures.

6.3.1 Structures: Node YY.STR

Structures are returned in Molfile V3000 format if they contain highlights, otherwise in V2000 format.

Structures may have been compressed using the java.util.zip.Deflater class. The compressed byte stream gets base64-encoded, padded by '=' to a multiple of 4. The Adler-32 checksum from Deflater plus a colon is prefixed.

Compression is controlled by the COMPRESS retrieval option described above.

6.3.2 Structures: Node YY.MARKUSH

Requested by a MARKUSH select item, the content of this node represents an expanded Markush structure. If the substance in question does not have Markush type, "MARKUSH" returns the non-highlighted Molfile of the normal structure in node YY.STR.

The content of YY.MARKUSH is inner XML under a <MARKUSH> root node. The structure of the inner XML is described by type "markushType" in the XSD and also represented in the DTD.

Subnode <ROOT> contains a Molfile representing the "Markush scaffold" and is similar to the Molfile in <YY.STR> but having higher display quality, subnodes <STRUCTURE> represent structured residue groups directly or indirectly referenced from the scaffold. Residue groups are carrying an arbitrary symbol in place of an element symbol. They can be nodes in the scaffold or be referenced in other residue groups.

6.3.3 Reactions: Nodes RY.STR, RY.RCT, and RY.PRO

Reactions can be returned in 3 ways, controlled by the retrieval options ISSUE_RXN/ISSUE_RCT.

Table 23: reaction nodes in the schema

Field	Definition
`false/true (default)`	Issue Molfiles for each of the 0 to R reactants (field RY.RCT) followed by 0 to P products (field RY.PRO). V2000/V3000 usage is as described above.
`true/false`	Issue a single V2000 (no highlights) or V3000 (with highlights) Rnxfile representing the entire reaction (field RY.STR).
`true/true`	Issue both types of data.
`false/false`	Interpreted like true/false.

7. APPENDIX

7.1 Where clause syntax

A where clause consists of one or more:

- Relational expressions or functions
- Logical operators, joining relational expressions or functions.
- Parentheses, properly nested.

7.1.1 Relational expressions are built with these operators:

=, <, <=, >, >=, `between ... and`, `in` (restricted use).

For "`in`", the fieldname must be one of:

- A primary key field followed by a list of primary key values. The expression can appear standalone. Lists are formed like in SQL.
- The name `itemno` followed by a list of item numbers in an ungrouped resultset. The expression must be preceded by "`contained(...) operator`" where `operator` is either `and` or `and not`.

  If the items are to come from a group or cluster, their specification must be like `groupno/itemno`. There has to be `contained(resultname,cluster_specifier)` in front.
- The name `groupno` followed by a list of specifiers '`groupno`' naming entire groups in a grouped resultset or cluster. Again, `contained(...) operator` must be in front, in the form `contained(resultname,cluster_specifier)`.

7.1.2 Unsupported operators:

<>, !=, `not in`

The field values must be enclosed into single quotes if they are non-numeric. Contained single quotes have to be doubled.

Within texts, '?' stands for any character and '*' for any string. The operator has to be '=' in this case.

Serveral alternate field values may be enumerated after the relational operator using unquoted semicolons as list separators, e.g.

```
field = value1 ; value2
```

Quoting (if any) has to be applied to each individual value. All relational operators except `between` and `in` are possible, but only = actually makes sense.

7.1.3 Logical operators are:

`and`, `or`, `and not`.

Note "`not`" may only come after "`and`".

Additional logical operators are proximity, near, next: the 2 or more field values requested must occur in the same fact (proximity), within a distance of 3 words (near), within a distance of 3 words and in the sequence given (next).

7.1.3.1 Functions are:

structure('molfile|rxnfile','keywords'). Returns true in case of (sub)structure match as determined by the keywords:

- `starting_material`: It must be a substance structure searched in reaction context. Hits are all reactions where an educt matches the structure searched for.
- `product`: Same restriction, same condition with product for educt.
- `all_reactions`: Same restriction, same condition, but searching both the educt and the product side, i.e. effectively merging the results of the 2 restrictions above.
- `reagent`: Same restriction, search for all reactions where one of the substances found originally occurs as a reagent.
- `catalyst`: Same restriction, search for all reactions where one of the substances found originally occurs as a catalyst.
- `solvent`: Same restriction, search for all reactions where one of the substances found originally occurs as a solvent.
- `reagent_or_catalyst`: Same restriction, search for all reactions where one of the substances found originally occurs as a reagent or a catalyst.
- `exact`: The hit structure should contain as many heavy atoms, bonds, fragments, rings, charges, and radicals as the query structure. For reactions, the restriction on the fragment count is lifted.
- `substructure`: The query structure can be embedded into the hit structure, none of the previous restrictions. Mutually exclusive to "exact", which is the default.
- `sub_hetereo`: exact search, but free substitution allowed on all non-C atoms.
- `isotopes`: If unset, the hit may contain isotopes only if the query does. Valid for both exact and substructure.
- `tautomers`: If set, tautomers of original hits are also found.
- `stereo_absolute`: All stereo centers in the query match the mapped centers in the hit.
- `similarity=... (value from 1 to 99)`: Request a similarity search rather than a (sub)structure search. The value controls the degree of similarity requested: low (more hits) or high (fewer hits).
- `stereo_relative`: All stereo centers in the query match the mapped centers in the hit or its mirror image (all centers synchronously inverted). Mutually exclusive to `stereo_absolute`, the default is a non-steric search.

- `separate_fragments`: Request that non-interconnected fragments of the query structure are mapped onto different fragments in the hit.
- `ignore_mappings`: Ignore requests of the query to specifically find reactant atoms mapped to product atoms.
- `salts`: If set, allow more fragments, charges, and radical dots to be present in the hit that in the query.
- `no_extra_rings`: If set, do not allow rings in the hit that are connecting two atoms in the query but are not yet present in the query.
- `charges`: Allow the hit to contain more charges than the query.
- `radicals`: Allow the hit to contain more radical dots than the query.
- `mixtures`: After a search for substances, add those substances to the result that reference a substance in the initial result as a mixture component.
- `markush`: After a search for substances, add those substances to the result that are referenced from an initial hit as a Markush structure scheme.
- `atoms=...`: Restrict the number of atoms in the hit to a (range of) positive integer(s). Ranges look like *lower hyphen upper*, e.g. 10-20.
- `fragments=...`: Restrict the number of fragments (interconnected atoms) in the hit to a (range of) positive integer(s).
- `rings=...`: Restrict the number of rings in the hit to a (range of) non-negative integer(s).
- `align`: On display, highlighted fragments found by the query will be rotated to a position where highlights are oriented similarly to the atoms in the query.

Structures may have been compressed using the java.util.zip.Deflater class. The compressed byte stream has to be base64-encoded, padded by '=' to a multiple of 4. The Adler-32 checksum from Deflater plus a colon may be prefixed.

Note Structures returned can be compressed in the same way, depending on a server config setting and the COMPRESS retrieval option.

contained('resultname') or **contained('resultname','cluster_specifier')**. Intersect or merge with all items in the result set. The form with `cluster_specifier` is used when '`itemno in ...`' or '`groupno in ...`' are following.

exists('factname'): Search for the existence of a fact.

7.2 Group-by clause syntax

```
fieldname [(asc|desc)] [(value|size)]
```

The resulting groups can be ordered by group key value or group size. Only a single or multiple specifications are allowed, controlled by the from clause containing either `group` (request a grouped result set) or `groups` (request clusters).

7.3 Order-by clause syntax

List of: *fieldname* [(asc|desc)]