EBI-MSD's API Project

Siamak Sobhany
Email: sobhany@ebi.ac.uk
URLs: http://www.ebi.ac.uk/msd
http://www.ebi.ac.uk/~sobhany

Database application program interface development to the EBI-MSD is a part of:

EU-TEMBLOR (New-generation bioinformatics) Project.

TEMBLOR is a the European Community Funded Research Programme under the "Quality of Life and Management of Living Resources" (1998-2002).

Prof Geoffrey J Barton

**School of Life Sciences,
WTB/MSI Complex, University of Dundee
Dow St
Dundee DD1 5EH
Scotland UK
Email: g.j.barton@dundee.ac.uk**

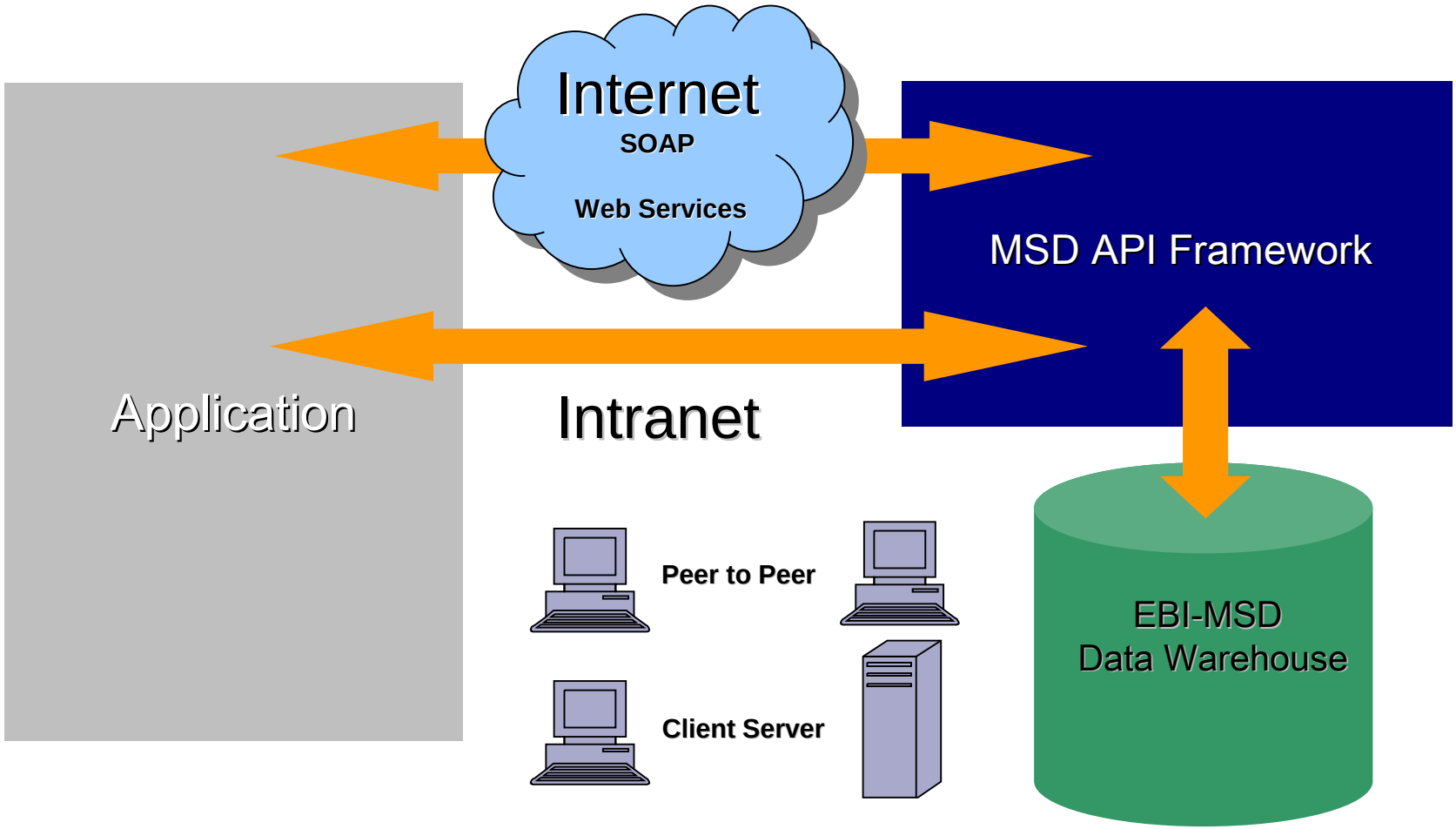
Dr Thomas W Hamelryck

**Dept. Ultrastructuur, Vrije Universiteit Brussel
Paardenstraat 65
1640 Sint-Genesius-Rode
Brussels
Belgium
E-mail:
thamelry@vub.ac.be**

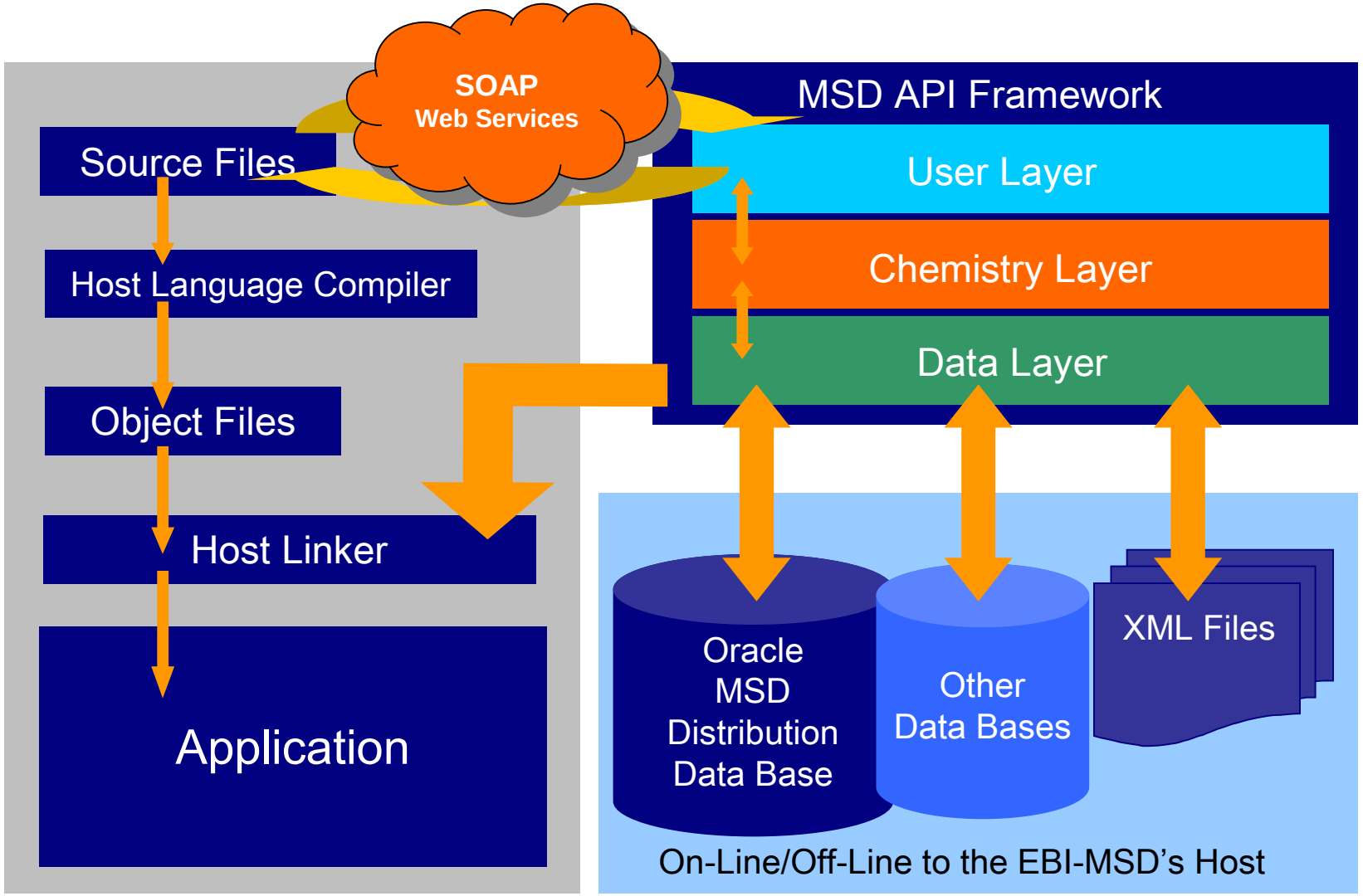
Project description:

This project will develop an Application Programming Interface (API) to the EBI-MSD database consist of a series of functions that external 3rd party software can use to allow their systems to access the EBI-MSD database independently.

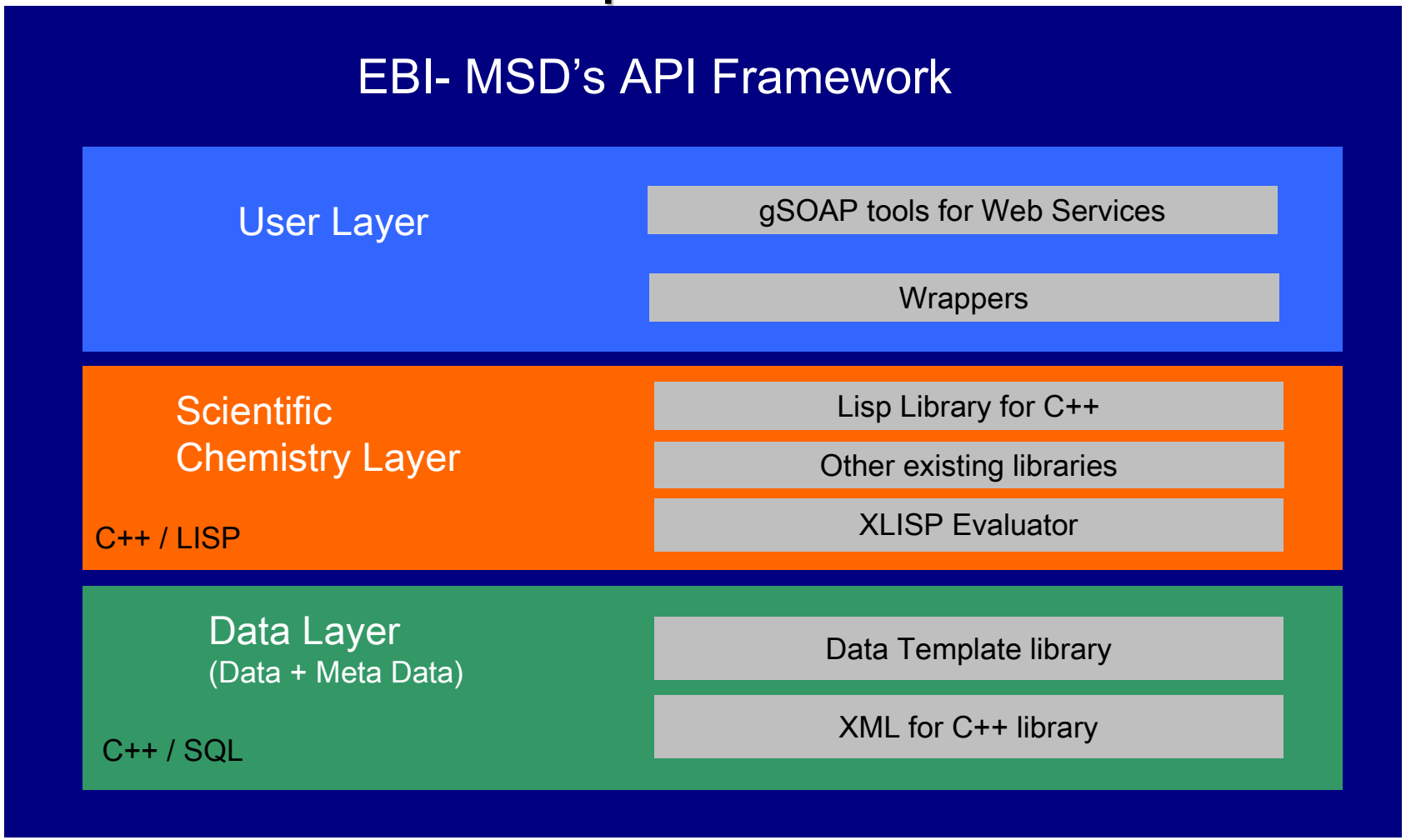
MSD API Infrastructure:

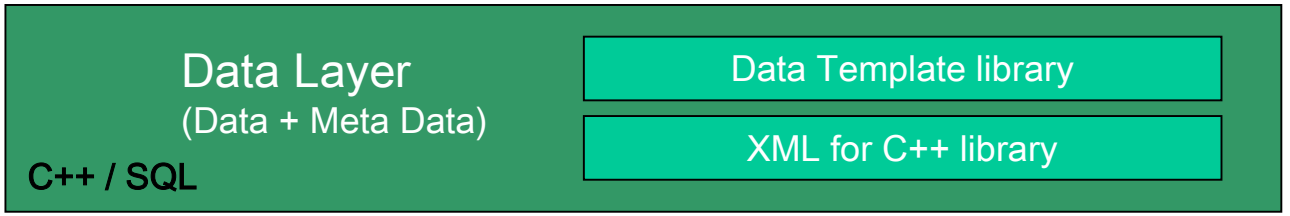


MSD API Framework Design:



MSD API Framework Implementation:





Data Layer: Data + Meta Data

Flexible API vs. rigid API

Data Warehouse is a **Live DatBase** and is changing not only from the point of view of Data but also from the structural aspects (because of producing and generating value added Data).
So we have to make API as dynamic and flexible to changes of data model as possible.
API includes Meta Data and Data model reading tools.

Data Layer technical:

Oracle, ODBC C++ Template Library (OTL):

To manipulate Oracle data or any other Data Bases such as MySQL, PostgreSQL, DB2,… through Oracle native call interfaces or ODBC bridges by SQL queries.

This library provides the following functionality:

Classes to design a scalable, shared server application that can support large numbers of users securely SQL access functions, for managing database access, processing SQL statements, calling stored procedures on an Oracle database server. Datatype mapping and manipulation functions, for manipulating data attributes of Oracle types with the best performance and efficiency.

With just a handful of concrete classes:

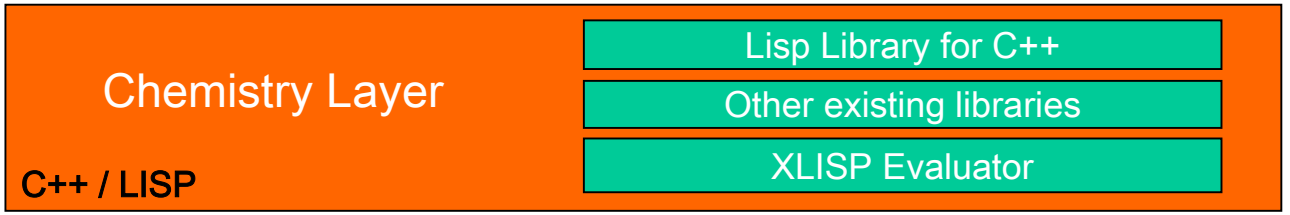
otl_stream, otl_connect, otl_exception, otl_long_string,This library gets expanded into direct database function calls, so it provides ultimate performance, reliability and thread safety in multi-processor environments as well as traditional batch programs. Its highly portable because is a single header file.

OTL stream concept:

Any SQL statement, PL/SQL block or a stored procedure call is characterized by its input / output [variables].

Xerces C++ Parser

Xerces-C++ is a validating XML parser written in a portable subset of C++. Xerces-C++ makes it easy to give Data Layer the ability to read and write XML data. A shared library is provided for parsing, generating, manipulating, and validating XML documents.
This part of Data Layer has not been implemented yet an is subject of nex stage in developing API.



Symbol based dynamic object and Data container.

Making on-the-fly combined result sets.

LISP Library in C++:

Implementation of Embedded AI sub-systems in C++

This will adds an Intelligent Agent part to MSD API Framework.

Supporting any application where dynamically typed objects are needed in C++. Gives us more of a dynamic Lisp prototyping environment than ordinary static C++.

XLISP Evaluator:

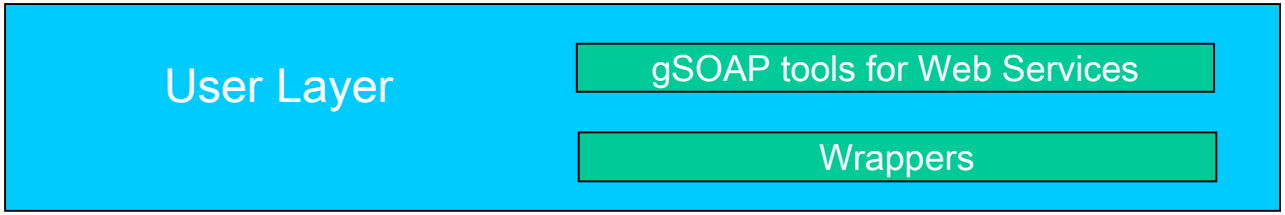
As built in LISP evaluator to load additional existing or new developed LISP programs and Scripts. This feature would be optional and for the extended phase of development:

Putting Semantic and Ontologies to Work

The scripting language comparison chart below gives you an overview of the features available in each of the most popular scripting languages today.

Features		Tcl	Perl	Python	Java Script	Visual Basic	LISP
Speed of use	Rapid development	✓	✓	✓	✓	✓	✓
	Flexible, rapid evolution	✓	✓	✓	✓	✓	✓
	Great regular expressions	✓	✓	✓			✓
Breadth of functionality	Easily extensible	✓		✓		✓	✓
	Embeddable	✓		✓			✓
	Easy GUIs	✓		✓*		✓	
	Internet and Web-enabled	✓	✓	✓	✓	✓	✓
Enterprise usage	Cross platform	✓	✓	✓	✓		✓
	Internationalization support	✓		✓	✓	✓	✓
	Thread safe	✓		✓*		✓	✓
	Database access	✓	✓	✓	✓	✓	✓
Artificial Intelligence	Dynamic Type Objects						✓
	Symbol Processing						✓
Popularity	Previous Resources						✓
	advocacy	✓	✓	✓	✓	✓	

✓* Python notes:
Easy GUIs are achived by embedding Tcl/Tk into Python and using the Tk toolkit as the Tkinter class.
Thread safety in Python is acheived with a single global lock around the bytecode interpreter, which can have scaling problems on multiprocessors. Blocking I/O calls are made outside the lock.



SOAP/XML Web Service and Client Applications in C and C++

SOAP(Simple Object Access Protocol):

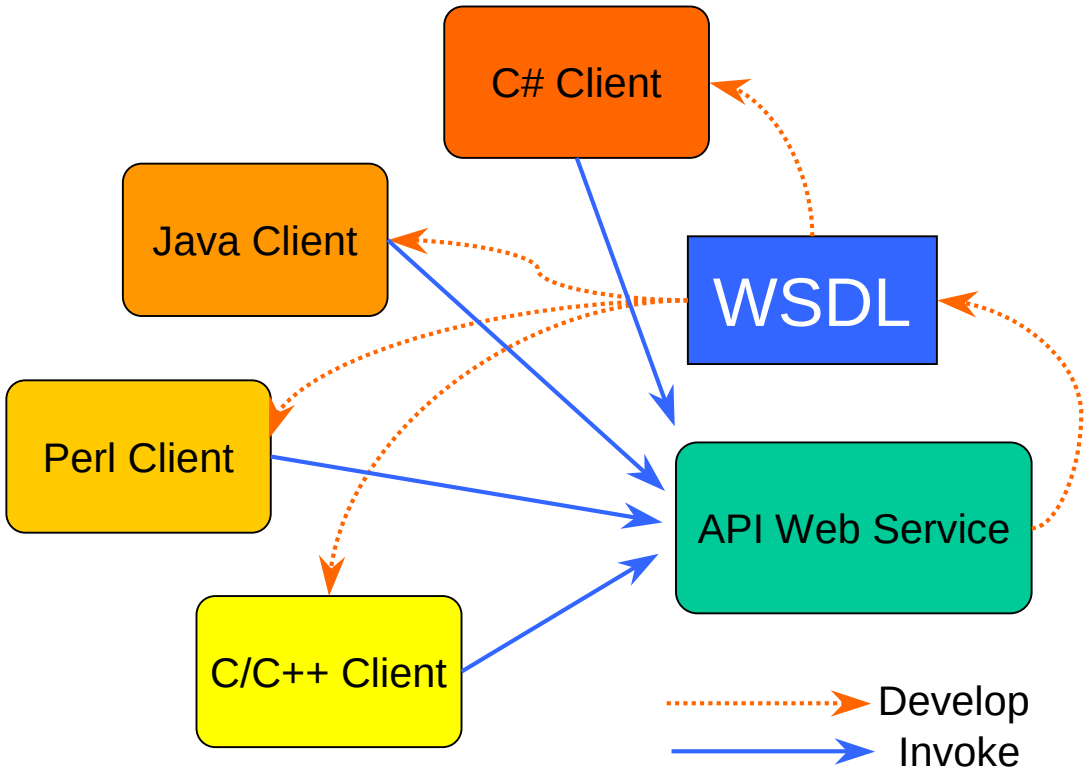
Is a versatile,simple and light-weight message exchange format in a distributed environment.
SOAP encapsulates **RPC** calls using the extensibility and flexibility of XML.The XML-based protocol is language and platform neutral, which means that information sharing relationships can be initiated among disparate parties, across different platforms, languages and programming environments.

SOAP is not a competitive technology to component systems and object-request broker architectures such as the **CORBA** component model and **DCOM**, but rather complements these technologies.
CORBA, DCOM, and Enterprise Java enable resource sharing within a single organization while SOAP technology aims to bridge the sharing of resources among disparate organizations possibly **located behind firewalls**.
SOAP applications exploit a wire-protocol (typically HTTP) to communicate with Web Services to retrieve dynamic content.

For example, This allows real-time ``**what-if''** scenarios and enables the development of agents that access real-time information. Other examples are control and visualization of large-scale simulations from a desktop computer, the sharing of laboratory results using cell phones, remote database access, and science portals.

Interoperability

SOAP is a language- and platform-neutral RPC protocol that adopts XML as the marshalling format. SOAP applications typically use the **firewall-friendly** HTTP transport protocol.



Ubiquity. The SOAP protocol and its industry-wide support promises to make services available to users anywhere, e.g. in cellphones, pocket PCs, PDAs, embedded systems, and desktop applications.

Simplicity. SOAP is a light-weight protocol based on XML. An example of a simple SOAP service is a sensor device that responds to a request by sending an XML string containing the sensor readout. This device requires limited computing capabilities and can be easily incorporated into an embedded system.

Services. SOAP Web Services are units of application logic providing data and services to other applications over the Internet or intranet. A Web Service can be as simple as a shell or Perl script that uses the Common Gateway Interface (CGI) of a Web server such as Apache. A web service can also be a server-side ASP, JSP, or PHP script, or an executable CGI application written in a programming language for which an off-the-shelf XML parser is available.

Transport. A SOAP message can be sent using HTTP, SMTP, TCP, UDP, and so on. A SOAP message can also be routed, meaning a server can receive a message, determine that it is not the final destination, and route the message elsewhere. During the routing, different transport protocols can be used.

Security. SOAP over HTTPS is secure. The entire HTTP message, including both the headers and the body of the HTTP message, is encrypted using public asymmetric encryption algorithms. SOAP extensions that include digital signatures are also available. Single sign-on (authentication) and delegation mechanisms required for Grid computing can be easily built on top of SOAP.

Firewalls. Firewalls can be configured to selectively allow SOAP messages to pass through, because the intent of a SOAP message can be determined from the message header.

Compression. HTTP1.1 supports gzipped transfer encodings, enabling compression of SOAP messages on the fly.

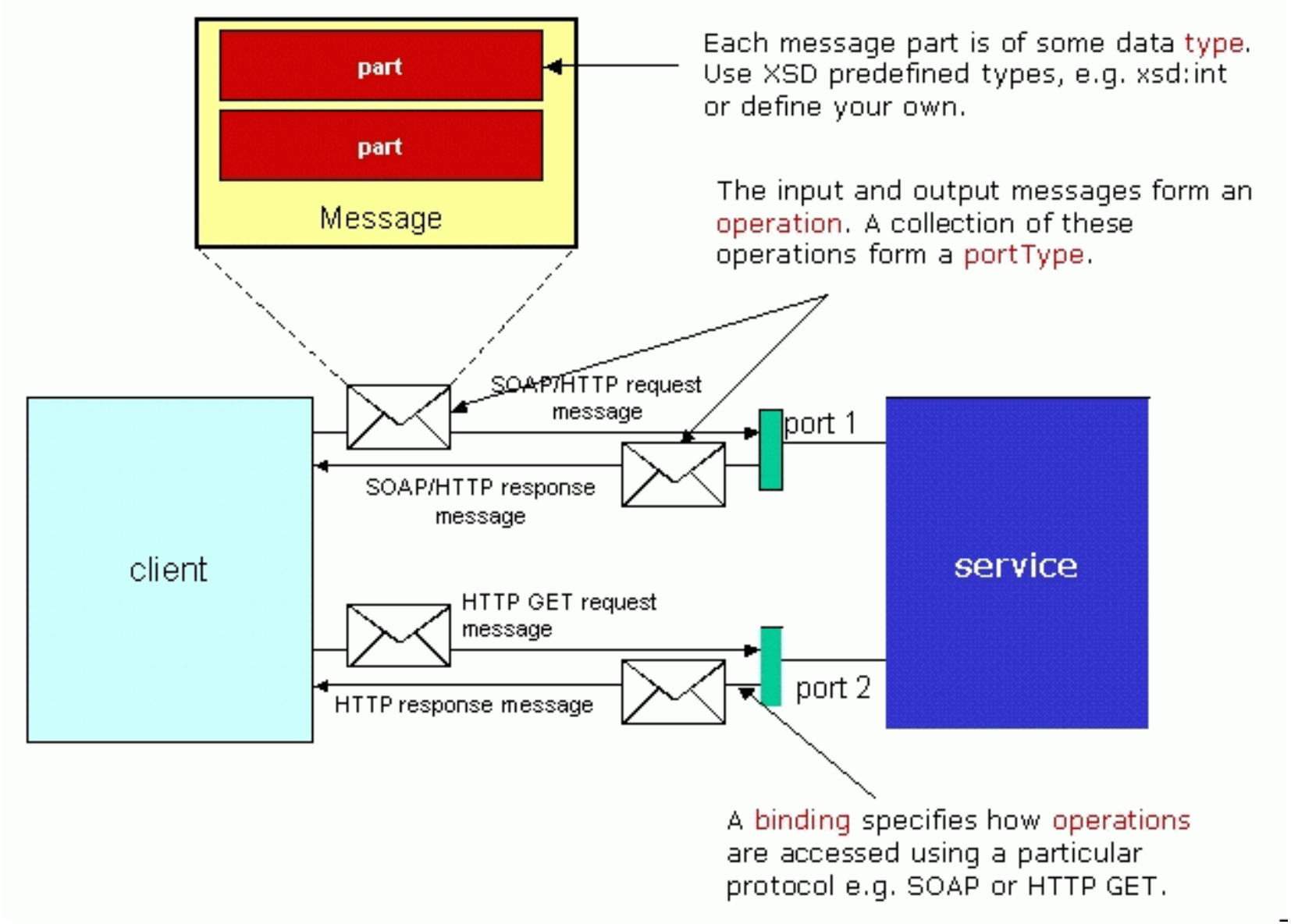
Transactions. SOAP provides transaction-handling capabilities through the SOAP message header part. The transaction-handling capabilities allow a state-full implementation of a Web Service by a server-side solution utilizing local persistent storage media.

Exceptions. SOAP supports remote exception handling.

The possible disadvantages of SOAP are:

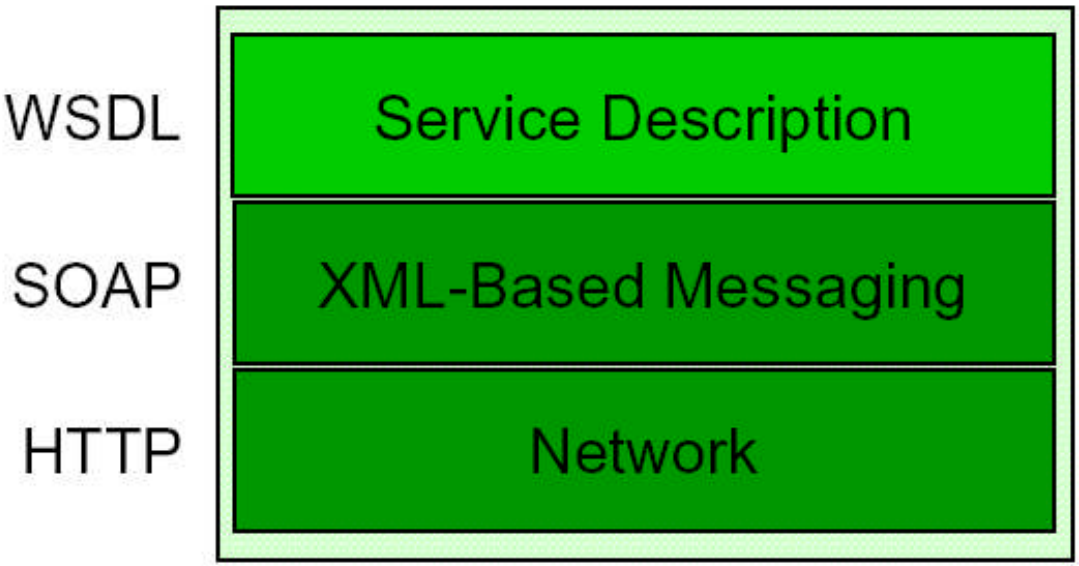
GC. The absence of mechanisms for distributed garbage collection (GC) and the absence of objects-by-reference.

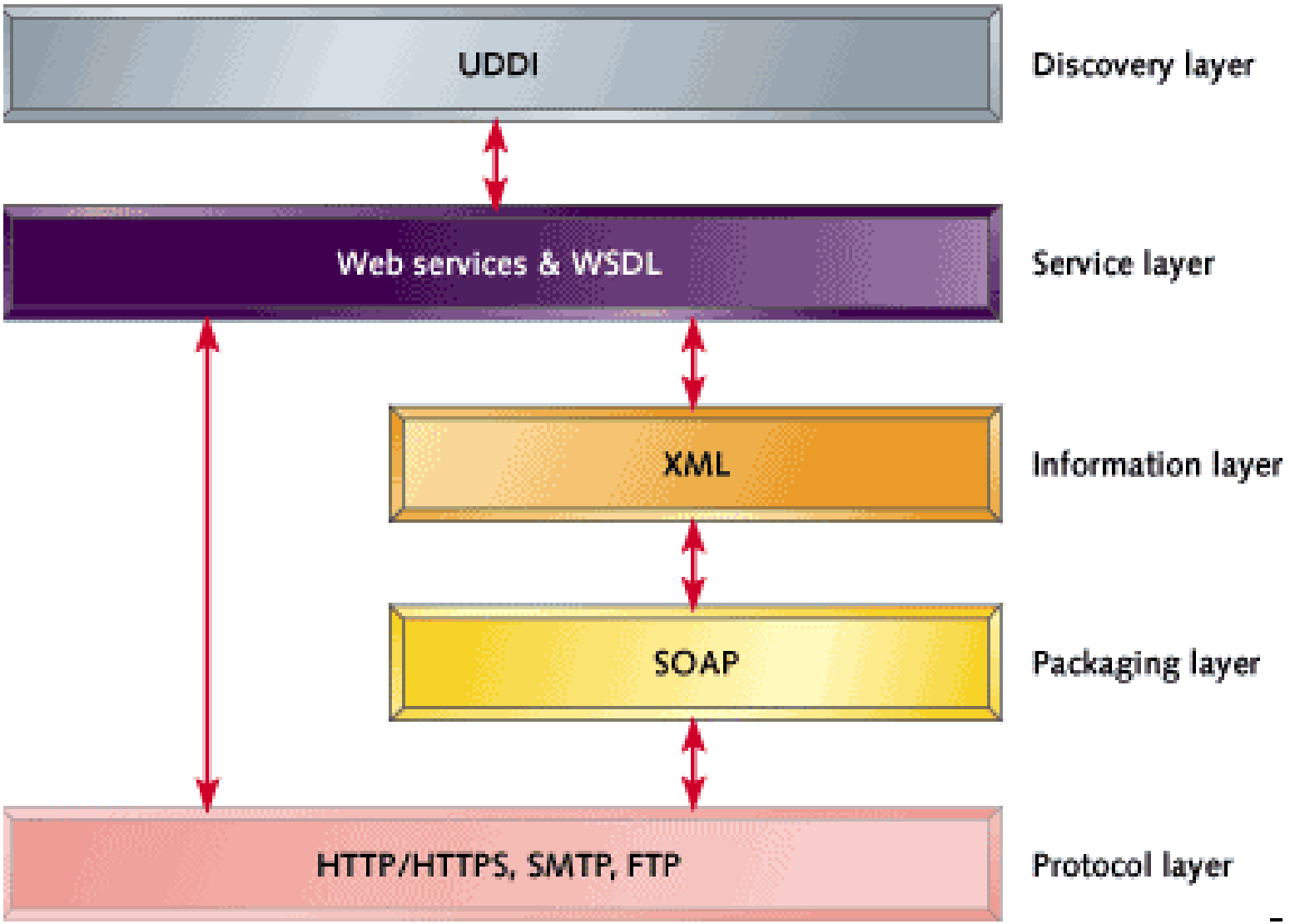
Floats. Floats and doubles are represented in decimal (text) form in XML, which can possibly contribute to a loss of precision. Other SOAP encodings such as hexBinary and Base64 can be used to encode e.g. IEEE 754 standard floating point values, but this may hamper the interoperability of systems that use other floating point representations.

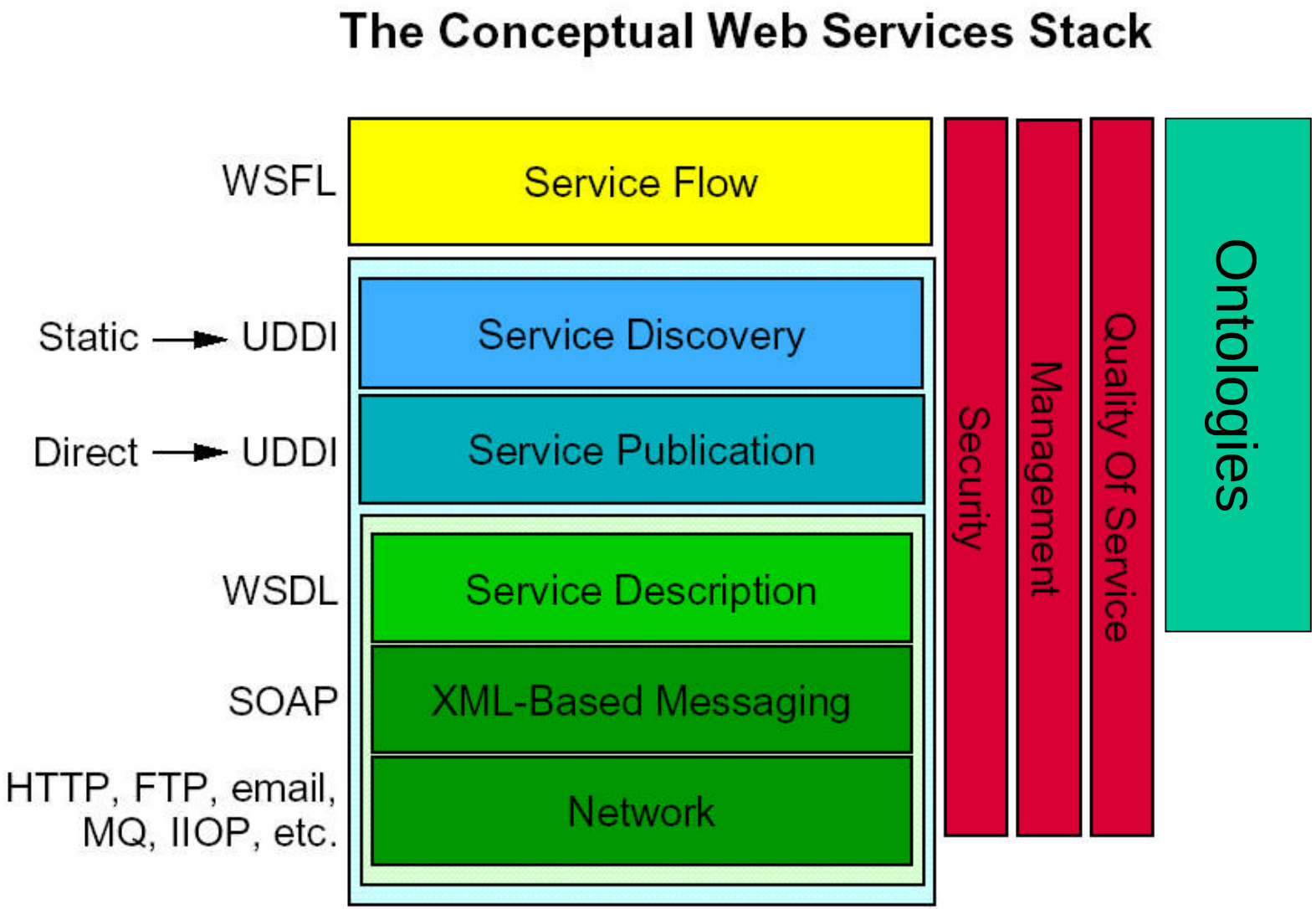

part
part
Message
Each message part is of some data type. Use XSD predefined types, e.g. xsd:int or define your own.
The input and output messages form an operation. A collection of these operations form a portType.
SOAP/HTTP request message
port 1
SOAP/HTTP response message
client
service
HTTP GET request message
port 2
HTTP response message
A binding specifies how operations are accessed using a particular protocol e.g. SOAP or HTTP GET.

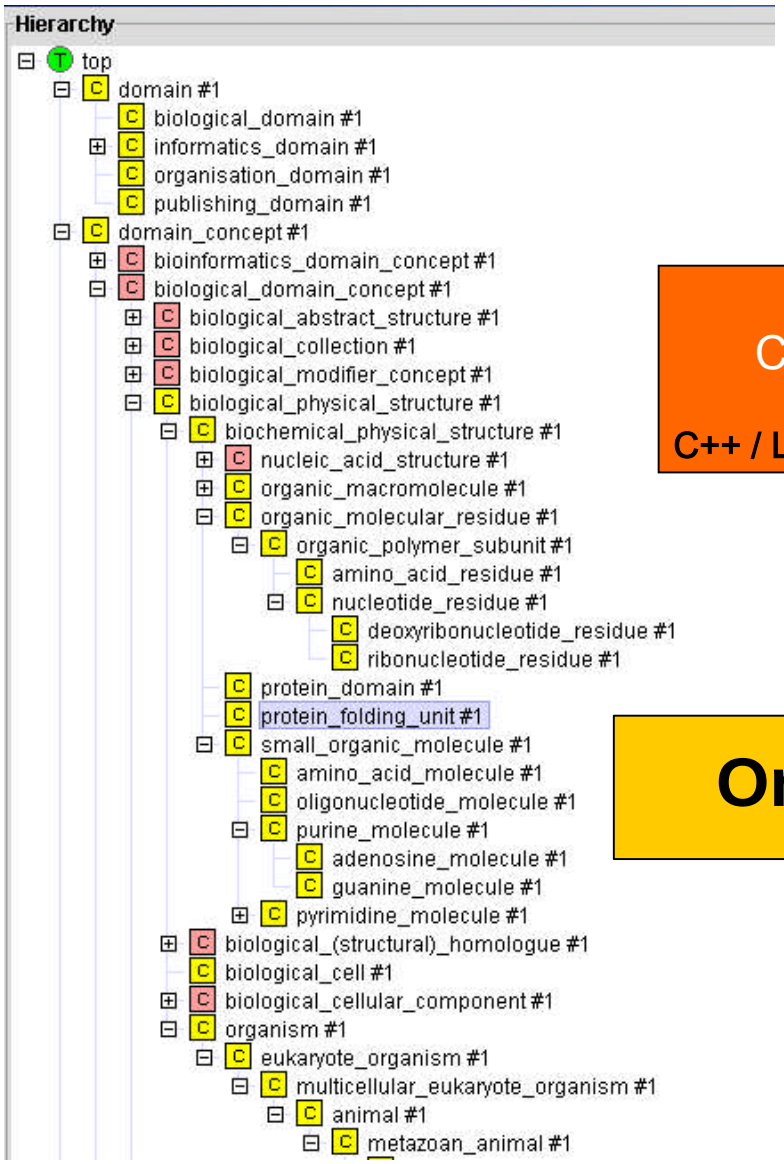
WSDL. The Web Service Description Language is an XML format for describing network services as abstract collections of communication endpoints capable of exchanging structured information. The platform- and language-neutral WSDL descriptions published by Web Services enable the automatic generation of SOAP stubs for the development of clients within a specific programming environment. The language-specific stubs can be used to invoke the remote methods of the Web Service.

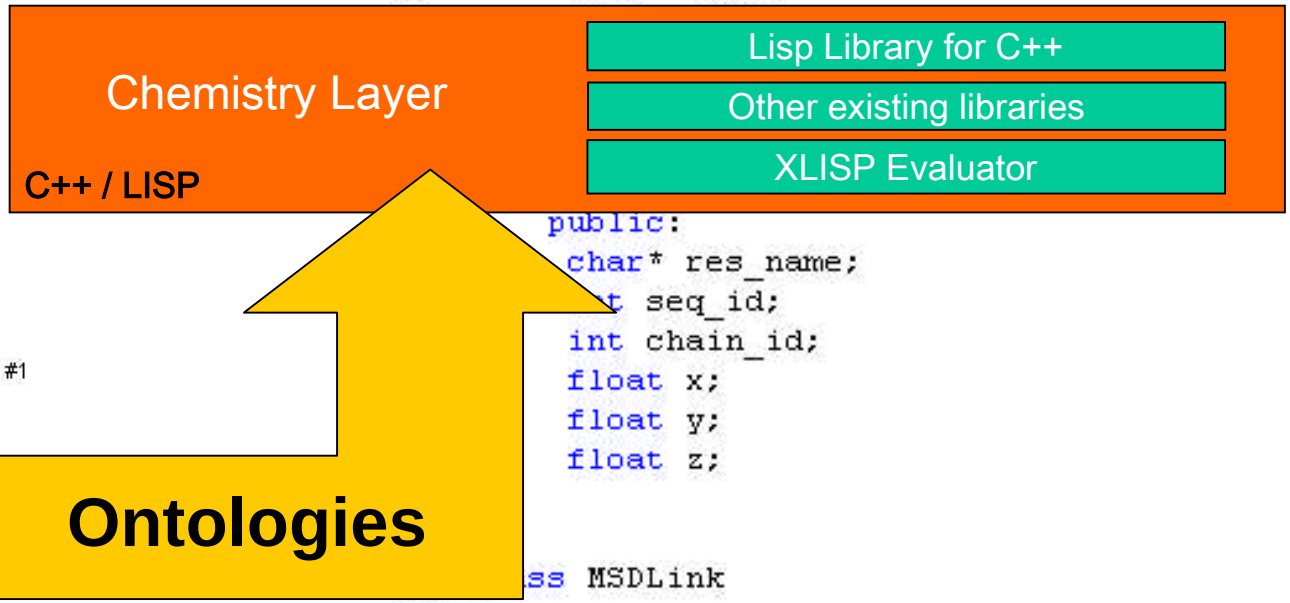
UDDI. The Universal Description, Discovery, and Integration specification provides a universal service for registry, lookup, discovery, and integration of world-wide business services. WSDL descriptions complement UDDI by providing the abstract interface to a service.


WSDL
Service Description
SOAP
XML-Based Messaging
HTTP
Network


UDDI
Discovery layer
Web services & WSDL
Service layer
XML
Information layer
SOAP
Packaging layer
HTTP/HTTPS, SMTP, FTP
Protocol layer


The Conceptual Web Services Stack
WSFL
Service Flow
Static → UDDI
Service Discovery
Direct → UDDI
Service Publication
WSDL
Service Description
SOAP
XML-Based Messaging
HTTP, FTP, email,
MQ, IIOP, etc.
Network
Security
Management
Quality Of Service
Ontologies






```
26
27
28  class MSDResidueChain
29  {
30      public:
31       char* name;
32       char* dssp;
                    public:
                    char* res_name;
                    t seq_id;
                     int chain_id;
                     float x;
                     float y;
                     float z;


                ss MSDLink
48  {
49      public:
50          char* res1;
51          char* res2;
52          char* type;
53  };
54  class MSDRsidueNeighbours
55  {
56    public:
57    int residue_id;
58    int entry_id;
59    float radius;
```

Find Entry ID's where Glucose (3-letter-code = GLC) has

GLC hbond LINK to ARG

and GLC hbond LINK to ASN

and GLC "ring" interacting with "ring" of a TRP

and ARG at Terminus of a STRAND

and ASN not in a HELIX

Then

For each Entry_ID get list of Residue_ID's that are

within 10Ang of the centre_of_gravity of the GLC molecule

in the Entry

Then

Foreach Entry_ID and foreach Residue_ID get ATOM properties

```
26
27
28  class MSDResidueChain
29  {
30      public:
31       char* name;
32       char* dssp;
33       int strand_id;
34       int seq_num;
35  };
36  class MSDAtom
37  {
38      public:
39       char* res_name;
40       int seq_id;
41       int chain_id;
42       float x;
43       float y;
44       float z;
45
46  };
47  class MSDLink
48  {
49      public:
50          char* res1;
51          char* res2;
52          char* type;
53  };
54  class MSDRsidueNeighbours
55  {
56    public:
57    int residue_id;
58    int entry_id;
59    float radius;
```

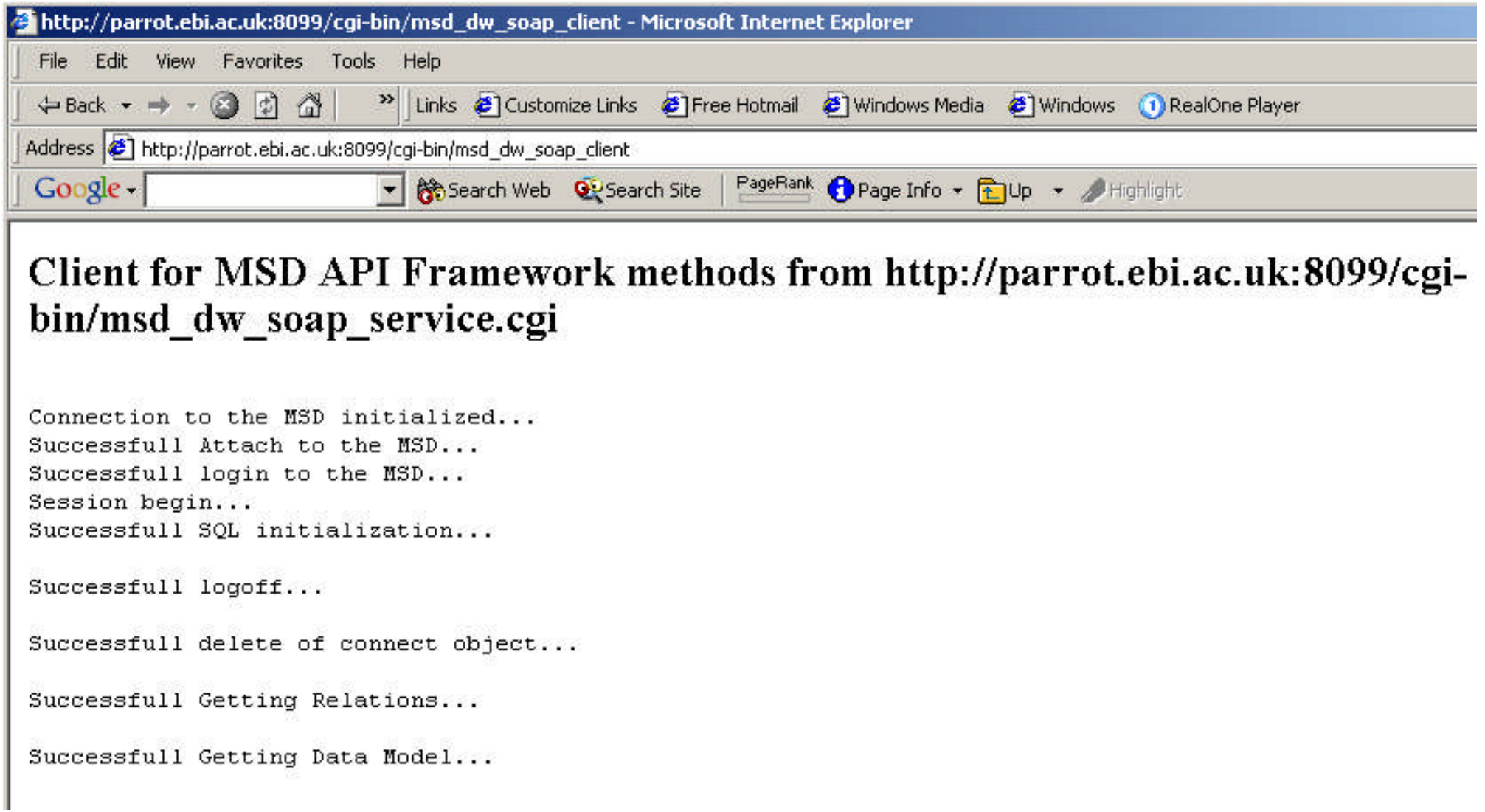
```
    ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+----1----+----
 1  #include "soapH.h"
 2  #include "msd_dw_soap_service.nsmap"
 3
 4  const char msd_dw_soap_service[] = "http://parrot.ebi.ac.uk:8099/cgi-bin/msd_dw_soap_service.cgi";
 5
 6  int main(int argc, char **argv)
 7  { struct soap *soap;
 8    soap= soap_new();
 9     if (! soap)
10    {
11     printf ("\nCan not allocate environment for Web services.\n");
12     return (1);
13    }
14    int result;
15    char* logstr="whouse/whouse@msdtrnsd";
16    char* logs;
17    logs = (char*)soap_malloc(soap, strlen(logstr)+1);
18    strcpy(logs, logstr);
19    char* sql;
20    char* statement="select /*+ INDEX_COMBINE(component component_id) */ *  from component where "
21                    "component_id >= :f<int> and component_id <= 50000*:f<int>";
22
23    sql = (char*)malloc(strlen(statement)+1);
24    strcpy(sql, statement);
25
26    int nomtabs=35;
27    printf("Content-type: text/html\r\n\r\n<html><h2>Client for MSD API Framework methods from %s</h2><pre>\n", msd_dw_so
28
29    if (soap_call_ns__c_msd_conninit(soap, msd_dw_soap_service, "urn:msd_dw_soap_service#c-msd-conninit",0,result)== SOAP_
30    { printf ("\nConnection to the MSD initialized...\n");
31    }
32    else
33    { printf ("Connection failed to the MSD...(Error %d) \n", result);
34      soap_print_fault(soap, stderr);
35      soap_print_fault_location(soap, stderr);
36    }
37
38    if (soap_call_ns__c_msd_server_attach(soap, msd_dw_soap_service, "urn:msd_dw_soap_service#c-msd-server-attach",0,"msd
39    { printf ("Successfull Attach to the MSD...\n");
40    }
41    else
42    {
43      printf ("Server not attached.(Error %d) \n", result);
44      soap_print_fault(soap, stderr);
45      soap_print_fault_location(soap, stderr);
46    }
47    if (soap_call_ns__c_msd_session_begin(soap, msd_dw_soap_service, "urn:msd_dw_soap_service#c-msd-session-begin",0,"who
48    { printf ("Successfull login to the MSD...\nSession begin...\n");
49    }
50    else
51    { printf ("Login failed to the MSD...(Error %d) \n", result);
```

```
#include "soapH.h"
#ifndef __MSD_DW_USER_H__
#include "msd_dw_user.h"
#endif
#include "msd_dw_soap_service.nsmap"
#define LEASETERM 60
#include <iostream>
#include <sys/stat.h> // for open()
using namespace std;
const char msd_dw_soap_service[] = "http://parrot.ebi.ac.uk:8099/cgi-bin/msd_dw_soap_service.cgi";
int main(int argc, char **argv)
{
   int m, s;
   struct soap soap;
   MSDConnect conn;
   soap_init(&soap);
   soap.user = (void*)&conn;        // associate factory with run-time
   soap.accept_timeout = LEASETERM; // term before leases runs out is used as a rate to purge objects
   soap.cookie_domain = "http://parrot.ebi.ac.uk:8099";
   soap.cookie_path = "~/apache/cgi-bin/"; // the path which is used to filter/set cookies with this destination
   int cookie_max =64;
   if (argc < 3)
    {
       soap_getenv_cookies(&soap); // CGI app: grab cookies from 'HTTP_COOKIE' env var
       soap_serve(&soap); // run as CGI application over the Web
    }
  else // run as stand-alone server on machine given by argv[1] listening to port argv[2]
  { m = soap_bind(&soap, argv[1], atoi(argv[2]), 100);
    if (m < 0)
    { soap_print_fault(&soap, stderr);
      exit(-1);
    }
    fprintf(stderr, "Socket connection successful: master socket = %d\n", m);
    for (int i = 1; ; i++)
    { s = soap_accept(&soap);
      if (s < 0)
      { soap_print_fault(&soap, stderr);
        exit(-1);
      }
      fprintf(stderr, "%d: accepted connection from IP = %d.%d.%d.%d socket = %d ... ", i, (int)(soap.ip>>24)&0xFF, (int)(soap.ip>>16)&0xFF,
      soap_serve(&soap);        // process request
      fprintf(stderr, "request served\n");
      soap_destroy(&soap);  // delete class instances
      soap_end(&soap);      // clean up everything and close socket
      soap_free_cookies(&soap); // remove all old cookies from database so no interference occurs with the arrival of new cookies
    }
  }
  soap_destroy(&soap);
  soap_end(&soap);
  //free(&soap);
```

```
54    return 0;
55  }
56
57  int ns__c_msd_create_connect_obj(struct soap *soap, int ConID, int &result){
58   result = c_msd_create_connect_obj(ConID);
59   return SOAP_OK;
60  }
61
62  int ns__c_msd_select_atom_data(struct soap *soap, int ConID, int SqlID, char* whereclause, int &result){
63   result = c_msd_select_atom_data(ConID, SqlID, whereclause);
64   return SOAP_OK;
65   }
66
67  int ns__c_msd_initialize(struct soap *soap, int ConID, int &result){
68   result = c_msd_initialize(ConID);
69   return SOAP_OK;
70   }
71  int ns__c_msd_conninit(struct soap *soap, int ConID, int &result){
72
73     MSDConnect* pconn = (MSDConnect *)soap->user;     // get handle from gSOAP environment
74
75    if (!soap->header)
76      soap->header = (struct SOAP_ENV__Header*)soap_malloc(soap, sizeof(struct SOAP_ENV__Header));
77    if (soap->header)
78    {
79      pconn->MSDInit(ConID);
80
81      result =0;
82
83    }
84    else
85      result = 1;
86    return SOAP_OK;
87  }
```

```
/*-------------------------------------------------------
 msd_dw_main.c

 Project: MSD API Framework
 Module:  Sample of main program for calling api

 To build:
    make -f msd_api_fw.mk buildc++ NAME=msd_dw_main

 Last updated: 18 April 2002 10:15
 S. Sobhany
------------------------------------------------------- */
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "msd_dw_user.h"

int main(int argc, char* argv[]) {
  double difs;
  time_t starts;
  time_t ends;
  int cr=1;
  int i=1;
  void* vcr=&cr;
  int     iparam=0;     /* integer parameter */
  double  dparam=0;     /* double parameter */
  char  sparam[16]="";    /* string parameter */
  int     fparam=0;     /* boolean flag */
  int xargc;
  char** xargv;
  char* usern="whouse";
```

```
89
90      while (argc > 0){
91       printf (*argv);
92       printf ("\n");
93       argv++;
94       argc--;
95      }
96    starts = time (NULL);
97    c_msd_lisp_init(xargc, xargv);
98    c_msd_lisp_load("xlisp.lsp");
99    printf ("Process starts...\n");
100   if (c_msd_conninit(0)==0)
101   {
102        printf ("\nConnection to the MSD initialized.\n");
103   }
104    if (c_msd_server_attach(0,"msdtrnsd")==0)
105   {
106        printf ("\nAttached to 'msdtrnsd' server successfully.\n");
107   }
108   if (c_msd_session_begin(0,un,pass)==0)
109   {
110     printf ("\nLogin to 'msdtrnsd' successfully.\n");
111   }
112   sql = (char*)malloc(strlen(statement)+1);
113   strcpy(sql, statement);
114   c_msd_sqlinit(0,0, sql);
115   c_msd_setcriteria(0,0,"i",1);
116   c_msd_makerecord(0,0);
117   while (c_msd_gethits(0,0)<3)
118   {
119      c_msd_getnext(0,0);
120   }
121   printf ("Second SQL initializing...\n");
```


http://parrot.ebi.ac.uk:8099/cgi-bin/msd_dw_soap_client - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Links Customize Links Free Hotmail Windows Media Windows RealOne Player
Address http://parrot.ebi.ac.uk:8099/cgi-bin/msd_dw_soap_client
Google
Search Web Search Site PageRank Page Info Up Highlight
Client for MSD API Framework methods from http://parrot.ebi.ac.uk:8099/cgi-bin/msd_dw_soap_service.cgi
Connection to the MSD initialized...
Successfull Attach to the MSD...
Successfull login to the MSD...
Session begin...
Successfull SQL initialization...
Successfull logoff...
Successfull delete of connect object...
Successfull Getting Relations...
Successfull Getting Data Model...

```
EBI-MSD's API Framework 1.00, 17 January 2002

Common LISP Compatibility module loaded.Process starts...

Connection to the MSD initialized.

Attached to 'msdtrnsd' server successfully.

Login to 'msdtrnsd' successfully.
COMPONENT_ID
2478
ENGINEERED
YES
MUTANT_FLAG

FRAGMENT_FLAG
TYROSINE KINASE DOMAIN
MUTATION_STRING
Y
SYNTHETIC
N
PREV_ID
0
NEXT_ID
0
PARENT_ENTITY_ID
8696
DE_ID
8697
DE_TYPE
DPES1
TAX_ID
9606
CHAIN_ID
2110
DEP_ID
2013
ACCESSION_CODE
1agw
AUTH_ID
ALA
AUTH_SEQ
512
AUTH_INSERT_CODE

PDB_ID
ALA
PDB_SEQ
512
PDB_INSERT_CODE
```

Supporting platforms and compilers:

g++ gcc 3.x on Unix, Linux all platforms and cygwin for Windows 32

Sun OS 5.0 and Compaq Tru64 Native compilers