

**KRISTAL-2002**

**Programmer's Manual for C++ User**

**GIIS/KISTI**

**KRISTAL-2002**

.

**Copyright 2000 - 2004 Group for Intelligent Information Systems, KISTI. All right reserved.**

**If you have any question or comment for this manual, please mail to [giis@kisti.re.kr](mailto:giis@kisti.re.kr).**

---

<b>1</b>	<b>C++</b>	<b>1</b>
1.1		1
<b>2</b>	<b>C++ API</b>	<b>3</b>
2.1		3
2.1.1		3
2.1.2		14
2.2		20
2.3	<b>API</b>	<b>26</b>
	GET_DB_INFO	27
	GET_META_INFO_QUERY	32
	GET_SET_INFO	35
	RETRIEVE	38
	RETRIEVE_SIMILAR_DOCUMENTS	42
	RETRIEVE_IN_RESULT	45
	GET_DOCUMENTS_FROM_RESULT	47

---

GET_DOCUMENTS_WITH_IDS .....	51
GET_DOCUMENTS_WITH_PRIMARY_KEY .....	54
GET_DOCUMENT_ID_WITH_DOCUMENT .....	57
BROWSE_ALL_DOCUMENTS.....	59
CALCULATE.....	61
SORT_BY_SECTION.....	63
PROCESS_DATABASE_SCHEMA.....	65
APPEND_DOCUMENT.....	67
DELETE_DOCUMENT .....	69
UPDATE_DOCUMENT .....	71
CHECK_STATUS .....	73
SAVE_USER_LOG.....	75
GET_XML_NODES_FROM_RESULT .....	77
GET_XML_NODES_WITH_IDS .....	81
GET_XML_NODES_INFO .....	85
GET_XML_TREE.....	87
REPRODUCE_XML_NODE.....	89

---

RETRIEVE_DOCUMENTS_WITH_FOREIGN_KEY .....	91
APPEND_XML_NODE .....	93
UPDATE_XML_NODE .....	95
DELETE_XML_NODE .....	97
MOVE_XML_NODE .....	99
MULTIPLE_RETRIEVE .....	101

---



# 1 C++

KRISTAL-2002

,

.

,

.

Compile

Header

Library

.

## 1.1

□ Header

KRISTAL-2002

API

ClientLib.h

include

API

.

가

(kshare.h)

.

□ (link)

*-lclient -lcom -lshare -lxcercs-c*

가

.

*-liconv*

가

.

가

.

- LINUX

-liberty -lm -ldl -lstdc++ -lnsl -lresolv

- Sun

-liberty -lm -ldl -lstdc++ -lnsl -lresolv -lsocket

---



## 2 C++ API

### 2.1

KRISTAL-2002 , ,  
가

Parameter , , ,

\$KRISTAL\_HOME/include/com/ObjectClasses.h .

#### 2.1.1

□ Parameter ( )

class Cparameter\_t

```
{
    public:
        vector<Cschema_info_t> schemas; // vector
        vector<Cdocument_t> documents; // vector
        vector<Csection_t> sections; // vector
        vector<string> section_namelist; // vector
        vector<CXML_node_t> nodes; // XML node vector
        vector<Cknowledge_document_t> knowledge_documents; //
        // knowledge vector
        vector<Cdisplay_t> displays; // ,
```

```

unsigned int start_position; //
unsigned int counter; //
int direction; // primary key ( , , , )
string primary_key; // primary key
bool order; // ( order=DESCENDING)
bool sorting_key_type; // (String, Integer)
bool flag; // XML system section 가

CXML_node_t pivot_node; // XML
unsigned int minDF; // DF
unsigned int maxDF; // DF
string delimiter; //
string target_section; //
string schema_text; // schema // text
unsigned int top_document_no; //
// threshold
vector<string> table_namelist; // vector
int space_operator; //
string query; //
int term_expansion; //
int remove_chars_word; //
vector<int> thesaurus_levels; //

string extended_query; //
vector<Csection_t> stopwords; // ( , )
vector<Csection_t> terms; // ( , )

```

---

```
    unsigned int estimated_totoal_df; //
    Cintegrate_retrieve_t integrate; //

    unsigned int set_id; // ID
    unsigned int total_df; //
    vector<Cdocument_group_t> document_groups; //

    vector<string>return_values; //
    int method; //
    float similarity_value; // (0.0 ~ 1.0)
    long unsigned int service_time; //

    int server_type; //
    string server_version; //
    string db_description; // DB
    int errcode; //
    string errmsg; //

public:
    Cparameter_t &operator=(const Cparameter_t &parameter2);
    Cparameter_t();
};

□

class Cschema_info_t
```

---

```
{
    public:
        string volume_dir;        // DB
        string schema_name;       //
        bool schema_type;        //          true:xml false: plain
        bool aux_flag;           // true:aux false:no aux
        long unsigned int total_document_no; //
        vector<Ctable_info_t> tables; //
        vector<Cbasic_section_t> basic_sections; //
        vector<Cvirtual_section_t> virtual_sections; // ㄱ
        vector<Cunion_section_t> union_sections; //
        vector<Cprimary_section_t> primary_sections; //
};
```

□

```
class Ctable_info_t
```

```
{
    public:
        string table_name;       //
        short int table_id;      //          id
        long unsigned int document_no; //
};
```

□ (Basic)

---

```
class Cbasic_section_t
{
    public: // basic section
        string section_name;    //
        string data_type;      //
        string index_type;     //
        string default_value;  //
        bool stemming;        //
        bool hanja;           //
        bool stopword;        //
};
```

□ 가 (Virtual)

```
class Cvirtual_section_t
{
    public:
        string virtual_section_name;    // 가
        vector<string> original_section_namelist; //
        string index_type;             //
        bool stemming;                 //
        bool hanja;                     //
        bool stopword;                 //
};
```

□ (Union)

---

```
class Cunion_section_t
{
    public:
        string union_section_name;    //
        vector<string> section_namelist; //
};
```

□ (Primary)

```
class Cprimary_section_t
{
    public:
        string primary_section_name; //
};
```

---

---

□ (Document)

```
class Cdocument_t
{
    public:
    unsigned int document_id;        //      ID
    short int table_id;             //      ID
    float weight;                   //
    string primary_key;             //
    vector<Csection_t> sections;     //

    //
    string getSectionValue(const string& section_name);
};
```

□ (Section)

```
class Csection_t {
    public :
        string section_name;        //
        string section_value;      //
        vector<unsigned int> offset_list; //      offset
        vector<unsigned int> length_list; //
    public :
        Csection_t();
};
```

---

```
Csection_t(string section_name, char* section_value, unsigned int  
    section_value_size);3  
int trans_binvalue(char* &section_value, unsigned int&  
    section_value_size);4  
};
```

---

<sup>3</sup> Binary . Section\_value

<sup>4</sup> Binary .

---



□ XML

```

class CXML_node_t
{
    // object type depend member variables Here./
public:
    bool search_flag;        // true        node,
                            // false       tree    node
    short int table_id;     //        ID
    string primary_key;    //
    vector<Csection_t> sections;        //        ,
    CXML_system_section_t system_section; // XML
public:
    CXML_node_t();
};

```

□ XML

```

class CXML_system_section_t
{
public:
    unsigned int node_id;    //        ID (default)5
    string title;           //        (default)

```

---

<sup>5</sup> Default

flag 가 true 가 .

---

```
int level; // (default)

string type;
unsigned int parent_node_id; // ID
unsigned int left_node_id; // ID
unsigned int right_node_id; // ID
unsigned int firstchild_node_id; // ID
int order;

public:
    CXML_system_section_t();
};
```

#### □ Display

```
class Cdisplay_t
{
    string section_name;
    int highlight_method;
    string start_tag; //
    string end_tag; //
    int summary_method; //
    unsigned int summary_length; //
};
```

---

□

```
class Cintegrate_retrieve_t
{
public:
    bool is_used;           //
    bool group_type;       //
    vector<string> section_namelist; //

    bool sorting_key_type; //
    bool order;            //
    unsigned int counter;  //      가

public:
    Cintegrate_retrieve_t();
};
```

□

```
class Cdocument_group_t
{
public:
    df_per_group;         //
    vector<Cdocument_t> documents; //

public:
    Cdocument_group_t();
};
```

---

## 2.1.2

□ (method)

```
#define BOOLEAN_MODEL 0
#define VECTOR_BOOLEAN_MODEL 1
#define VECTOR_HASH_MODEL 2
#define VECTOR_HASH_PRUNING_MODEL 3
```

□ (direction)

```
#define FIRST_NODE 0 //
#define LAST_NODE 1 //
#define PREVIOUS_NODE 2 //
#define NEXT_NODE 3 //
```

□ XML Node (direction)

```
#define XML_ROOT_NODE 0 // Root
#define XML_PARENT_NODE 1 //
#define XML_LEFT_NODE 2 //
#define XML_RIGHT_NODE 3 //
#define XML_CHILD_NODE 4 //
```

---

```
#define XML_SIBLING //
#define XML_ANCESTOR //
```

□ Space ( )

(space\_operator)

```
#define SPACE_AND 0
#define SPACE_OR 1
#define SPACE_NOT 2
#define SPACE_WITHIN 3
#define SPACE_NEAR 4
```

□ (order)

```
#define ASCENDING FALSE //
#define DESCENDING TRUE //
```

□ ( )

(sorting\_key\_type)

```
#define K_STRING false //
#define K_NUMBER true //
```

---

□ (schema\_type)

```
#define XML_SCHEMA true // XML
#define PLAIN_SCHEMA false // Plain
```

□ (displays.highlight\_method)

```
LISTING_LOCATION offset length
, INSERTING_TAG 가
.
EACH
, ALL
.
```

```
#define NONE 0 //
#define LISTING_LOCATION_EACH 1
#define INSERTING_TAG_EACH 2
#define LISTING_LOCATION_ALL 3
#define INSERTING_TAG_ALL 4
```

□ (displays.summary\_method)

```
displays
. summary_method , summary_length
. NORMAL_SUMMARY
.
```

```
#define NONE 0 //
```

```
#define NORMAL_SUMMARY 1
```

```
□ (term_expansion)
```

```
KRISTAL-2002
```

```
가 . .6
```

```
#define NO_QUERY_TERM_EXPANSION 0 //
```

```
#define PARTIAL_QUERY_TERM_EXPANSION 1 // ""
```

```
#define FULL_QUERY_TERM_EXPANSION 2 //
```

```
NO_QUERY_TERM_EXPANSION .
```

```
가 . ( : ' '
' ' 가 .)
```

```
FULL_QUERY_TERM_EXPANSION
```

```
. ( : INDEX_BY_MA
' ' 가 ' '
' ' )
```

```
PARTIAL_QUERY_TERM_EXPANSION
```

```
FULL_QUERY_TERM_
```

```
EXPANSION ("")
```

```

INDEX_BY_MA/INDEX_BY_TOKEN . “
    ” FULL_QUERY_TERM_EXPANSION ((( /W1
) | ) /W1 ) PARTIAL (
/W1 ) . PARTIAL

```

□ (remove\_chars\_word)

가 가 .  
<sup>7</sup>

```
#define NO_REMOVE_CHARS_WORD 0 //
```

```
#define REMOVE_SINGLE_CHAR_WORD 1 //
```

□ (integrate.group\_type)

가 가 .  
가

```
#define TABLE_GROUP 0 //
```

<sup>7</sup> 2 .



```
#define SCHEMA_GROUP 1 //
```

```
□ (query)
```

```
, , , ,
```

```
. int float .
```

```
#define MAX "MAX"
```

```
#define MIN "MIN"
```

```
#define AVG "AVG"
```

```
#define SUM "SUM"
```

```
#define CNT "COUNT"
```

---

## 2.2

(Client Library) Class

```
class ClientLIB
{
    private:
        string m_ipaddr;
        int m_port;
        Message *m_msg;
    public:
        ClientLIB(const string ip_addr, int port);
        ~ClientLIB();
        void SetAddress(const string ip_addr, int port);
        int Request(const string process,
                   Cparameter_t &in_parameter,
                   Cparameter_tt &out_parameter);
        string getVersion();
};
```

ClientLIB Class

Private

---

**SetAddress(const string ip\_addr, int port)**

- ip .

**Request(const string process, Cparameter\_t &in\_parameter,  
Cparameter\_t &out\_parameter)**

- Kristal .

**getVersion()**

- .

Request .  
,  
가 getVersion  
. 2002.1.x.x  
. (  
)

---

```

Request
KRISTAL (daemon) , (table) 가 C++
. IP
"127.0.0.1", "50000" .

#include <ClientLib.h>
#include <ObjectClasses.h>

1 int main ()
2 {
3     ClientLIB c("127.0.0.1", 50000);
4     Cparameter_t in, out;
5     int e;
6
7
8     e = c.Request(GET_DB_INFO,in,out);
9     if (e)
10    {
11        cout << "ERROR : " << out.errmsg << "(" << out.errcode << ")" <<endl;
12        return 1;
13    }
14    cout << " # of Schemas : " << out.schemas.size() << endl;
15 }

3 : ClientLIB . ClientLIB IP PORT
, IP PORT 가
.

4 : ClientLIB , (in)
(out) . Parameter

```

```

8 : ClientLIB Request . Kristal-2002
API Request
.
4 in , out
.
9 -13 : Request 가 0 가 . Kristal
가 . (
, )
KRISTAL-2002 errmsg
errcode .

```

### 2.2.1 Binary Data

```

(Blob ) KRISTAL-2002
Text . text string
가 . KRISTAL-2002 Client Library
Text

```

Csection\_t(string section\_name, char\* section\_value, unsigned int

```

        section_value_size);
        ,
        Base64
        text
        .
    )
    char* input ;
    unsigned int input_size ;
    // input   input_size   가   가

    Csection_t Asection("BINARY_SECTION", input, input_size); //

    Cdocument_t Adoc ;
    Adoc.sections.puh_back(Asection) ;

int trans_binvalue(char* &section_value, unsigned int& section_value_size);

    )

    Cdocument_t Outdoc ; //   가

    char* output ;
    unsigned int output_size ;
    string section_name=Outdoc.sections[0].section_name ;
    Outdoc.sections[0].trans_binvalue(output, output_size) ;

```

---

```
// output  
...  
::free(output) ; //
```

---

## 2.3 API

Cparameter\_t

4-15

kprograms

\$KRISTAL\_HOME/include/com/ObjectClasses.h

KRISTAL-2002 client

\*

---



**SERVICE NAME**

GET\_DB\_INFO

**DESCRIPTION**

DB Schema .  
 schema , table , Basic Section, Virtual Section, Union  
 Section, Primary Section, Database Description . Service\_Time  
 API Kristal 가 .  
 mili-second .

**IN PARAMETER****OUT PARAMETER**

```
string userarea_text; // 가 Database Schema

vector<Cschema_info_t> schemas.*;

long Service_Time;
```

**EXAMPLE**

```
//
Cparameter_t p_in, p_out;

// DB (DB Information)
// 가 .

ret_val = clientLib.Request(GET_DB_INFO, p_in, p_out);
cout << "User Area Text : [" << p_out.userarea_text << "]" << endl;
```

```

cout << "#####" << endl;
cout << " DB Information(The Total Number of Schemas: ";
cout << p_out.schemas.size() << ")" << endl;
cout << "#####" << endl;

for (int i = 0; i < p_out.schemas.size(); i++) {
    //
    cout << "=====" << endl;
    cout << "Schema Name : " << p_out.schemas[i].schema_name << endl;
    cout << "Volume Directory : " << p_out.schemas[i].volume_dir << endl;
    cout << "Tables List : ";
    for (int j = 0; j < p_out.schemas[i].tables.size(); j++) {
        cout << p_out.schemas[i].tables[j].table_name;
        cout << "(" << p_out.schemas[i].tables[j].table_id << "),"";
    }
    cout << endl;

    //
    cout << "\t" << "< BASIC SECTION >" << endl;
    for (int j = 0; j < p_out.schemas[i].basic_sections.size(); j++) {
        cout << "\t-----" << endl;
        cout << "\tSECTION NAME: ";
        cout << p_out.schemas[i].basic_sections[j].section_name << endl;
        cout << "\tDATA TYPE    : ";
        cout << p_out.schemas[i].basic_sections[j].data_type << endl;
        cout << "\tINDEX TYPE    : ";
        cout << p_out.schemas[i].basic_sections[j].index_type << endl;
    }
}

```

---

---

```

        cout << "\t\tDEF. VALUE   : "
        cout << p_out.schemas[i].basic_sections[j].default_value << endl;
        cout << "\t\tSTEMMING?    : ";
        cout << p_out.schemas[i].basic_sections[j].stemming << endl;
        cout << "\t\tHANJA CONV? : ";
        cout << p_out.schemas[i].basic_sections[j].hanja << endl;
        cout << "\t\tSTOPWORD?   : ";
        cout << p_out.schemas[i].basic_sections[j].stopword << endl;
        cout << "\t\t-----" << endl;
        cout << endl << endl;
    }

// 가
    cout << "\t" << "< VIRTUAL SECTION >" << endl;
    for (int j = 0; j < p_out.schemas[i].virtual_sections.size(); j++) {
        cout << "\t\t-----" << endl;
        cout << "\t\tSECTION NAME: " ;
        cout << p_out.schemas[i].virtual_sections[j].virtual_section_name << endl;
        cout << "\t\tORG. NAME   : ";
        cout << p_out.schemas[i].virtual_sections[j].original_section_name << endl;
        cout << "\t\tINDEX TYPE   : ";
        cout << p_out.schemas[i].virtual_sections[j].index_type << endl;
        cout << "\t\tSTEMMING?    : ";
        cout << p_out.schemas[i].virtual_sections[j].stemming << endl;
        cout << "\t\tHANJA CONV? : ";
        cout << p_out.schemas[i].virtual_sections[j].hanja << endl;
        cout << "\t\tSTOPWORD?   : ";

```

---

```
    cout << p_out.schemas[i].virtual_sections[j].stopword << endl;
    cout << "\t\t-----" << endl;
    cout << endl << endl;
}

//
for (int j = 0; j < p_out.schemas[i].union_sections.size(); j++) {
    cout << "\t\t-----" << endl;
    cout << "\t\tSECTION NAME: ";
    cout << p_out.schemas[i].union_sections[j].union_section_name << endl;
    cout << "\t\tSECTION LIST: ";
    int secsize = p_out.schemas[i].union_sections[j].section_namelist.size();
    for (int k = 0; k < secsize; k++) {
        cout << p_out.schemas[i].union_sections[j].section_namelist[k] << ", ";
    }
    cout << endl;
    cout << "\t\t-----" << endl;
    cout << endl << endl;
}

//
for (int j = 0; j < p_out.schemas[i].primary_sections.size(); j++) {
    cout << "\t\t-----" << endl;
    cout << "\t\tSECTION NAME: ";
    cout << p_out.schemas[i].primary_sections[j].primary_section_name
    cout << endl;
    cout << "\t\t-----" << endl;
}
```

---

```
        cout << endl << endl;
    }

    cout << "======" << endl;
}
```

---

**SERVICE NAME**

GET\_META\_INFO\_QUERY

**DESCRIPTION**

```

,
,
,
,
.

```

**IN PARAMETER**

unsigned int set\_id;

**OUT PARAMETER**

```

string query;
string extended_query;
vector<Csection_t> stopwords; //   가   section
vector<Csection_t> terms; //   가   section
vector<string> table_namelist; //
int space_operator; //
int method; //
long unsigned int Service_Time; //

```

**EXAMPLE**

```

Cparameter_t p_in, p_out; //
p_in.set_id = set_id;

```

---

---

```
int ret_val = clientLib.Request(GET_META_INFO_QUERY, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

//

cout << "=====" << endl;
cout << "\t[      ] : " << UECONVCODE(p_out.query) << endl;
cout << "\t[      ] : " << UECONVCODE(p_out.extended_query) << endl;
cout << "\t[      ] : " << endl;
for (int i = 0; i < p_out.stopwords.size(); i++) {
    cout << "\t\t[" << UECONVCODE(p_out.stopwords[i].section_name) << "] : ";
    cout << UECONVCODE(p_out.stopwords[i].section_value) << endl;
}

cout << "\t[Term      ] : " << endl;
for (int i = 0; i < p_out.terms.size(); i++) {
    cout << "\t\t[" << UECONVCODE(p_out.terms[i].section_name) << "] : ";
    cout << UECONVCODE(p_out.terms[i].section_value) << endl;
}

cout << "\t[Table      ] : " << endl;
for (int i = 0; i < p_out.table_namelist.size(); i++) {
    cout << "\t\t[" << UECONVCODE(p_out.table_namelist[i]) << endl;
}

cout << "\t[      ] : " << UECONVCODE(p_out.method) << endl;
```

---

```
cout << "\t[          ] : " << UECONVCODE(p_out.space_operator) <<  
endl;
```

```
cout << "======" << endl;
```

---



**SERVICE NAME**

GET\_SET\_INFO

**DESCRIPTION**

GET\_META\_INFO\_QUERY  
 . GET\_META\_INFO\_QUERY  
 ,  
 ,  
 ( ) Set  
 가

**IN PARAMETER**

unsigned int set\_id;

**OUT PARAMETER**

```
string query;
string extended_query;
vector<Csection_t> stopwords; // 가 section
vector<Csection_t> terms; // 가 section
vector<string> table_namelist; //
int space_operator; //
int method; //
int estimated_total_df; //
long unsigned int Service_Time; //
```

**EXAMPLE**

```

Cparameter_t p_in, p_out;           //
p_in.set_id = set_id;

int ret_val = clientLib.Request(GET_SET_INFO, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

//
cout << "=====" << endl;
cout << "\t[      ] : " << UECONVCODE(p_out.query) << endl;
cout << "\t[      ] : " << UECONVCODE(p_out.extended_query) << endl;
cout << "\t[      ] : " << endl;
for (int i = 0; i < p_out.stopwords.size(); i++) {
    cout << "\t\t[" << UECONVCODE(p_out.stopwords[i].section_name) << "] : ";
    cout << UECONVCODE(p_out.stopwords[i].section_value) << endl;
}

cout << "\t[Term      ] : " << endl;
for (int i = 0; i < p_out.terms.size(); i++) {
    cout << "\t\t[" << UECONVCODE(p_out.terms[i].section_name) << "] : ";
    cout << UECONVCODE(p_out.terms[i].section_value) << endl;
}

cout << "\t[Table      ] : " << endl;
for (int i = 0; i < p_out.table_namelist.size(); i++) {
    cout << "\t\t[" << UECONVCODE(p_out.table_namelist[i]) << endl;

```

---

```
    }  
    cout << "\t[      ] : " << UECONVCODE(p_out.method) << endl;  
    cout << "\t[      ] : " << UECONVCODE(p_out.space_operator) <<  
    endl;  
    cout << "\t[      ] : " << UECONVCODE(p_out.estimated_total_df) <<  
    endl;  
  
    cout << "=====" << endl;
```

---

**SERVICE NAME**

RETRIEVE

**DESCRIPTION**

.  
 ,  
 ,  
 , ( )  
 2.1 , UTF-8  
 , (order : 가  
 , order=DESCENDING)  
 . ASCENDING .  
 , ( .)  
 .  
 Thesaurus\_levels  
 .  
 , 가  
 가 . integrate  
 .  
 is\_used true .  
 가 , . Integrate

---

order . , 가  
 , 가 .  
 integrate  
 가 가  
 .  
 :  
 A (3 )  
 - -  
 - -  
 - -  
 B (2 )  
 - -  
 - -  
 C (1 )  
 - -

### IN PARAMETER

```
vector<string> table_namelist;
int space_operator;
int method;
string query;
int term_expansion;
int remove_chars_word;
bool order;
vector<int> thesaurus_levels;
Cintegrate_retrieve_t integrate;
```

---

**OUT PARAMETER**

```

unsigned int set_id;
unsigned int total_df;
unsigned int estimated_total_df;           //
vector<Cdocument_group> document_groups;

long unsigned int Service_Time;

```

**EXAMPLE**

```

Cparameter_t p_in, p_out;                 //

//=====
//
//=====
p_in.table_namelist = table_namelist;     //
p_in.space_operator = space_op; //          (ObjectClasses.h      )
p_in.term_expansion = FULL_QUERY_TERM_EXPANSION; //

p_in.remove_chars_word = 1; // 1
p_in.method= ret_method; //                (ObjectClasses.h      )
p_in.order = ASCENDING; //                 (ObjectClasses.h      ),
p_in.order=DESCENDING

```

---

```
string u_query="TIK: "; // UTF-8
EUCKR_TO_UTF8(query, u_query);
p_in.query = query; //

//=====
//
//=====

int ret_val = clientLib.Request(RETRIEVE, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "===== " << endl;
cout << " : " << p_out.total_df << endl;
cout << " : " << p_out.set_id << endl;
cout << "===== " << endl;
```

---

**SERVICE NAME**

RETRIEVE\_SIMILAR\_DOCUMENTS

**DESCRIPTION**

```

vector<string> table_namelist;
vector<Cdocument_t> documents;
float similarity_value;
unsigned int set_id;

```

가 . , ,  
, . Cdocument\_t  
documents  
. Cdocument\_t weight sections  
가 . 가 가  
가 .  
가 Set\_id 가  
. .  
set\_id 0 가  
, 가 .

**IN PARAMETER**

```

vector<string> table_namelist;
vector<Cdocument_t> documents;
float similarity_value;
unsigned int set_id;

```

**OUT PARAMETER**



```

unsigned int set_id;
unsigned int total_df;
long unsigned int Service_Time;

```

**EXAMPLE**

```

p_in.table_namelist = table_namelist;           //
vector<Csection_t> sections;                    //
Csection_t oneSection;
oneSection.section_name="TIK"; //
oneSection.section_value=""; //
sections.push_back(oneSection);
p_in.documents.sections=sections;
p_in.similarity_value = threshold;            //
p_in.set_id = p_out.set_id;                  //

p_in.set_id=0

//
ret_val = clientLib.Request(RETRIEVE_SIMILAR_DOCUMENTS, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "=====" << endl;

```

```
cout << "                : " << p_out.total_df << endl;
cout << "                : " << p_out.set_id << endl;
cout << "===== " << endl;
```

---

**SERVICE NAME**

RETRIEVE\_IN\_RESULT

**DESCRIPTION**

Thesaurus\_levels

ID

가

**IN PARAMETER**

```
unsigned int set_id;
int term_expansion;
int remove_chars_word;
string query;
vector<int> thesaurus_levels;
```

**OUT PARAMETER**

```
unsigned int set_id;
unsigned int total_df;
unsigned int estimated_total_df; //
long unsigned int Service_Time;
```

---

**EXAMPLE**

```
Cparameter_t p_in, p_out;           //
p_in.set_id = set_id;              //
string u_query="Tik:  ";           //      UTF-8
EUCKR_TO_UTF8(query, u_query);
p_in.query = query;                //

//

int ret_val = clientLib.Request(RETRIEVE_IN_RESULT, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "=====" << endl;
cout << "                : " << p_out.total_df << endl;
cout << "                : " << p_out.set_id << endl;
    cout << "=====" << endl;
```

---

**SERVICE NAME**

GET\_DOCUMENTS\_FROM\_RESULT

**DESCRIPTION**

가 , 가  
 , 가  
 가 . ALL\_SECTIONS  
 가 .  
 Documents primary\_key 가 primary key section  
 KRISTAL ,  
 GET\_DOCUMENTS\_BY\_PRIMARY\_KEY primary\_key  
 .  
 가  
 displays . Displays ,  
 .  
 가 . LISTING  
 Csection\_t offset\_list length\_list  
 , INSERTING\_TAG\_\* 가  
 start\_tag end\_tag tag  
 tag .

**IN PARAMETER**

```
Vector<Cdisplay_t> displays;
```

```
unsigned int set_id;
unsigned int start_position;
unsigned int counter;
vector<string> section_namelist;
```

### OUT PARAMETER

```
vector<Cdocument_t> documents;

long unsigned int Service_Time;
```

### EXAMPLE

```
p_in.set_id = set_id; //
p_in.start_position = start_pos; // 가
p_in.counter = 10; // 가
p_in.section_namelist = section_namelist; //

Cdisplay_t adisplay;
adisplay.section_name=section_namelist[0]; //
adisplay.highlight_method = INSERTING_TAG_ALL; //
// INSERTING_TAG

adisplay.start_tag="<font color='red'>"; //
adisplay.end_tag="</font>"; //
adisplay.summary_method=NORMAL_SUMMARY; //
```

---

---

```
adisplay.summary_length=500;                                     //

p_in.displays.push_back(adisplay);

//                가
ret_val = clientLib.Request(GET_DOCUMENTS_FROM_RESULT, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "=====" << endl;

//
for (int i = 0; i < p_out.documents.size(); i++) {
    cout << "<DocID:";
    cout << p_out.documents[i].document_id;
    cout << "> <TableID:";
    cout << p_out.documents[i].table_id;
    cout << ">" << endl;
    for (int j = 0; j < p_out.documents[i].sections.size(); j++) {
        cout << "\t";
        cout << "[";
        // EUC-KR
        cout << UECONVCODE(p_out.documents[i].sections[j].section_name);
        cout << "]:";
    }
}
```

---

```
        // EUC-KR
        cout << UECONVCODE(p_out.documents[i].sections[j].section_value);
        cout << endl;
    }
}
cout << "======" << endl << endl;
```

---



**SERVICE NAME**

GET\_DOCUMENTS\_WITH\_IDS

**DESCRIPTION**

가 . GET\_DOCUMENTS\_FROM\_RESULT  
 . GET\_DOCUMENTS\_FROM\_RESULT  
 가 ,  
 GET\_DOCUMENTS\_WITH\_IDS ID ID  
 가 .  
 가 . 가  
 가 .  
 .  
 (GET\_DOCUMENTS\_FROM\_RESULT GET\_DOCUMENTS\_WITH\_IDS  
 .)  
 documents weight sections . Documents  
 primary\_key 가 primary key section  
 kristal ,  
 GET\_DOCUMENTS\_WITH\_PRIMARY\_KEY primary\_key  
 .  
 GET\_DOCUMENT\_WITH\_ID  
 set\_id 가  
 . GET\_DOCUMENTS\_FROM\_RESULT  
 .

**IN PARAMETER**

```

Vector<Cdisplay_t> displays;
unsigned int set_id;
vector<Cdocument_t> documents;
vector<string> section_namelist;

```

**OUT PARAMETER**

```

vector<Cdocument_t> documents;

long unsigned int Service_Time;

```

**EXAMPLE**

```

Cparameter_t p_in, p_out;          //
//////////////////////////////////////
// 1.                               가
//////////////////////////////////////
Cdocument_t doc;
doc.document_id = doc_id;          //
doc.table_id = table_id;           //
p_in.documents.push_back(doc);
// Display
//   가
int ret_val = clientLib.Request(GET_DOCUMENTS_WITH_IDS, p_in, p_out);

```

---

```
if (ret_val != 0) {
    cout << "ERROR :" << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

//
for (int i = 0; i < p_out.documents.size(); i++) {
    cout << "<DocID:";
    cout << p_out.documents[i].document_id;
    cout << "> <TableID:";
    cout << p_out.documents[i].table_id;
    cout << ">" << endl;
    for (int j = 0; j < p_out.documents[i].sections.size(); j++) {
        cout << "\t";
        cout << "[";
        // EUC-KR
        cout << UECONVCODE(p_out.documents[i].sections[j].section_name);
        cout << "]:";
        // EUC-KR
        cout << UECONVCODE(p_out.documents[i].sections[j].section_value);
        cout << endl;
    }
}

cout << "===== " << endl << endl;
```

---

**SERVICE NAME**

GET\_DOCUMENTS\_WITH\_PRIMARY\_KEY

**DESCRIPTION**

GET\_DOCUMENTS\_XXX

primary\_key .

, , ,

direction primary\_key\_counter . direction ,

primary\_key . Direction

2.1 .

가 . Table name

table 가 .

**IN PARAMETER**

```
int direction; // previous, first
string primary_key; // request_primary_key
unsigned int counter; //request_primary
vector<string> table_namelist;
vector<string> section_namelist;
```

**OUT PARAMETER**

```
vector<Cdocument_t> documents;
```

```
long unsigned int Service_Time;
```

### EXAMPLE

```
Cparameter_t p_in, p_out;           //
p_in.direction = dir;               //
p_in.primary_key = p_key;           //
p_in.counter = (unsigned int) cnt;  //
p_in.table_namelist = table_namelist; // 가
p_in.section_namelist = section_namelist; //

int ret_val = clientLib.Request(GET_DOCUMENTS_WITH_PRIMARY_KEY,
p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

for (int i = 0; i < p_out.documents.size(); i++) {
    cout << "[DOCID:" << p_out.documents[i].document_id << "] ";
    cout << "[TABLEID:" << p_out.documents[i].table_id << "]" << endl;
    for (int j = 0; j < p_out.documents[i].sections.size(); j++) {
        string secname = p_out.documents[i].sections[j].section_name;
        string secval = p_out.documents[i].sections[j].section_value;

        cout << "[#" << UECONVCODE(secname) << "]";
```

---

```
    cout << "[" << secval.size() << "];  
    cout << " = ";  
    cout << "[" << UECONVCODE(secval) << "];  
    cout << endl;  
    }  
    cout << endl;  
    }
```

---



```
Cdocument_t doc;
p_in.documents.push_back(doc);
for (int i = 0; i < section_namelist.size(); i++) {
    Csection_t section;    //
    section.section_name = section_namelist[i];
    section.section_value = EUCONVCODE("      +
section_namelist[i] + ")    .");

    p_in.documents[0].sections.push_back(section); //      가
}

//
int ret_val = clientLib.Request(GET_DOCUMENT_ID_WITH_DOCUMENT, p_in,
p_out);

if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << " DOC ID : " << p_out.documents[0].document_id << endl;
```

---



**SERVICE NAME**

BROWSE\_ALL\_DOCUMENTS

**DESCRIPTION**

Table Set ID .

**IN PARAMETER**

vector&lt;string&gt; table\_namelist ;

**OUT PARAMETER**

```
unsigned int set_id;
unsigned int total_df;
long unsigned int Service_Time;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out; //
String table= "BLUE01" ;
p_in.table_namelist(table) ;

//
int ret_val = clientLib.Request(BROWSE_ALL_DOCUMENTS, p_in, p_out);

if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
```

---

```
        return -1;
    }
    cout << " Set ID : " << p_out.set_id << endl;
    cout << " Total DF : " << p_out.total_df << endl;
```

---

**SERVICE NAME**

CALCULATE

**DESCRIPTION**

MAX,MIN,AVG,SUM,CNT . Query  
가 .  
\* ,

**IN PARAMETER**

```
String query ; //
Vector<string> table_namelist ; //
Vector<string> section_namelist ;
```

**OUT PARAMETER**

```
Vector<string> return_values;
long unsigned int Service_Time;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out; //
String table= "BLUE01" ;
String section= "ABS"
```

```
p_in.table_namelist(table) ;
p_in.section_namelist(section) ;
string query= "MIN"

//
int ret_val = clientLib.Request(CALCULATE, p_in, p_out);

if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}
cout << "[      ] " << endl;
int k=0;
for (int i = 0; i < p_in.table_namelist.size(); i++) {
    cout << "\t[Table] : " << p_in.table_namelist[i] << endl;
    for (int j = 0; j < p_in.section_namelist.size(); j++) {
        cout << "\t\t[section] : " << p_in.section_namelist[i] << "-";
        cout << p_out.return_values[k] << endl;
        k++;
    }
}
}
```

---



```
Cparameter_t p_in, p_out;           //
p_in.set_id = set_id;               //
p_in.section_namelist = section_namelist2; //
p_in.order = order;                 //
p_in.sorting_key_type = sorting_key_type; //

int ret_val = clientLib.Request(SORT_BY_SECTION, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "=====" << endl;
cout << "                : " << p_out.total_df << endl;
cout << "                : " << p_out.set_id << endl;
cout << "=====" << endl;
```

---

**SERVICE NAME**

PROCESS\_DATABASE\_SCHEMA

**DESCRIPTION**

가 XML

**IN PARAMETER**

String schema\_text; //

**OUT PARAMETER**

long unsigned int Service\_Time;

**EXAMPLE**

```
//=====
//                                     blue_0001
//
//=====
Cparameter_t p_in, p_out;          //

//=====
//
//=====
p_in.schema_text = "<?xml version=\"1.0\"?>\n";
p_in.schema_text += "<DatabaseSchema>\n";
p_in.schema_text += "    <UseDatabase    database-name=\"BLUE-DB\"
```

---

```

volume-dir="TEMP"/>\n";
p_in.schema_text += "<CreateTable table-name=\"blue_0001\" with-schema=\"sch
ema01\"/>\n";
p_in.schema_text += "</DatabaseSchema>\n";

//=====
//
//=====

int ret_val = clientLib.Request(PROCESS_DATABASE_SCHEMA, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "SUCESS" << endl;

```

#### ADDITIONAL DESCRIPTIONS

DeleteDatabase)	“volume-dir”	(CreateDatabase, UseDatabase, NULL
-----------------	--------------	---------------------------------------



**SERVICE NAME**

APPEND\_DOCUMENT

**DESCRIPTION**

. Documents sections,  
table\_id .  
documents  
document\_id, weight . table\_id document\_id 가

**IN PARAMETER**

```
vector<Cdocument_t> documents;
```

**OUT PARAMETER**

```
vector<Cdocument_t> documents;  
long unsigned int Service_Time;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out; //  

/////////////////////////////////////  

//  

/////////////////////////////////////  

Cdocument_t doc;  

doc.table_id = table_id;
```

```
p_in.documents.push_back(doc); //
for (int i = 0; i < section_namelist.size(); i++) {
    Csection_t section; //
    section.section_name = section_namelist[i];
    section.section_value = EUCONVCODE("      +
section_namelist[i] + ")      .");

    p_in.documents[0].sections.push_back(section); //      가
}

//
int ret_val = clientLib.Request(APPEND_DOCUMENT, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}
cout << "[Append Id] : " << p_out.documents[0].document_id << endl;
```

---

**SERVICE NAME**

DELETE\_DOCUMENT

**DESCRIPTION**

```

                . Documents
                . Documents weight
sections      .

```

**IN PARAMETER**

```
vector<Cdocument_t> documents;
```

**OUT PARAMETER**

```
long unsigned int Service_Time;
```

**EXAMPLE**

```

Cparameter_t p_in, p_out;           //

////////////////////////////////////
//
////////////////////////////////////

Cdocument_t doc;
doc.table_id = table_id;           //
doc.document_id = docID;           //
p_in.documents.push_back(doc);

```

```
//  
int ret_val = clientLib.Request(DELETE_DOCUMENT, p_in, p_out);  
if (ret_val != 0) {  
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;  
    return -1;  
}
```

---

**SERVICE NAME**

UPDATE\_DOCUMENT

**DESCRIPTION**

. Documents sections

. documents weight .

**IN PARAMETER**

vector&lt;Cdocument\_t&gt; documents;

**OUT PARAMETER**

long unsigned int Service\_Time;

**EXAMPLE**

```
Cparameter_t p_in, p_out;           //
////////////////////////////////////
//
////////////////////////////////////
Cdocument_t doc;
doc.table_id = table_id;           //
doc.document_id = docID;          //
p_in.documents.push_back(doc);
```

---

```
for (int i = 0; i < section_namelist.size(); i++) {
    Csection_t section;    //
    section.section_name = section_namelist[i];
    section.section_value = EUCONVCODE("      +
section_namelist[i] + ")    .");

    p_in.documents[0].sections.push_back(section); //      가
}

//
int ret_val = clientLib.Request(UPDATE_DOCUMENT, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}
```

---

**SERVICE NAME**

CHECK\_STATUS

**DESCRIPTION**

```

        check      .      . Flag      IP
                가      . True      가
        False      . Server_type      Kristal
                . Server_version      ,
        DB_description      .

```

**IN PARAMETER****OUT PARAMETER**

```

    Bool flag;
    int server_type;
    String server_version;
    String db_description;
    long unsigned int Service_Time;

```

**EXAMPLE**

```

Cparameter_t p_in, p_out;      //

int ret_val = clientLib.Request(CHECK_STATUS, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;

```

```
        return -1;
    }
    cout << "[Authorization] : " << p_out.flag << endl;
    cout << "[Server_Type] : " << p_out.server_type << endl;
    cout << "[Version] : " << p_out.server_version << endl;
    cout << "[Description] : " << p_out.db_description << endl;
```

---



**SERVICE NAME**

SAVE\_USER\_LOG

**DESCRIPTION**

가 .  
u\_kristald.log . u\_kristald.log  
100MBytes u\_kristald.log.01 ,  
u\_kristald.log . u\_kristald.log.01 u\_kristald.log.10 10  
가 , 1GBytes .  
가 ,  
.

**IN PARAMETER**

String userarea\_text;

**OUT PARAMETER****EXAMPLE**

```
Cparameter_t p_in, p_out; //
p_in.userarea_text = “ ”;
int ret_val = clientLib.Request(SAVE_USER_LOG, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
```

```
    return -1;  
}
```

---

**SERVICE NAME**

GET\_XML\_NODES\_FROM\_RESULT

**DESCRIPTION**

XML Node 가 .

XML ,

. Flag XML Node

가 가

. Default false .

XML

GET\_XML\_NODES\_FROM\_RESULT

, Path .

XML .

search\_flag true false

path .

가

displays . Displays ,

. 가 . LISTING

Csection\_t offset\_list length\_list

, INSERTING\_TAG\_\* 가

start\_tag end\_tag tag

tag .

**IN PARAMETER**

```
Vector<Cdisplay_t> displays;
unsigned int set_id;
unsigned int start_position;
unsigned int counter;
bool flag;
vector<string> section_namelist;
```

**OUT PARAMETER**

```
vector<CXML_node_t> nodes;
long unsigned int Service_Time;
```

**EXAMPLE**

```
p_in.set_id = set_id; //
p_in.start_position = start_pos; // 가
p_in.counter = 10; // 가
p_in.flag = false; // 가 가

p_in.section_namelist = section_namelist; //

Cdisplay_t adisplay;
adisplay.section_name=section_namelist[0]; //
adisplay.highlight_method= INSERTING_TAG_ALL; //
```

---

---

```
//          INSERTING_TAG

adisplay.start_tag="<font color='red'>";          //
adisplay.end_tag="</font>";          //
p_in.displays.push_back(adisplay);

//          가
ret_val = clientLib.Request(GET_XML_NODES_FROM_RESULT, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "===== " << endl;

//
for (int i = 0; i < p_out.nodes.size(); i++) {
    cout << "<search_flag:";
    cout << p_out.nodes[i].search_flag; //          node
        node    path
    cout << "> <TableID:";
    cout << p_out.nodes[i].table_id;
    cout << "<NodeID:";
    cout << p_out.nodes[i].system_section.node_id;
    cout << p_out.nodes[i].system_section.title;
    cout << p_out.nodes[i].system_section.level; //          가
```

---

```
cout << ">" << endl;
for (int j = 0; j < p_out.nodes[i].sections.size(); j++) {
    cout << "\t";
    cout << "[";
    // EUC-KR
    cout << UECONVCODE(p_out.nodes[i].sections[j].section_name);
    cout << ":";
    // EUC-KR
    cout << UECONVCODE(p_out.nodes[i].sections[j].section_value);
    cout << endl;
}
}
cout << "===== " << endl << endl;
```

---

**SERVICE NAME**

GET\_XML\_NODES\_WITH\_IDS

**DESCRIPTION**

Node 가 . GET\_XML\_NODES\_FROM\_RESULT  
 . GET\_XML\_NODES\_FROM\_RESULT  
 가 ,

GET\_XML\_NODES\_WITH\_IDS ID ID  
 가 .  
 가 . ( 가  
 가 .)

Flag XML Node 가  
 가 . Default false .

GET\_DOCUMENT\_WITH\_ID  
 set\_id 가  
 . GET\_DOCUMENTS\_FROM\_RESULT  
 .

**IN PARAMETER**

```

Vector<Cdisplay_t> displays;
Unsigned int set_id;
vector<CXML_node_t> nodes;
vector<string> section_namelist;
bool flag;

```

**OUT PARAMETER**

```
vector<CXML_node_t> nodes;
long unsigned int Service_Time;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out;          //
////////////////////////////////////
// 1.                               가
////////////////////////////////////
CXML_node_t node;
node.system_section.node_id = node_id;          //
node.table_id = table_id;                       //
p_in.nodes.push_back(node);
p_in.flag = false;                              //          가          가

p_in.section_namelist = section_namelist;      //

//          가
ret_val = clientLib.Request(GET_XML_NODES_WITH_IDS, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}
```



```
}

cout << "===== " << endl;

//
for (int i = 0; i < p_out.nodes.size(); i++) {
    cout << "<search_flag:";
    cout << p_out.nodes[i].search_flag; // node
        node    path
    cout << "> <TableID:";
    cout << p_out.nodes[i].table_id;
    cout << "<NodeID:";
    cout << p_out.nodes[i].system_section.node_id;
    cout << p_out.nodes[i].system_section.title;
    cout << p_out.nodes[i].system_section.level; // 가

    cout << ">" << endl;
    for (int j = 0; j < p_out.nodes[i].sections.size(); j++) {
        cout << "\t";
        cout << "[";
        // EUC-KR
        cout << UECONVCODE(p_out.nodes[i].sections[j].section_name);
        cout << "]:";
        // EUC-KR
        cout << UECONVCODE(p_out.nodes[i].sections[j].section_value);
        cout << endl;
    }
}
```

---



**SERVICE NAME**

GET\_XML\_NODES\_INFO

**DESCRIPTION**

XML  
 가 . XML\_ROOT\_NODE , XML\_PARENT\_NODE,  
 XML\_LEFT\_NODE , XML\_RIGHT\_NODE, XML\_CHILD\_NODE .  
 가 XML\_LEFT\_NODE, XML\_RIGHT\_NODE  
 가 1 가 .  
 table\_id node\_id  
 . ( 가 GET\_DOCUMENTS\_WITH\_PRIMARYKEY  
 .)

**IN PARAMETER**

```
CXML_node_t pivot_node;           //
int direction;                     //
int counter;                       //
```

**OUT PARAMETER**

```
vector<CXML_node_t> nodes;
long unsigned int Service_Time;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out;         //
p_in.pivot_node.table_id=table_id ; //
```

```
p_in.pivot_node.system_section.node_id=pivot_node_id ;
p_in.direction = XML_LEFT_NODE ;           //          가

p_in.counter = 10 ;
//=====
//
//=====
int ret_val = clientLib.Request(GET_XML_NODES_INFO, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}
```

---

**SERVICE NAME**

GET\_XML\_TREE

**DESCRIPTION**

DFS . XML Tree . node

**IN PARAMETER**

CXML\_node\_t pivot\_node; //

**OUT PARAMETER**

long unsigned int Service\_Time;  
vector<CXML\_node\_t> nodes;

**EXAMPLE**

```
Cparameter_t p_in, p_out; //

p_in.pivot_node.table_id=table_id ; //
p_in.pivot_node.system_section.node_id=pivot_node_id ;

//=====
//
//=====

int ret_val = clientLib.Request(GET_XML_TREE, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
```

---

```
    return -1;  
}
```

---

**SERVICE NAME**

REPRODUCE\_XML\_NODE

**DESCRIPTION**

XML

**IN PARAMETER**

```
vector<CXML_node_t> nodes;  
string element_name;  
string attribute_name;
```

**OUT PARAMETER**

```
vector<CXML_node_t> nodes;  
long unsigned int Service_Time;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out;      //  
CXML_node_t node;  
node.system_section.node_id = node_id;  //
```

---

```
node.table_id = table_id;           //
p_in.nodes.push_back(node);
p_in.element_name = "    ";       //
p_in.attribute_name = "    ";     //

//          가
ret_val = clientLib.Request(REPRODUCE_XML_NODE, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

//
string xmlFile = p_out.nodes[0].sections[0].section_name;
xmlDoc = "<?xml version='1.0' encoding='UTF-8'?>\n" + xmlDoc;
// xml
cout << xmlDoc << endl;
```

---



**SERVICE NAME**

RETRIEVE\_DOCUMENTS\_WITH\_FOREIGN\_KEY

**DESCRIPTION**

documents document\_id table\_id  
가 ,  
section\_namelist 가  
Target\_section , Order 가  
Navigation .

**IN PARAMETER**

```
vector<Cdocument_t> documents;
vector<string> section_namelist;
vector<string> table_namelist;
string target_section;
bool order;
```

**OUT PARAMETER**

```
unsigned int set_id;
unsigned int total_df;
long unsigned int Service_Time;
```

**EXAMPLE**

```

Cparameter_t p_in, p_out;           //
CDocument_t aDoc;
aDoc.table_id=table_id;
aDoc.document_id=document_id;
p_in.documents.push_back(aDoc);

string asection="TIK";
p_in.section_namelist.push_back(asection);
p_in.table_namelist.push_back("BLUE01");

p_in.target_section="ABS";
p_in.order=ASCENDING;

//
int          ret_val          =
clientLib.Request(RETRIEVE_DOCUMENTS_WITH_FOREIGN_KEY, p_in,
p_out);

if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "=====" << endl;
cout << "          : " << p_out.total_df << endl;
cout << "          : " << p_out.set_id << endl;
    cout << "=====" << endl;

```

**SERVICE NAME**

APPEND\_XML\_NODE

**DESCRIPTION**

XML  
 , , XML\_LEFT\_NODE,  
 XML\_RIGHT\_NODE, XML\_CHILD\_NODE

**IN PARAMETER**

```
vector<CXML_node_t> nodes;           //
CXML_node_t pivot_node;           //
int direction;                     //
```

**OUT PARAMETER**

```
long unsigned int Service_Time;
vector<CXML_node_t> nodes;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out;           //

//=====
//
//=====
CXML_node_t_t node;
node.table_id = table_id;           //
p_in.nodes.push_back(node);
```

```

//=====
//
//=====
for (int i = 0; i < section_namelist.size(); i++) {
    Csection_t section;          //
    section.section_name = section_namelist[i];
    section.section_value = EUCONVCODE("      (" +
section_namelist[i] + ")      .");

    p_in.nodes[0].sections.push_back(section); //      가
}

p_in.pivot_node.table_id=table_id ;          //
p_in.pivot_node.system_section.node_id=pivot_node_id ;
p_in.direction = XML_CHILD_NODE ;           //

//=====
//
//=====
int ret_val = clientLib.Request(APPEND_XML_NODE, p_in, p_out);
if (ret_val != 0)
{
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

```

**SERVICE NAME**

UPDATE\_XML\_NODE

**DESCRIPTION**

XML . UPDATE\_DOCUMENT .

**IN PARAMETER**

vector&lt;CXML\_node\_t&gt; nodes; //

**OUT PARAMETER**

long unsigned int Service\_Time;

**EXAMPLE**

```
Cparameter_t p_in, p_out; //

//=====
//
//=====
CXML_node_t node;
node.table_id = table_id; //
node.system_section.node_id = node_Id ; //
p_in.nodes.push_back(node);

//=====
//
//=====
for (int i = 0; i < section_namelist.size(); i++)
{
```

---

```
    Csection_t section;           //
    section.section_name = section_namelist[i];
    section.section_value = EUCONVCODE("      +
section_namelist[i] + ")      .");

    p_in.nodes[0].sections.push_back(section); //      가
}

//=====
//
//=====
int ret_val = clientLib.Request(UPDATE_XML_NODE, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}
```

---

**SERVICE NAME**

DELETE\_XML\_NODE

**DESCRIPTION**

XML . 가 가  
, 가  
.

**IN PARAMETER**

vector&lt;CXML\_node\_t&gt; nodes; //

**OUT PARAMETER**unsigned int total\_df;  
long unsigned int Service\_Time;**EXAMPLE**

```
Cparameter_t p_in, p_out; //  
  
//=====  
//  
//=====  
CXML_node_t node;  
node.table_id = table_id; //  
node.system_section.node_id = node_id; //  
p_in.nodes.push_back(node);  
  
//=====
```

---

```
//  
//=====
```

```
int ret_val = clientLib.Request(DELETE_XML_NODE, p_in, p_out);  
if (ret_val != 0) {  
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;  
    return -1;  
}  
cout << "                = " << p_out.total_df << endl;
```

---



**SERVICE NAME**

MOVE\_XML\_NODE

**DESCRIPTION**

```
XML ,
XML_LEFT_NODE, XML_RIGHT_NODE, XML_CHILD_NODE
node_id
```

**IN PARAMETER**

```
vector<CXML_node_t> nodes; //
CXML_node_t pivot_node; //
int direction; //
```

**OUT PARAMETER**

```
long unsigned int Service_Time;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out; //

CXML_node_t_t node;
node.table_id = table_id;
node.system_section.node_id =node_id ;
p_in.nodes.push_back(node);

p_in.pivot_node.table_id=table_id ; //
p_in.pivot_node.system_section.node_id=pivot_node_id ;
p_in.direction = XML_CHILD_NODE ; //
```

---

```
//=====
//
//=====
int ret_val = clientLib.Request(MOVE_XML_NODE, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}
```

---



**OUT PARAMETER**

```
unsigned int total_df;
unsigned int estimated_total_df;           //
vector<Cdocument_group> document_groups;

long unsigned int Service_Time;
```

**EXAMPLE**

```
Cparameter_t p_in, p_out;                //

//=====
//
//=====

p_in.table_namelist = table_namelist;    //
// table_namelist      table      2 , 1 , 3

p_in.group_counts.push_back(2);
p_in.group_counts.push_back(1);
p_in.group_counts.push_back(3);

p_in.space_operator = space_op; //      (ObjectClasses.h      )
p_in.term_expansion = FULL_QUERY_TERM_EXPANSION; //
```

---

---

```

p_in.remove_chars_word = 1; // 1
p_in.method= ret_method;      //          (ObjectClasses.h      )
p_in.order = ASCENDING;      //          (ObjectClasses.h      )

string u_query="TIK:      ";    //          UTF-8
EUCKR_TO_UTF8(query, u_query);
p_in.query = query;          //

//=====
//
//=====

int ret_val = clientLib.Request(MULTIPLE_RETRIEVE, p_in, p_out);
if (ret_val != 0) {
    cout << "ERROR : " << p_out.errmsg << "(" << p_out.errcode << ")" << endl;
    return -1;
}

cout << "===== " << endl;
cout << "          : " << p_out.total_df << endl;
cout << "===== " << endl;
//          set id
for (int i=0; i< document_groups.size(); i++) {
    cout << "Group [ " << i << "]" : " << document_groups[i].set_id << " - " <<
document_groups[i].df_per_group; << endl;
}

```

---

