MARVINBOT

mARC API REFERENCE

Core 1.0 codename Syncytio

NRMX

28/08/2014

Revision 1.5

consistent with core `Build = "1.2014.08.28.18.35" and higher`

Object	Methods	properties
server	6	25
session	24	18
table	22	2
contexts	16	6
results	17	7
total	**83**	**58**

Doc Version 1.0

history

Revision 1.5 :

ContextToContext	examples of use	
error messages	added	
server.GetTasks	modified	
contexts.onTop	example correction	
session.index	implemented	
session.predict	renamed session.completion	
session.Predict	depleted	
session.ApplySpectrum	added	
session.GetInstances	modified	
example corrections	Bug18, bug 38, bug 34	
page API reference	modified	
Contexts.Normalize	enhanced	compatible
Contexts.fetch	modified	compatible
contexts.SetProperties	enhancement	compatible
Contexts.Getproperties	modified	compatible
contexts.ApplySpectrum	relocated in	session.ApplySpectrum
contexts.ApplySpectrum	depleted	
Results.fetch	modified	compatible
Results.Getproperties	modified	compatible
results.SetProperties	enhancement	compatible
Bug 13	solved	

Revision 1.4 :

Session.GetSpectrum	added
Session.SetSpectrum	added

Revision 1.3 :

more examples

Results.Add	depleted
Results.RollDown	depleted
Results.RollUp	depleted
Results.OnTop	added

Revision 1.2 :

Results.GetFormat	depleted
Results.SetFormat	depleted
Results.SetAdd	depleted
Results.RollDown	depleted
Results.RollUp	depleted
contexts.SortByGenerality	depleted
contexts.SortByActivity	depleted
Contexts.RollDown	depleted
Contexts.RollUp	depleted

Contexts.GetElements	depleted
Contexts.OnTop	added
Contexts.SortBy	added
Contexts.Normalize	added
Contexts.Fetch	added

Revision 1.1 :

Contexts.Clear	depleted
Results.Clear	depleted
Session.Clear	modified
General Overview	modified

Revision 1.0	initial version

Table of content

General overview

A mARC application is build by the following Objects (Classes)

SERVER
SESSION
TABLE (alias TBL)

inside a session,

class names, object names or method names are **not** case sensitive
an object name **cannot** be KNW_ABSTRACT, KNW_MEANING, KNW_LANGAGE, NONE ,ROWID , NULL, ACT, ORIGIN,DEFAULT

an object has always an owner, explicitely, or implicetely the server

an object name can only be composed by alphanumeric chars, and the char « _ » (underscore)

Example valid:	623table, auteur_1
Example not valid :	table#3, nom de l'auteur, etc…

Referencing an object :
classname:object_name
TABLE:wikimaster2 (the names are not case sensitive)

Referencing a global class method
classname.method_name (params)

Referencing an object's method :
classname:object_name.method (params)

several function calls can be issued inside a same command, each of them must be delimited by the character " **;** "
Example :

```
SessionId  TBL:wikimaster2.Update ( <Id>  <colname1>  <val1>) ;
SessionId  TABLE:wikimaster2.Insert  (
                                      <colname_1>  <val_1>,
                                      ….
                                      <colname_n> <val_n>,
                              ) ;
```

in such a script, a comment can be written between the chars « / » et « ; ».
Example :
/ this is a comment **;**

Architecture

A Marvin mARC application is build with
one mARC objects (organic like neural nets)
an optional set of TABLE objects (database table)

client/server data exchanges use SESSION objects

each SESSION object owns

- one CONTEXTS objects to handle the semantic contexts
- one RESULTS objects to handle table data , and document search
- one INHIBITOR context
- one PROFILER context

Intrinsic data types

Type	Taille (octets)	Amplitude	epsilon	description
INT8	1	-127 … 127		entier 8 bit
UINT8	1	0 … 255		entier 8 bit non signé
INT (INT32)	4	-2 147 483 647 … 2 147 483 647		entier 32 bit signé
UINT (UINT32)	4	0 … 4 294 967 294		entier non signé
SIMPLEDATE (SD)	4	An : -16384 … 16383		Date an :mois :jour
CHAR	1-254			chaine 8 bit signé. Type size obligatoire
FLOAT	4	-3.40282347E+38 3.40282347E+38	1.17549434E-38	flottant (precision 7 chiffres)
DOUBLE	8	-1.7976931348623158E+308 1.7976931348623158E+308	2.2250738585072013E-308	flottant (precision 15 chiffres)
INT64	8	-9 223 372 036 854 775 807 9 223 372 036 854 775 807		entier 64 bit signé
UINT64	8	0 … 18 446 744 073 709 551 614		entier 64 bit non signé
BIN	0-2Go			champs binaire. Taille variable
STRING	0-2Go			champs chaine de caractère longue. Taille variable
ROWID	4			Identifiant de ligne d'une table. Equivalent à INT32 dans cette version
CONTEXT	0-2Go			Contexte. Taille variable (usage interne en général)
SESSIONID	21			Identifiant de session. Equivalent à string dans cette version

SIMPLEDATE input may be interpreted according these formats :

DD-MM-YYYY	04-07-2003
DD.MM.YYYY	04.07.2003
YYYY/MM/DD	2003/07/04
MM/DD/YYYY	07/04/2003
Y :M :D	2003:7:4 (internal native format)

Comparison operators

Used by the object RESULTS, generally via it's method SelectBy operating on the Result Set (RS) on top of the RS stack of the RESULT object.

Opérateur	**alias**	**code**	**parametres**	**description**
>	GT	0	1	Greater than op1
<	LT	1	1	Lower than op1
>=	GTE	2	1	Greater or Equal to op1
<=	LTE	3	1	Lower or Equal à op1
Between	BT	4	2	Between [op1, op2] (including op1 and op2)
=	EQ	5	1	Equal to op1
!=	NEQ	6	1	Différent from op1
&	AND	7	1	Logical AND with op1, if result != 0, then true
|	OR	8	1	Logical OR with op1, if result != 0, then true
BeginWith	BW	9	1	String begins with op1
EndWith	EW	10	1	String ends with op1
Contains	CO	11	1	String contains op1

Connecting to a mARC Server

direct IP socket connections

Communication with the Engine is made via TCP/IP sockets: commands and responses are strings of characters with a space as separator. This socket level is encapsulated in a light communication protocol to ensure data integrity.

Communication Level

In order to ensure data integrity at the socket protocol level every request and every response to and from the Engine must begin with a header. This header contains the number of characters of the request/response, under the following format:

#x#yyyyyyyyy

yyyyyyyyy represents the length of the query data in 9 characters max and x represents the number of characters in yyyyyyyyy.

Note: there is **one and only one** space between the header and the remaining data.

Examples:

request **(a dummy request)**

512 void void GetDefaults

This request is 25 characters long, so you have to send the following string:

#2#25 512 void void GetDefaults

return string **(a dummy response)**

512 1 75 60 2 35 2 2 0 1 6 1 20 9 0.4 0 0 0 0 1 0 0

This response is 51 characters long, so the server sends the following string:

#2#51 512 1 75 60 2 35 2 2 0 1 6 1 20 9 0.4 0 0 0 0 1 0 0

you can have a look at the **ToProtocol** and **FromProtocol** methods of the php KMServer class that handle Marvin's protocol string conversion.

Further more the **Send** and **Receive** methods of the php KMServer class shows a classical TCP data exchange, including the management of this tiny protocol.

Application Level

In order to interact with the server, at an application level, you need to get a valid KM session Id.

Just issue the command string to server :

-1 CONNECT (null) ;

the return string will look like this

623 1 1 1 8 0 CHALLENGE <0 \>

623 is the new KM session Id (nb : the full return format is described below)

you can have a look at the **OpenKMSession()** method of the php KMServer class to have an example code for this task.

Every parameter of a method call are transmitted under the shape of a binary string (called GPBinary or GPString), except for symbolic parameters like *null* of *default*

the same convention is used for returned data values

<xxx param\>

example : 123 is written as <3 123\>

example "123" is written as <3 123\>

example : empty string is <0 \>

example : boolean true is <1 1\> or <4 true/>

example : boolean false is <1 0\> or <5 false/>

Boolean data in a returned string are represented by <1 0\> (false) and <1 1\> (true)

the returned string looks always like that :

Id Ret Rows Cols Cols*[Type Size Name] Rows*[Cols*[Value]]

For every native type except the binary type, data are encoded under there ASCII representation.

In a returned string, data types are encoded according to the following convention

type	code
`int, int32`	1
`uint, uint32`	2
`int8`	3
`uint8`	4

char	5
int64	6
uint64	7
string	8
float	9
double	10
bool	11
SimpleDate	12
RowId	13
SessionId	14

Example : one returned string after a function call

"1024 1 2 5 1 0 F_INT 5 20 F_CHAR20 8 0 F_STRING 9 0 F_FLOAT 10 0 F_DOUBLE <2 10/> <10 abcdefghij/> <20 ABCDEFGHIJABCDEFGHIJ/> <11 1.23456E-30/> <20 1.2345678901234E-100/> <3 999/> <11 ewxcvbnjhgf/> <30 ABCyjklDEFGHIJABdfgChhhDEFGHIJ/> <11 1.23456E-30/> <20 1.2345678901234E-100/>"

```
1024                              // session Id
1                                 // error state : ok
2                                 // 2 lines
5                                 // 5 columns

1 0 INT                                                            // Col 1 : Type 1,
Size 0, name = « F_INT »
5 20 CHAR20                                                        // Col 2: Type 5,
Size 20, name = "F_CHAR20"
8 0 STRING                                                         // Col 3: Type 8,
Size 0, name = "F_STRING"
9 0 FLOAT                                                          // Col 4: Type 9,
Size 0, name = "F_FLOAT"
10 0 DOUBLE                                                        // Col 5 : Type 10,
Size 0, name = « F_DOUBLE »

                                                                   // data line 1 :
<2 10/>                                                            // col 1
<10 abcdefghij/>                                                   //
<20 ABCDEFGHIJABCDEFGHIJ/>                                         //
<11 1.23456E-30/>                                                  //
<20 1.2345678901234E-100/>                                         // col 5

                                                                   // data line 2 :
<3 999/>                                                           // col 1
<11 ewxcvbnjhgf/>                                                  //
<30 ABCyjklDEFGHIJABdfgChhhDEFGHIJ/>                               //
<11 1.23456E-30/>                                                  //
<20 1.2345678901234E-100/>                                         // col 5
```

API Reference

According to previous considerations,

In this Document, and in order to avoid complexification and to enhance useability, we shall use the following conventions for coding examples.

something like :

example 1

```
contexts.new ( );
contexts.SetProperties ( " context_string = atome_d_hydrogène") ;
contextToContext ( );
contexts.Fetch ();
contexts.Fetch (5,1);
```

is more readable than the serialized version that have to be transmitted to the server through the mARC protocol

```
contexts.new  (  ) ;
contexts.SetProperties  ( <35  context_string = atome_d_hydrogène/> ) ;
contextToContext  (  ) ;
contexts.Fetch  (  ) ;
contexts.Fetch  ( <1 5/>,<1 1/> ) ;
```

moreover,

something like

```
contexts.Fetch (5,1);
contexts.Fetch ("5","1");
contexts.Fetch  ( <1 5/>,<1 1/> ) ;
```

will be strictly equivalent, but the first one is more easy to interpret

example 2

```
contexts.Clear();
contexts.new ( );
contexts.SetProperties ( " context_string = atome_d_hydrogène") ;
contexts.new ( );
contexts.GetProperties ("context_string",2);
```

will represent the following serialized data sent to the server

```
contexts.Clear (  ) ;
contexts.new  (  ) ;
contexts.SetProperties  ( <35  context_string = atome_d_hydrogène/> ) ;
contexts.new  (  ) ;
contexts.GetProperties  ( <14 context_string/>,<1 2/> ) ;
```

once again, this three representations

```
contexts.GetProperties ("context_string",2);
contexts.GetProperties (context_string,2);
contexts.GetProperties  ( <14 context_string/>,<1 2/> ) ;
```

are equivalent, but the first one is more easy to understand, and the last one is the only one permitted at protocol level.

In this document, the code example will use as often as possible the first representation of the code.

Return values

at protocol level, the returned value of a method is representing a table, instead of a single value like a C routine.

it will look like :

```
GetProperties ();
```

return value

```
3 1 18 4 8 0 prop_name 8 0 prop_value 8 0 prop_type 8 0 prop_access <4 name/> <9 anonymous/> <6 string/> <2 rw/> <9 last_time/> <8
2.5e-002/> <6 double/> <1 r/> <8 owner_IP/> <9 127.0.0.1/> <6 string/> <1 r/> <10 owner_port/> <4 3259/> <6 string/> <1 r/> <2 id/> <1
3/> <6 string/> <1 r/> <8 priority/> <1 3/> <5 uint8/> <2 rw/> <15 session_timeout/> <2 -1/> <5 int32/> <2 rw/> <12 exec_timeout/> <4
5000/> <5 int32/> <2 rw/> <13 context_count/> <2 46/> <5 int32/> <1 r/> <12 result_count/> <1 0/> <5 int32/> <1 r/> <15
spectrum_string/> <271 min_atom = 1; max_atom = -1; min_generality = 0; max_generality = 99; min_activity = 1; max_activity = -1;
max_context = 5; min_context_size = 2; max_context_size = 25; min_context_activity = 25; max_context_activity = -1; max_record = 1000;
depth = 0; evaluate = false; /> <6 string/> <2 rw/> <23 profiler_context_string/> <0 /> <6 string/> <2 rw/> <24 inhibitor_context_string/> <0
/> <6 string/> <2 rw/> <21 result_max_stack_size/> <2 16/> <5 int32/> <2 rw/> <15 result_embedded/> <4 true/> <4 bool/> <2 rw/> <21
result_line_separator/> <4 CRLF/> <6 string/> <2 rw/> <23 result_column_separator/> <5 COMMA/> <6 string/> <2 rw/> <15
result_DBCursor/> <1 0/> <5 int32/> <2 rw/>  ;
```

in this documentation, return values will be represented via a formatted table, instead of the protocol return value. Both of them are strictly equivalent, but the second is a bit more readable ☺.
But keep in mind that what is really travelling between the client and the server, is the previous representation.

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	anonymous	string	rw
last_time	1.187	double	r
owner_IP	127.0.0.1	string	r
owner_port	3256	string	r
id	1	string	r
priority	3	uint8	rw
session_timeout	-1	int32	rw

exec_timeout	5000	int32	rw
context_count	2	int32	r
result_count	0	int32	r
spectrum_string	min_atom = 1; max_atom = -1; min_generality = 0; max_generality = 99; min_activity = 1; max_activity = -1; max_context = 5; min_context_size = 2; max_context_size = 25; min_context_activity = 25; max_context_activity = -1; max_record = 1000; depth = 0; evaluate = false;	string	rw
profiler_context_string		string	rw
inhibitor_context_string		string	rw
result_max_stack_size	16	int32	rw
result_embedded	true	bool	rw
result_line_separator	CRLF	string	rw
result_column_separator	COMMA	string	rw
result_DBCursor	0	int32	rw

Often, for simplicity purpose, the first line, representing the data type of each column will also be omitted, and the representation of the return value will become something like :

prop_name	prop_value	prop_type	prop_access
name	anonymous	string	rw
last_time	1.187	double	r
owner_IP	127.0.0.1	string	r
owner_port	3256	string	r
id	1	string	r
priority	3	uint8	rw
session_timeout	-1	int32	rw
exec_timeout	5000	int32	rw
context_count	2	int32	r
result_count	0	int32	r
spectrum_string	min_atom = 1; max_atom = -1; min_generality = 0; max_generality = 99; min_activity = 1; max_activity = -1; max_context = 5; min_context_size = 2; max_context_size = 25; min_context_activity = 25; max_context_activity = -1; max_record = 1000; depth = 0; evaluate = false;	string	rw
profiler_context_string		string	rw
inhibitor_context_string		string	rw
result_max_stack_size	16	int32	rw
result_embedded	true	bool	rw
result_line_separator	CRLF	string	rw
result_column_separator	COMMA	string	rw
result_DBCursor	0	int32	rw

Case sensitivity

Let's remember, the API is **NOT** case sensitive.
So things like

```
contexts.GetProperties  ( <14 context_string/>,<1 2/> ) ;
Contexts.getProperties  ( <14 context_string/>,<1 2/> ) ;
contexts.getproperties  ( <14 context_string/>,<1 2/> ) ;
CONTEXTS.GETPROPERTIES  ( <14 conText_String/>,<1 2/> ) ;
```

are strictly equivalent, and possible variation of the layout of the code depending of there different contributors must not be interpreted as an error or a bug.

Server

property name		description	defaults & values
name	RW	name of the mARC	mARC
port	R	server's listen port	1254
type	R	core type	syncytiotrophoblaste
model	R	capacity	1Mp 4Mp 16Mp
version	R	server's version	1.0 Beta
build	R	1.2014.07.10.15.05	
connection_count	R	number of connected sockets	0
command_threads	RW	request thread pool size	8
time_local	R		12:43:26:409
time_gmt	R		10:43:26:409
up_time	R	time elapsed since start in ms	65310.6679863281
idle_time	R	net request inactivity time in ms	43700.0474310014
cache_size	RW	DB maximum cache size in KB	32000
cache_used	R	DB used cache size in KB	0
cache_hits	R	DB cache hits in %	0
exec_timeout_default	RW	maximum wait time for request in ms	5000
session_timeout_default	RW	max session inactivity time before killing	-1
marc_relations	R	number of associative relations	0
marc_shapes	R	number of shapes (eg word, aggregates)	0
marc_references	R	number of reference links to documents	0
marc_particles	R	total number of active mARC	0
indexation_cache_used	R	sie in KB of used indexation cache	0
indexation_cache_size	RW	DB used cache size in KB	8000
indexation_timeout	RW	max indexation lock time before abort	-1 (never)
marc_quality	R	0-5 good, 5-10 correct, >10 low	]0,+INF[

Methods	parameters	parameters values
ShutDown	**string** option	"","restart"
GetApi	**void**	
SetProperties	**string** accessor, …**string** accessorN	
GetProperties	**string** accessor, …**string** accessorN	
GetConnected	**int32** start, **int32** count	start = 1 count = -1
GetTasks	**void**	

SERVER.ShutDown

description

shuts down the server, and optionnaly restart it

prototype

```
ShutDown ( string option)
```

if option is "restart", the server will then try to restart

defaults

```
option = ""
```

error messages

```
x :    "unknown error";
```

examples

```
ShutDown ( );
```

the server is down

```
ShutDown ("restart" );
```

the server is down, and will then restart

SERVER.GetApi

description

Lists all API methods of the current server

prototype

```
Server.GetApi (void);
```

defaults

```
void
```

error messages

```
x :    "unknown error";
```

examples

```
Server.GetApi ();
```

Class_id	Method_Id	Method
0	0	SERVER.ShutDown
0	1	SERVER.GetApi
0	2	SERVER.SetProperties
0	3	SERVER.GetProperties
0	4	SERVER.GetConnected
0	5	SERVER.GetTasks
1	0	SESSION.Connect
1	1	SESSION:xxx.Clear
…	…	…

SERVER.SetProperties

description

Access to server's properties

Trying to change the value of a Read Only property will not generate an error.

To see which properties are avalaible, see SERVER.GetProperties

To change one property, use a directive as : propertyname = propertyvalue
To change several properties values in one command, separate each directive with the character semi column (;)

an accessor is a string like :

"propertyname = value"

and can be extended like

"propertyname1 = value1, … propertynameN = valueN"

Depending of your client application using only one extended accessor, as a parameter, is equivalent as using several accessors as parameters

prototype

```
Server.SetProperties ( string accessor, …string accessorN )
```

defaults

void

error messages

```
1 :    "parameter error";
2 :    "unable to decode parameter string";
3 :    "unable to write properties";
4 :     custom error string depending of the properties;
x :    "unknown error";
```

```
custom error messages :
```

```
"error setting property server." + XXX
"set cache_size : illegal value "+ XXX + " 1000 KB minimum"
"set cache_size : "+ XXX + " exceeds available memory
```

examples

```
Server.SetProperties ( " name = example; indexation_cache_size = 16000"  )
```

is equivalent to

```
Server.SetProperties ( "name = example", "indexation_cache_size = 16000"  )
```

```
returns
void
```

SERVER.GetProperties

description

Gets one or several Session properties.

To access one property, use a directive as : propertyname
To access several properties values in one command, separate each directive with the character semi column (;)

if there are no parameter, all properties will be accessed

prototype

```
GetProperties (string accessor);
accessor = "propertyname1; …;propertynameN"
```

defaults

void

error messages

```
x :    "unknown error";
```

examples

Server.GetProperties ( )

prop_name	prop_value	prop_type	prop_access
name	SNet_1254	string	rw
port	1254	string	r
type	mARC syncytiotrophoblaste	string	r
model	16 Mp	string	r
version	1.0 Beta	string	r
build	1.2014.06.19.15.05 Win x86_64 release	string	r
connection_count	2	int32	r
command_threads	8	int32	rw
time_local	12:43:26:409	string	r
time_gmt	10:43:26:409	string	r
up_time	65310.6679863281	double	r
idle_time	43700.0474310014	double	r
cache_size	32000	uint32	rw
cache_used	66	uint32	r
cache_hits	0	int32	r
exec_timeout_default	5000	int32	rw
session_timeout_default	-1	int32	rw
marc_relations	0	int64	r
marc_shapes	0	int64	r
marc_references	0	int64	r
marc_particles	1	int64	r
indexation_cache_used	0	int64	r
indexation_cache_size	8000	int64	rw
indexation_timeout	300	int64	rw
marc_quality	6.	double	r

Server.GetProperties ("name ; version")

prop_name	prop_value	prop_type	prop_access
name	SNet_1254	string	rw
version	1.0 Beta	string	r

SERVER.GetConnected

description
Gets all currently socket connected to the server

prototype
```
Server.GetConnected ( int32 start, int32 count)
```

defaults
```
start = 1   (the first connection)
count = -1 (all until the end is reached)
```

error messages
```
1 :    "parameter start out of range";
2 :    "parameter count out of range";
3 :    "unable generate response string";
x :    "unknown error";
```

```
Server.GetConnected ( )
```

IP	Port
127.0.0.1	3431
127.0.0.1	3432
127.0.0.1	3433

```
Server.GetConnected (2)
```

IP	Port
127.0.0.1	3433
127.0.0.1	3434

```
Server.GetConnected (1,2)
```

IP	Port
127.0.0.1	3431
127.0.0.1	3432

```
Server.GetConnected (1,10)
```

IP	Port
127.0.0.1	3431
127.0.0.1	3432
127.0.0.1	3433

parameters **start** and **count** are usefull when the number of connections is high.

it is then possible to get all the connections by fetching them 100 by 100 in order to optimize the network and server load.

start is in base 1 eg, the first connection is number 1

```
Server.GetConnected (1, 10)        //gets the 10 first connections
Server.GetConnected (11, 10)       //gets the 10 next connections
....
Server.GetConnected (91, 10)       //gets 10 connections from 91 to 100
```

if the returned number of lines is less than the value of parameter **count**, all connections will have been fetched.

SERVER.GetTasks

description

Sometimes, the Server performs background tasks that are logged.
it is possible to get the description of all current tasks through this method

it returns a list off all current server tasks

prototype

```
Server.GetTasks (void)
```

defaults

void

error messages

```
1 :    "mARC access error";
2 :    "unable to lock tasklist";
3 :    "unable to handle parameters start an count";
4 :    "unable generate response string";
x:     "unknown error";
```

examples

```
Server.GetTasks ()
```

Task	completion	current	from	to	elapsed
Rebuilding mARC from DB:	5	2008	0	33940	983

A task description is always formatted according to following format

the task description

the task completion in %

the task limits

the elapsed time in ms, since the start of the task

Session

property name		description	defaults & values
name	rw	name of the session	anonymous
last_time	r	Last execution time for this session in ms	0.0
owner_IP	r	IP adress of the session's owner	void
owner_port	r	Session's owner port	void
id	r	The session Id for this session	void
priority	rw	Priority of this session	3 value in [0 – 7]
session_timeout	rw	Idle activity before closing automatically the session. in ms	-1 (for ever)
exec_timeout	rw	5000	int32
context_count	r	size of the context's stack	0
result_count	r	size of the result's stack	0
spectrum_string		spectrum string descriptor	""
profiler_context_string	rw	profiler string descriptor	""
inhibitor_context_string	rw	inhibitor string descriptor	""
result_max_stack_size	rw	maximum result's stack size	16
result_embedded	rw	is fetch of a RS a GPbinary protocole string. if false a usual string is emitted	true
result_line_separator	rw	line separator when result_embedded = false	default CRLF [CRLF, CR, LF,TAB,..]
result_column_separator	rw	column separator when result_embedded = false	default "," [CRLF, CR, LF,TAB,..]
result_DBCursor	rw	next position for a fetch	default 0] 0, +INF [

Methods	parameters	parameters values
Connect	NULL	
Clear	string option1,.., string optionN	["all", "contexts", results", "profiler", "inhibitor", "variables"]
ContextToInhibitor	void	
ContextToProfiler	void	
InhibitorToContext	void	
ProfilerToContext	void	
GetInstances	int32 start, int32 count	start = 1 count = -1
GetLastDBInfo	void	
GetProperties	string accessor	"propertyname1; …;propertynameN"
SetProperties	string accessor, …string accessorN	
MarcSave	void	
MarcReload	void	
MarcClear	void	
MarcRebuild	string columns, int32 begin_rowid, int32 end_row_id, string mode	mode = ["none", "ref"]
MarcPublish	void	
DocToContext	int32 rowid, bool spectrum	spectrum = false
ContextToDoc	void	
StringToContext	string signal, bool learn	learn = false
ContextToContext	void	
Store	string text, string mode, Int32 rowid	mode = ["raw", "ranked", "unique"] rowid = -1 (no indexation)
Index	int32 rowid	
Predict	string text	

SESSION.Connect

description
Main connection entry point. this method return a valid session Id that can be used for further commands

prototype
Session.Connect (NULL)

defaults
void

error messages

```
1 :    "session creation failure";
2 :    "unable link session to socket";
3 :    "system error";
x :    "unknown error";
```

examples

see Connecting to a mARC Server

SESSION.GetInstances

description

Fetches the list of all active session at server level.

a maximum of 100 session descriptors can be accessed at each method call

prototype

```
Session.Getinstances (int32 start, int32 count)
maximum count value = 100;
```

defaults

```
start = 1   (the first connection)
count = -1 (all until the end is reached)
```

error messages

```
1 :     "illegal parameter start. start must be >=1";
2 :     "unable to get parameter count";
3 :     "unable to generate response string";
x :     "unknown";
```

examples

```
Session.Getinstances ( );
```

id	name	persistant	owner_ip	owner_port	priority	exec_timeout	session_timeout
0_4HMa51d57vcgB		false	127.0.0.1	1193	3	5000	-1
1_568w9O13kp84X	main_session	false	127.0.0.1	1194	3	5000	-1
2_NvS0cS4Qei6		false	127.0.0.1	1195	3	5000	-1
3_mnmkS8r382ftSy		false	127.0.0.1	1196	3	5000	-1

```
Session.Getinstances (2);
```

id	name	persistant	owner_ip	owner_port	priority	exec_timeout	session_timeout
1_568w9O13kp84X	main_session	false	127.0.0.1	1194	3	5000	-1
2_NvS0cS4Qei6		false	127.0.0.1	1195	3	5000	-1
3_mnmkS8r382ftSy		false	127.0.0.1	1196	3	5000	-1

```
Session.Getinstances (1, 2);
```

id	name	persistant	owner_ip	owner_port	priority	exec_timeout	session_timeout
0_4HMa51d57vcgB		false	127.0.0.1	1193	3	5000	-1
1_568w9O13kp84X	main_session	false	127.0.0.1	1194	3	5000	-1

parameters **start** and **count** are usefull when the number of sessions is high.

it is then possible to get all the connections by fetching them 100 by 100 in order to optimize the network and server load.

start is in base 1 eg, the first session in the list is at index 1

if the returned number of lines is less than the value of parameter **count**, all connections will have been fetched.

SESSION.Clear

description

Resets all properties and objects of a session.
Especially the Contexts and Results.

It frees memory and ressources used a session, including

Contexts stack

Results stack

Inhibitor context

Profiler context

Session variables

if no options are used, all Session object will be cleared

depending of the option used, only one or several selected ressources will be cleared

prototype

```
Session.Clear (string option1, string option2…, string optionN)
option = ["all", "contexts", "results","profiler","inhibitor","variables"]
```

defaults

option = "all";

error messages

```
1 :   "illegal option. must be [all, contexts, results,profiler,inhibitor,variables]";
x :    "unknown";
```

examples

```
Session.Clear ()
```

everything is cleared

```
Session.Clear ("contexts", "results")

both Context, and Result Stacks will be cleared
```

SESSION.GetProperties

description

Gets one or several Session properties.

To access one property, use a directive as : propertyname
To access several properties values in one command, separate each directive with the character semi column (;)

if there are no parameter, all properties will be accessed

prototype

```
GetProperties (string accessor);
accessor = "propertyname1; …;propertynameN"
```

defaults

void

error messages

```
1 :    "parameter error";
2 :    "unable to decode parameter string";
3 :    "unknown property server.XXX";
4 :    "unable to generate response string";
x :     "unknown";
```

examples

```
GetProperties ();
```

prop_name	prop_value	prop_type	prop_access
name	anonymous	string	rw
last_time	2.214	double	r
owner_IP	127.0.0.1	string	r
owner_port	1715	string	r
id	0	string	r
priority	3	uint8	rw
session_timeout	-1	int32	rw
exec_timeout	5000	int32	rw
context_count	1	int32	r
result_count	0	int32	r
spectrum_string	min_atom = 1; max_atom = -1; min_generality = 0; max_generality = -1; min_activity = 25; max_activity = -1; max_context = 5; min_context_size = 2; max_context_size = 25; min_context_activity = 25; max_context_activity = -1; max_record = 1000; depth = 0; evaluate = true;	string	rw
profiler_context_string	# mARC CONTEXT 200 afrique 100 évolution 100	string	rw
inhibitor_context_string		string	rw
result_max_stack_size	16	int32	rw
result_embedded	true	bool	rw
result_line_separator	CRLF	string	rw
result_column_separator	,	string	rw
result_DBCursor	0	int32	rw

```
GetProperties (" last_time; context_count");
```

prop_name	prop_value	prop_type	prop_access
last_time	6.1e-002	double	r
context_count	1	int32	r

SESSION.SetProperties

description

Access to session's properties

Trying to change the value of a Read Only property will **not** generate an error.

To see which properties are avalaible, see SESSION.GetProperties

To change one property, use a directive as : propertyname = propertyvalue
To change several properties values in one command, separate each directive with the character semi column (;)

an accessor is a string like :

"propertyname = value"

and can be extended like

"propertyname1 = value1, … propertynameN = valueN"

Depending of your client application using only one extended accessor, as a parameter, is equivalent as using several accessors as parameters

prototype

```
Session.SetProperties ( string accessor, …string accessorN )
or simply
SetProperties ( string accessor, …string accessorN )
```

defaults

void

error messages

```
1 :    "parameter error";
2 :    "unable to decode parameter string";
3 :    "unable to write properties";
4 :     custom error message depending of properties;
x :     "unknown";
```

```
custom properties messages
"error setting property session.XXX"
"error session_timeout cannot be infinite (value <=0) for a persistant session. reset to default ";
```

examples

```
session.getProperties ( " priority ; session_timeout" );
```

prop_name	prop_value	prop_type	prop_access
priority	3	uint8	rw
session_timeout	-1	int32	rw

```
session.setProperties ( "priority  = 4; session_timeout = 3000" ); or
session.setProperties ( "priority  = 4","session_timeout = 3000" );
```

```
session.getProperties ( " priority ; session_timeout" );
```

prop_name	prop_value	prop_type	prop_access
priority	4	uint8	rw
session_timeout	3000	int32	rw

SESSION.ContextToInhibitor

description

Gets the topmost context of the context's stack, and consolidate the inhibitor context with it.

The main differences with affecting the profiler through the Session.inhibitor_context_string property are :

- it is faster
- the inhibitor context will be consolidated, eg, it's previous content will be kept

prototype

Session.ContextToInhibitor (void)

defaults

void

error messages

```
1 :    "empty stack";
2 :    "unable to transfert inhibitor from stack";
3 :    "output serialization";
x :    "unknown";
```

examples

```
Clear();
GetProperties ( " context_count ; inhibitor_context_string;" )
```

prop_name	prop_value	prop_type	prop_access
context_count	0	int32	r
inhibitor_context_string		string	rw

```
stringToContext  ("afrique évolution") ;
```

```
GetProperties ( " context_count ; inhibitor_context_string" )
```

prop_name	prop_value	prop_type	prop_access
context_count	1	int32	r
inhibitor_context_string		string	rw

contextToInhibitor ( ) ;

```
GetProperties ( " context_count ; inhibitor_context_string" )
```

prop_name	prop_value	prop_type	prop_access
context_count	1	int32	r
inhibitor_context_string	# mARC CONTEXT 200 afrique 100 évolution 100	string	rw

SESSION.ContextToProfiler

description

Gets the topmost context of the context's stack, and consolidate the profiler context with it.

The main differences with affecting the profiler through the Session.profiler_context_string property are :

- it is faster
- the profiler context will be consolidated, eg, it's previous content will be kept

prototype

Session.ContextToInhibitor (void)

defaults

void

error messages

```
1 :    "empty stack";
2 :    "unable to transfert profiler from stack";
3 :    "output serialization";
x :    "unknown";
```

examples

```
Clear();
GetProperties ( " context_count ; profiler_context_string;" )
```

prop_name	prop_value	prop_type	prop_access
context_count	0	int32	r
profiler_context_string		string	rw

```
stringToContext  ("afrique évolution") ;

GetProperties ( " context_count ; profiler_context_string" )
```

prop_name	prop_value	prop_type	prop_access
context_count	1	int32	r
profiler_context_string		string	rw

contextToProfiler ( ) ;

```
GetProperties ( " context_count ; profiler_context_string" )
```

prop_name	prop_value	prop_type	prop_access
context_count	1	int32	r
profiler_context_string	# mARC CONTEXT 200 afrique 100 évolution 100	string	rw

SESSION.InhibitorToContext

description

Transfers the inhibitor context into a new context on the context's stack

if the *inhibitor_context_string* is void, the new context on top of the stack is empty (it's count is 0)

prototype

```
InhibitorToContext(void)
```

defaults

```
void
```

error messages

```
1 :    "unable to create new context on stack";
2 :    "unable to transfert inhibitor on stack";
3 :    "output serialization";
x :    "unknown";
```

examples

```
InhibitorToContext ( );
```

SESSION.ProfilerToContext

description

Transfers the profiler context into a new context on the context's stack

if the *profiler_context_string* is void, the new context on top of the stack is empty (it's count is 0)

prototype

ProfilerToContext(void)

defaults

void

error messages

```
1 :    "unable to create new context on stack";
2 :    "unable to transfert profiler on stack";
3 :    "output serialization";
x :    "unknown";
```

examples

```
ProfilerToContext ( );
```

SESSION.GetLastDBInfo

description
Retrieves informations about the **last** Data Base operation that occured inside the present session.

prototype
SESSION.GetLastDBInfo (void)

defaults
void

error messages
```
x :    "unknown error";
```

examples

```
table:eumaster.delete ( " 33237"   ) ;
```

deleted
1

```
SESSION.GetLastDBInfo (  ) ;
```
or simply
```
GetLastDBInfo (  ) ;
```

table	operation	id	status
EuMaster	delete	33237	ok

```
table:eudetail.delete ( "33237", " 33236", "33235 " ) ;
```

deleted
3

```
GetLastDBInfo (  ) ;
```

table	operation	id	status
EuDetail	delete	33235	ok

SESSION.MarcSave

description

Saving the mARC binary to the disk

the marc will be saved in the mARC repository in the folder

data\knowledge\marc\marc.knw.

during the process, the previous version is saved as *marc.bak*

the *marc.bak* file will be deleted if the saving process if successful.

in case of incident, and if a *marc.bak* file can be seen in the *data\knowledge\marc\* the server will try to restore the previous state. If not possible, the server will request human supervision, and shutdown.

all these events will be logged in the file *\message\year_month_day.log*

a server task will be created, that can be caught through a

`Server.GetTasks ( );` command

an internal MarcPublish will be done if the indexation buffer is not empty

prototype

```
MarcSave (void)
this process is logged
```

defaults

```
void
```

error messages

```
1 :    "mARC Read Locked";
2 :    "unable to save mARC. see logfile";
3 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
MarcSave ( );
```

```
Server.GetTasks ()
```

Task	Completion	Current	From	To
saving mARC to file:	1	17000	0	1662033

SESSION.MarcReload

not yet implemented

description

Reloads the mARC from it's last state.

a server task will be created

this event is logged

during this operation, all requests, except Server.xxx requests will be returned with an error "message = mARC reloading, please wait"

prototype

MarcReload (void) ;

defaults

void

error messages

examples

```
MarcReload ();
```

SESSION.MarcClear

description
Clears the mARC content, and the document context field (KNW_ABSTRACT) of all lines of the associated master table, if there is one.

this event will be logged.

a server task is created

after a clear () command, the content of the mARC is void, and all indexation informations will be cleared too.

the content of the master table, except the KNW_Abstract field, is not affected

the empty mARC will be saved on disk, and the previous disk state will be destroyed

prototype
MarcClear (void)

defaults

error messages
```
1 :    "mARC Write Locked";
2 :    "unable void mARC";
3 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
MarcClear ( );
```

the mARC will be completely empty.

the KNW_Abstract fields of the associated master table are set to NULL.

SESSION.MarcRebuild

description

Use only if a master table has been created and linked to the mARC.

This is an administration method

It is a convenient and fast way to reconstruct the mARC content by using the content of the mARC master table as data source.

a server task is created

the events related to the use of this method are logged

prototype

```
MarcRebuild (
                    string columns,
                    int32 begin_rowid,
                    int32 end_row_id,
                    string mode
                    )
```

mode can be one of the following

"none", "ref"

if *mode* is "ref", an indexation will be performed using the current processed rowid of the master table

defaults

mode = "none"

error messages

```
1 :    "mARC Write Locked";
2 :    "rebuild failure";
3 :    "unable to generate response string";
x :    "unknown error";
```

Supposing that the master table contains the fields : *text*, *title* among others.

```
MarcRebuild ("text title", 0, 1000);
```

will store the content of the fields *text* and *title* into the mARC, for each lines of the table from line 0 (included) to line 1000 (included)

```
MarcRebuild ("text title", 0, 1000, "ref");
```

will store AND index the content of the fields *text* and *title* into the mARC, for each lines of the table from line 0 (included) to line 1000 (included)

SESSION.MarcPublish

description
used by applications that uses the internal indexation system of the mARC Server.

When a document is indexed through the Store ( ), or Index ( ) commands, the indexation information is stored into the Indexation Buffer.

at this time, the documents are not yet accessible through a `ContextToDoc ( )` command.

The indexation buffer will automatically flushed when :

- it reaches it's maximum size (property Server.indexation_cache_size (in KB))
- if a `MarcSave ()` command is issued
- if a `MarcPublish( )` command is issued

once the indexation cache flushed, all documents indexed during this time will be publishad, eg, they will be accessible through requests.

prototype
MarcPublish (void)

defaults
void

error messages
```
1 :    "mARC write locked";
2 :    "publish failure";
3 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
MarcPublish ( ) ;
```

SESSION. DocToContext

description

available only when the indexation system of the mARC Server is used, eg, when a master table is associated to the mARC.

the indexation context of a document of the master table (the knw_abstract column content of the document) will be transferred into a new context on top of the contexts stack.

further operation on context can the be performed in order to contextually analyze the document.

if the spectrum parameter is set to true, the session Spectrum will automatically be applies to the new context

prototype

```
DocToContext (int32 rowid, bool spectrum);
```

defaults

```
spectrum = false
```

error messages

```
1 :    "unable to access parameters";
2 :    "missing master table";
3 :    "unable to create context";
4 :    "bad or corrupted dbid ";
5 :    "error during contextual analysis ";
6 :    "unable to generate response string";
x :    "unknown";
```

examples

```
DocToContext (2263);
contexts.getproperties ( " context_string")
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT 6641 liaison_chimique 100 minérale 100 cristaux 99 chimique 99 liaison 98 atome 97 1 96 et_la 95 l 94 la 93 et 92 ioniques 90 électronégatif 88 électronégativité 87 chargés 87 covalent 86 polyèdre 85 cation 85 encombrement 84 électronique_d_un_atome 83 diriger 83 électronique 81 x_B 80 B 80 x 79 sont_donc 78 donc 78 sont 77 mesure_que 76 que 76 mesure 75 subissent 74 électronégativités 73 fluor 73 …	string	rw

```
DocToContext (2263,true);
contexts.getproperties ( " context_string") ;
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT 3501 charge_partielle 100 électronique_d_un_atome 91 fluor 87 attirer 87 décroît 86 croît 86 électronégativités 86 cations 83 cation 83 lui_donner 82 x_B 82 polyèdre 80 polarisation 80 diriger 75 décrira 74 encombrement 73 chargés 73 électronégativité 72 Li 72 subissent 72 covalent 68 électronégatif 68 négative 68 ionique 64 renforcement 64 déformation 57 nuage 57 fréquent 51 …	string	rw

SESSION.ContextToDoc

description

available only when the indexation system of the mARC Server is used, eg, when a master table is associated to the mARC.

the topmost context of the context's stack is converted into a new result set of documents, that are ranked according to the importance of that context inside them.

it's behavior is dependant of the current spectrum, especially from the following properties of the spectrum.

```
Spectrum.max_record
Spectrum.min_activity
```

prototype

```
ContextToDoc (void)
```

defaults

```
void
```

error messages

```
1 :    "empty contexts stack";
2 :    "unable to convert context into RS";
3 :    "unable to generate response string";
x :    "unknown";
```

examples

```
clear();
StringToContext ("atome");
ContextToDoc ();
Results.SortBy("Act","false");
Results.SetProperties ( " format = "rowid act titre""  ) ;
Results.Fetch ( "20","1"  );
```

RowId	Act	titre
11853	259	ATOME
2263	179	CRISTAUX - Cristallochimie minérale
13888	148	ATOME
15854	100	ATOME
5133	100	ATOME
1365	100	ATOME
16814	99	ATOME HABILLÉ
22230	98	BORE
28295	88	BÉRYLLIUM
12821	88	ATOMIQUE (PHYSIQUE)
29637	87	DALTON
20138	87	AROMATICITÉ
32876	86	COORDINATION (chimie) - Chimie de coordination
13833	86	ARYLES (RADICAUX)
6296	85	X (RAYONS)
26062	84	LIAISONS CHIMIQUES - Liaison et classification
16372	84	POMPAGE OPTIQUE
3976	84	AROMATICITÉ
19054	81	QUANTIQUE (MÉCANIQUE) - Le formalisme de la mécanique quantique
13953	81	BENZÉNOÏDES

SESSION.StringToContext

description

converts a string to a new context on the context's stack.

it is equivalent to affecting contexts.context_string property, except that

- it creates a new context on top of the stack
- it applies the current Spectrum to the process of converting the string signal into an internal mARC context.
- it can learn new shapes if the optional *learn* parameter is set to true

by default no new shapes will be learned during this process.

prototype

StringToContext (string signal, bool learn)

defaults

learn = false

error messages

```
2 :    "unable to access parameters";
3 :    "unable to convert string to context";
x :    "unknown error";
```

examples

session.GetProperties (" context_Count")

prop_name	prop_value	prop_type	prop_access
context_count	0	int32	r

StringToContext (" armée rouge seconde guerre mondiale") ;

session.GetProperties (" context_Count")

prop_name	prop_value	prop_type	prop_access
context_count	1	int32	r

contexts.getproperties (" context_string ; atom_count")

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT 200 armée_rouge 100 seconde_guerre_mondiale 100	string	rw
atom_count	2	int32	rw

SESSION: ContextToContext

Dev and quality control pending

description

It is one the main methos in order to interact and query with the mARC.

It uses the topmost context of the context's stack in order to perform a contextual analysis

It's behavior will depend on :

- the current Spectrum
- the content of the profiler context SESSION.ContextToProfiler
- the content of the inhibitor context SESSION.ContextToInhibitor

a full explanation will be found in the mARC user documentation, chapter contextual and semantic programming.

at least three new contexts (maybe empty) will be created on the contexts Stack.

it returns the number of newly created sub contexts.

If N subcontexts, the two last represents respectively the categories contexts, and the shapes contexts, the other N-2 subcontexts are associative contexts.

source context : source

example of categories context : armée_rouge rouge_sang couleur_rouge….

example of shape context : rouges rougie….

example of associative context : blanc, noir, vert, bleu, blanche….

example of associative context : lumière longeur_d_onde nm …

prototype

```
Session.ContextToContext (void)
```

or simply

```
ContextToContext (void)
```

defaults

void

error messages

examples

```
ContextToContext ();
```

sub_contexts
3

example ***a simple request script***

```
//step 1 : preparing the script
clear();
SetSpectrum (evaluate = false; max_record = 100; max_atom = 20; depth = 8 ;min_activity = 25);
SetSpectrum (min_context_activity = 25; max_context = 5; min_context_size = 5);

//step 2 :inputing the request
StringToContext ( mer roug barent okh ); //the request
Contexts.SetProperties (name = signal);
```

#	act	atoms	name
1	-1	4	signal

#	shape	activity	gen_class	generality	id
1	okh	99	0	3	139905
2	roug	99	0	3	139906
3	mer	98	11	191	205
4	barent	97	0	3	80252

```
//step 3 : contextual processing of the request
ContextToContext ();
```

#	act	atoms	name
1	100	20	ctx_5
2	64	11	ctx_4
3	48	8	ctx_3
4	43	6	ctx_2
5	26	6	ctx_1
6	1810	20	shapes
7	490	5	categories
8	-1	4	atoms
9	196	4	result
10	-1	4	signal

#	shape	activity	gen_class	generality	id
1	mer_d_okhotsk	100	0	16	82593
2	mer_rouge	99	2	36	55745
3	voisins_de_la_mer	98	0	12	53932
4	mer_de_barents	97	1	27	103491
5	glaces_de_mer	96	1	20	74021

#	shape	activity	gen_class	generality	id
1	okh	50	0	3	139905
2	roug	49	0	3	139906
3	mer	49	11	191	205
4	barent	48	0	3	80252

```
//step 4 : exploiting the result, by accessing document
Contexts.OnTop(result);                         //moving the result on top of the stack
```

#	act	atoms	name
1	196	4	result
2	100	20	ctx_5
3	64	11	ctx_4
4	48	8	ctx_3
5	43	6	ctx_2
6	26	6	ctx_1
7	1810	20	shapes
8	490	5	categories
9	-1	4	atoms
10	-1	4	signal

```
Contexts.SortBy (activity, descending );
//computing a contextual database RS

ContextToDoc();
//step 5 : preparing  the result set for display
Results.SortBy(act,descending);
//this RS contains contextually activated docs (best quality)
```

```
Results.SetProperties (name = context);
Results.Dup();
//this RS contains more keyword activated docs (less quality)
Results.SetProperties (name = no_context); Results.SelectBy (act,<=,100);
Results.Normalize();
Results.Swap();
Results.SelectBy (act,>,100);
Results.Normalize();
```

#	count	name	table
1	9	context	EuMaster
2	98	no_context	EuMaster

	int32	int32	char
#	RowId	Act	titre
0	2970	100	ROUGE (MER)
1	3659	65	ÉGYPTE DEPUIS L'ISLAM - La mise en valeur du pays
2	1763	60	SUEZ
3	7198	55	SÉDIMENTOLOGIE
4	23252	52	SIBÉRIE
5	17844	52	U.R.S.S. ET EX-U.R.S.S. - Géographie
6	487	48	ÉGYPTE DEPUIS L'ISLAM - La mise en valeur du pays
7	12005	48	SIBÉRIE
8	4727	43	EXODE (histoire des Hébreux)

the complete script is following

/____

/ a simple search engine request processor

/____

```
clear();
SetSpectrum (evaluate = false; max_record = 100; max_atom = 20; depth = 8 ;min_activity = 25);
SetSpectrum (min_context_activity = 25; max_context = 5; min_context_size = 5);

StringToContext ( mer rouge barente okhotsk ); //the request
Contexts.SetProperties (name = signal);
ContextToContext ();
Contexts.OnTop(result);
Contexts.SortBy (activity, descending );
ContextToDoc();

Results.SortBy(act,descending);
Results.SetProperties (name = context);
Results.Dup();
Results.SetProperties (name = no_context);
Results.SelectBy (act,<=,100);
Results.Normalize();
Results.Swap();
Results.SelectBy (act,>,100);
Results.Normalize();
```

(to be continued on next page…)

There are several way to use the contextual elements on the stack

1/ retrieving the contextual result set (less quality)

```
/putting the RS on top of the stack (in this case Results.Swap() would have
been equivalent
Results.OnTop (context);

/gets the table associated with the RS
Results.GetProperties (owner_table);
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
owner_table	EuMaster	string	rw

```
/gets the format of the table associated with the RS
table:eumaster.GetStructure ( );
```

name	type	size
id	INT32	0
titre	CHAR	106
soustitre	CHAR	141
rtf	STRING	0
texte	STRING	0
KNW_ABSTRACT	ABSTRACT	0
KNW_LANGAGE	INT32	0
KNW_MEANING	INT32	0

```
/sets the output format of the RS, before fetching the data
Results.SetProperties (format = rowid act titre);

/fetching the 10 first lines of the RS
Results.Fetch (10,1);
```

	int32	int32	char
#	RowId	Act	titre
0	2970	100	ROUGE (MER)
1	3659	65	ÉGYPTE DEPUIS L'ISLAM - La mise en valeur du pays
2	1763	60	SUEZ
3	7198	55	SÉDIMENTOLOGIE
4	23252	52	SIBÉRIE
5	17844	52	U.R.S.S. ET EX-U.R.S.S. - Géographie
6	487	48	ÉGYPTE DEPUIS L'ISLAM - La mise en valeur du pays
7	12005	48	SIBÉRIE
8	4727	43	EXODE (histoire des Hébreux)

(to be continued on next page…)

1/ retrieving the 10 first categories detected for the request

```
/retrieving data in a table
Contexts.OnTop (categories) ;
Contexts.Fetch(10,1);
```

#	shape	activity	gen_class	generality	id
1	mer_d_okhotsk	100	0	16	82593
2	mer_rouge	99	2	36	55745
3	voisins_de_la_mer	98	0	12	53932
4	mer_de_barents	97	1	27	103491
5	glaces_de_mer	96	1	20	74021

```
/or retrieving data as a string
contexts.GetProperties (context_string, categories);
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT 490 mer_d_okhotsk 100 mer_rouge 99 voisins_de_la_mer 98 mer_de_barents 97 glaces_de_mer 96	string	rw

NB :

such a string can be directly used by the method `StringToContext` in order to evaluate a new request string

the tabular output through `Contexts.Fetch`, is more easy to use for application that have to layout the data on a screen, or to design user interfaces.

(to be continued on next page…)

example ***using a document of the master table as a query***

a very similar script can be used in order to get a query by document, or a recommendation of articles based on implicit contextual similarities.

```
clear();
SetSpectrum (evaluate = true; max_record = 100; max_atom = 20; depth = 8 ;min_activity = 25);

/gets the context of document 16274 of the master table
DocToContext (16274);
/gets the more precise term of the indexation vector
Contexts.SortBy (generality, ascending);
/performs an implicit contextual analysis (evaluate = true) and applies the spectrum limits
ApplySpectrum ();

ContextToDoc();

Results.SortBy(act,descending);
Results.SetProperties (name = context);
Results.Dup();
Results.SetProperties (name = no_context);
Results.SelectBy (act,<=,100);
Results.Normalize();
Results.Swap();
Results.SelectBy (act,>,100);
Results.Normalize();
```

#	act	atoms	name
1	1812	17	doc_16274

#	shape	activity	gen_class	generality	id
1	superfluidité	100	1	32	79236
2	superfluide	88	1	20	56825
3	très_hautes	70	2	34	79334
4	hélium_4	65	1	32	56894
5	bose	64	2	34	56827
6	1020	53	2	34	58869
7	ont_aussi	51	2	44	71361
8	bose_einstein	51	1	19	90304
9	hélium_3	43	2	35	14779
10	cryogénie	38	1	27	113979
11	hélium	32	2	43	5296
12	devrait_être	29	2	44	63556
13	bosons	29	2	37	41650
14	neutrons	29	3	49	2579
15	orienté	26	2	41	6948
16	déformer	25	2	35	8750
17	vibrer	25	2	35	97634

SESSION: Store

description

the main back office method for the mARC.

it stores a signal into the mARC

new shapes and contextual structures will be detected and learned by the mARC.

there is no need for tuning parameters, the mARC handles automatically the learning process.

it returns void

By default, the Store method only performs contextual learning

but depending of the parameters *mode* and *rowid*, it can perform concurrently :

- creation of a term vector on the top of the context's stack
- index the incoming text signal to a rowid of the associated master table of the mARC
- get a term vector representing the incoming text signal

mode can be one of the following value

- "none"
- "ranked"
- "raw"
- "unique"

Warning : if mode is different of "none", a new context is created on top of the context's stack

therefore, you will have to handle it, by dropping after use if necessary. This can be done by using *contexts*.Drop ( ), *session*.Clear("contexts")

mode = "ranked" : the term vector is exactly the same than the term vector evaluated by the default indexation algorithm. This can be used if you do not want to use the internal indexation mechanism of the mARC Server.

```
example :
mode = "ranked"
Store ("one text to be stored or to be learned", "ranked")
or equivalent
Store ("one text", "ranked", -1)
possible context :
stored 100 text 99 learned 98 to_be 97 one 96 to 95 be 94 or 93

mode = "raw"
possible context :
one 100 text 100 to_be 100 stored 100 or 100 to_be 100 learned 100

mode = "unique"
possible context :
one 100 text 100 to_be 100 stored 100 or 100 learned 100
```

prototype

```
Store (string text, string mode, Int32 rowid)
```

defaults

```
mode = "none"
rowid = -1
```

error messages

```
1 :    "unable to access parameters";
2 :    "empty parameter text";
3 :    "invalid parameter mode. must be [none, raw,ranked,unique]";
4 :    "undefined master table";
5 :    "store failure";
6 :    "unable to generate response string";
x :    "unknown error";
```

examples

SESSION.Index

description

Indexes and links the topmost context of the context's stack to a master table rowed.

this method can be used to build a custom indexation algorithm.

its is different from the store method, since it does not use the default indexation algorithm

a full explanation of the indexation process will be found in the mARC user Documentation, chapter indexation

prototype

```
Session.Index (int32 rowid)
```

defaults

```
void
```

error messages

error messages

```
1 :    "unable to get parameters ";
2 :    "empty stack ";
3 :    "no master table ";
4 :     dbid  + " is not a valid line in the master table  ";
5 :    "indexation failure ";
x :    "unknown ";
```

examples

SESSION.GetSpectrum

description

Gets the current contextual spectrum property of the current session, under the shape of a table of a set of properties.

If you need a string compatible with the SetSpectrum methos, you can use the Session.GetProperty ("spectrum_string") instead.

prototype

```
Session.GetSpectrum (void);
or more simply
GetSpectrum ();
```

defaults

```
void
```

error messages

```
x :    "unknown ";
```

examples

```
GetSpectrum (  );
```

string	string	string
name	value	type
min_atom	1	int32
max_atom	-1	int32
min_generality	0	int32
max_generality	99	int32
min_activity	1	int32
max_activity	-1	int32
max_context	5	int32
min_context_size	2	int32
max_context_size	25	int32
min_context_activity	25	int32
max_context_activity	-1	int32
max_record	1000	int32
depth	0	int32
evaluate	false	bool

SESSION.SetSpectrum

description

Sets the current contextual spectrum property of the current session, under the shape of a table of a set of properties.

SetSpectrum can change one or several spectrum properties at a time, by using an accessor.

prototype

```
Session.SetSpectrum (string accessor);
or more simply
SetSpectrum (string accessor);
accessor = "propertyname1; …;propertynameN"
```

defaults

```
void
```

error messages

```
1 :    "unable to decode parameter string";
3 :    "unable to generate response string";
x:     "unknown";
```

examples

```
SetSpectrum ("min_atom = 3 ; min_generality = 5; evaluate = true", );
SetSpectrum  ( <50 min_atom = 3 ; min_generality = 5; evaluate = true/>);
VOID
GetSpectrum (  );
```

string	string	string
name	value	type
min_atom	**3**	int32
max_atom	-1	int32
min_generality	**5**	int32
max_generality	99	int32
min_activity	1	int32
max_activity	-1	int32
max_context	5	int32
min_context_size	2	int32
max_context_size	25	int32
min_context_activity	25	int32
max_context_activity	-1	int32
max_record	1000	int32
depth	0	int32
evaluate	**true**	bool

SESSION.ApplySpectrum

description
Modifies the context on top of the context's Stack, according to the current Spectrum of the session

prototype
```
Session.ApplySpectrum (void)
```

defaults
```
void
```

error messages

```
1 :    "unable to get parameters ";
2 :    "empty stack ";
3 :    "index out of range, must be in range [1,stack_count] ";
4 :    "unable to apply spectrum ";
x :    "unknown ";
```

examples

SESSION.Completion

not yet implemented

description

This method is useful in order to predict or complete a user entry string.

it return NULL

2 new contexts are created on top of the context's stack

WARNING : you will have to handle these 2 new contexts, generally to drop them after use

the topmost context will contain the best possible future entries

the second one will contain what has been detected by the mARC

prototype

```
Session.Predict (string text)
```

defaults

void

error messages

examples

```
Predict ("armée ro");
```

context #1 :	rouge 100 romaine 95 protection 43…..
context #2 :	armée 100

Table

A TABLE object, is a classical database table.
Data a structured in lines and columns
Each columns name in a table must be unique.
Columns of type STRING and BINARY are variable size columns
For one line, the value of a column can be NULL
Each line is referenced by a unique ID, called **RowId**
RowId is an INT32 type
Each line in a table can be accessed through its **RowId**
A table can own several indexes linked to its columns
There are 2 types of indexes, **Bindex** and **Kindex**
Kindexes can only be created on a table with the master attribute
A **master table** is allways owned by the mARC
There is **at maximum one** master table
Every operation on a table generates a result set (**RS**)
A **RS** is a collection of triplets under the shape : RowId, Activity, Origin
Each triplet is called a **reference**
Values of a **reference** can be accessed trough the symbolic variable **RowId**, **Act**, **Origin**

NULL values :

when data in a column of a line is not affected, it has the symbolic value NULL.
This attribute is implemented as a data type value of the column, according to the table below

type	NULL value
INT8	-128
UINT8	255
INT (INT32)	-2 147 483 648
UINT (UINT32)	4 294 967 295
SIMPLEDATE (SD)	xx :15 :xx (mois = 15)
CHAR	Size = 255
FLOAT	1.17549435E-38
DOUBLE	2.2250738585072014E-308
INT64	-9 223 372 036 854 775 808
UINT64	18 446 744 073 709 551 615
BIN	RowId -1
STRING	RowId -1
ROWID	-1
CONTEXT	RowId -1

At creation time, all values are initialized to NULL

examples assuming that session Id is 0

since function calls are allways done in the scope of a table (except for global methods and properties) , you will have to use a syntax like :

TABLE:table_name.function ( )
or to access a property :
TABLE:table_name.GetPropertyName ()
TABLE:table_name. SetPropertyName ()

example : Table:EuMaster.Get**Instances** ( );

TABLE Properties

Property name		**R, W or RW**	
Instances	**Global** property Returns all names of instanciated tables on the server	R	0 table.getInstances ( )
Lines	The number of lines in a given table	R	0 table:wikimaster2.getLines( ) 0 1 1 1 1 0 Lines <7 2526415/> ;
Structure	Retrieves the structure of a given table. Returns every column_name, column_type, column_size (only needed for the char type of column)	R	0 tbl:wikimaster2.getStructure ( ) 0 1 12 3 8 0 Name 8 0 Type 1 0 Size <5 title/> <4 CHAR/> <3 200/> <11 link_wikifr/> <4 CHAR/> <3 250/> <12 link_wikieng/> <4 CHAR/> <3 250/> <8 date_maj/> <10 SIMPLEDATE/> <1 0/> <8 date_enr/> <10 SIMPLEDATE/> <1 0/> <5 texte/> <6 STRING/> <1 0/> <4 html/> <6 STRING/> <1 0/> <7 summary/> <6 STRING/> <1 0/> <12 KNW_ABSTRACT/> <8 ABSTRACT/> <1 0/> <11 KNW_LANGAGE/> <5 INT32/> <1 0/> <11 KNW_MEANING/> <5 INT32/> <1 0/> <13 KNW_SEMORIGIN/> <6 UINT32/> <1 0/> ;
Bindexes	Returns all active Btree indexes created on a given table indexed_column, status, *(ready or not)* progress, *(0 - 100%)* unique, *(unique values only, or not)*	R	0 tbl:wikimaster2.getBindexes ( ) 0 1 1 4 8 0 Column 8 0 Status 8 0 Progress 11 0 Unique <23 btree_wikimaster2_title/> <5 ready/> <3 100/> <5 false/> ; one Btree is active on the column **title** of table **wikimaster2,** it is ready, and is not by unique values
Kindexes	Returns all active Ktree indexes created on a given table	R	0 tbl:wikimaster2.getKindexes( ) 0 1 2 3 8 0 Column 8 0 Status 8 0 Progress <23 ktree_wikimaster2_title/> <5 ready/> <3 100/> <23 ktree_wikimaster2_texte/> <5 ready/> <3 100/> ; 2 Ktree has been defined on columns **title** and **texte** of the table wikimaster2 each time a line is inserted or updated, a new learning and indexation process will occur on the line involded, for its **title** and **texte** data

TABLE **Methods**

Method name	params	return	comment
Create	**global** method creates a new table		see below *creating a new table*
Kill	**global** method <table_name>	none	Kills a table
Delete	RowId 1, ,,,,,, RowId n	none	Deletes one or several lines in the table. 0 TBL:wikimaster2.delete (<2 23/>, <1 8/>) deletes lines 23 and 8 of the table
DataAdd	<RowId>, <column name>, <val>	none	Adds data to a variable length field, or to a char field, until the max size of it is reached. The field must not be NULL
Update	RowId , Column_1, val_1, …, Column_n, val_n	none	Updates the values of one or several fields
Select	<mode>, <column>, <operation>, <op1>, [<op2>]	The result Count of the resulting result set	see below *selecting data from a table*
Insert	Column_1, val_1, …, Column_n, val_n	**RowId.** Returns the RowId of the newly created line	Inserts a new line in a table
ReadBlock	<RowId>l, < Column>, [<position>], [< size]>	<TotalSize>, <ReadCount>, <NextPosition>, <Data>	Streaming a variable size field. Return the total size of the field, the size of the read data (in byte) the next position to read (0 if finished) the data read from the column see below streaming*data from a variable size column.*

Method name	**params**	**return**	**comment**
ReadLine	<RowId>, <column_name 1>, ……. <column_name n>,	Values of the columns	0 tbl:wikimaster2.readline (<3 203/>, <5 title/>, <5 rowid/>, <11 knw_meaning/>,) 0 1 1 3 5 201 title 1 0 RowId 1 4 KNW_MEANING <15 XM2001 Crusader/> <3 203/> <1 0/> ; in a master table you can use, **RowId knw_langage** and **knw_meaning** as column names for variable size columns, only the first 256 chars are retrieved by this function in order to get the full content of such a column, use **ReadBlock**
ReadFirstLine	<column_name 1>, ……. <column_name n>,	Values of the columns	Reads the first valid line, which RowId becomes the **current Id** stored in the **Results :FetchId** property 0 tbl:wikimaster2.Readfirstline (<5 title/>, <5 rowid/>,) 0 1 1 2 5 201 title 1 0 RowId <1 X/> <1 0/> ;
ReadNextLine	<column_name 1>, ……. <column_name n>,	Values of the columns	Reads the next line afetr the **current Id** 0 tbl:wikimaster2.ReadNextLine (<5 title/>, <5 rowid/>,) 0 1 1 2 5 201 title 1 0 RowId <4 X!NK/> <1 1/> ; 0 tbl:wikimaster2.ReadNextLine (<5 title/>, <5 rowid/>,) 0 1 1 2 5 201 title 1 0 RowId <3 X&Y/> <1 2/> ;
BIndexCreate	<column>, [<unique>]	none	Creates a Btree over a given column. Optionally with an attribute unique (caution maybe bugged in this version)
BIndexDelete	<column>	none	Kills a Btree existing on a given column
BIndexRebuild	<column>	none	Rebuilds fro scratch a Btree on a given column
KIndexCreate	<column>	none	Creates a Ktreee index on a given column (generally containing textual data)
KIndexDelete	<column>	none	
KIndexRebuild	<column>	none	

Creating a new table.

Create (

String *name*,	the name of the table
objet *owner*,	the owner, NULL or the name of KNW for a master table
string *location*,	if NULL in the server's repository, else a file path
int *previsional_size*,	NULL, or possible number of lines (for optimization)
string *type*,	<simple>, or <master>, default is <simple>
string *structure*	a descriptor of the columns : name, type, size ;

);

creating a master table **wikimaster2**, int the current repository, with a 1 000 000 lines initial capacity.

```
Table.Create (
      "wikimaster2",
      NULL,
      1000000,
      "master",
      " title CHAR 200, link_wikifr CHAR 250, link_wikieng CHAR 250,
      date_maj SIMPLEDATE , date_enr SIMPLEDATE , texte STRING , html
      STRING , summary STRING"
);
```

```
name =             "wikimaster2"     <11 wikimaster2/>
location =         NULL              NULL or <0 />
prev_size =        100000            <6 100000/>
type =             "master"          <6 master/>

structure =        "
                   title             char 200,
                   link_wikifr       char 250,
                   link_wikieng      char 250,
                   date_maj          simpledate,
                   date_enr          simpledate,
                   texte             string,
                   html              string,
                   summary           string
                   "
```

the command send to the server will be :

```
0 TABLE.CREATE(<11 wikimaster2/>, <0 />, <6 100000/>, <6 MASTER/>, <149 title CHAR 200,
link_wikifr CHAR 250, link_wikieng CHAR 250, date_maj SIMPLEDATE , date_enr SIMPLEDATE ,
texte STRING , html STRING , summary STRING />)
```

NB : types are writen in caps for comprehension purpose, but remember, types specification are not case sensitive, you could have used **string** instead of **STRING.**

Selecting data from a table.

Use the Select method in order to select lines of a table according to certain criterion.

Select is a classical procedural method, **not** a mARC based method.

Select uses a column name in order to test which line will be selected.

The colum **must be indexed** via a Bindex (classical Btree index)

The selected lines RowId are stored in a Result Set Object (RS) on the RS stack of the RESULTS object. (notice that the max number of lines in a RS is limited by default to 100 000).

depending of the <mode> parameter,

it will create a new RS, <mode> = <3 new/>

add it's result inside the RS on top of the RESULTS stack, <mode> = <3 add/>

or clear the top RS of the stack before filling it , <mode> = <5 clear/>

Example : assuming session Id = 4

```
4 TABLE:wikimaster2.GetBindexes ( );   / gets the Btree existing on the table wikimaster2
```

```
4 1 1 4 8 0 Column 8 0 Status 8 0 Progress 11 0 Unique <23 btree_wikimaster2_title/> <5
ready/> <3 100/> <5 false/>  ;
```

there is only one, defined over the column **title** of the table **wikimaster2.**

One can only make a select upon that column

*1/ let's find all lines of the table whose title is between « **a** » and « **b** »*

```
4 TABLE:wikimaster2.SELECT(<3 new/>, <5 title/>, <7 Between/>, <1 a/>, <1 b/>)
```

```
4 1 1 2 1 0 ResultCount 1 0 NextPos <6 100000/> <6 100000/>  ;
```

100 000 (the max size of a RS) has been selected from the table

*2/ let's find all lines of the table whose title is between « **a** » and « **ab** »*

```
4 TABLE:wikimaster2.SELECT(<3 new/>, <5 title/>, <7 Between/>, <1 a/>, <2 ab/>)

4 1 1 2  1 0 ResultCount       1 0 NextPos
         <4 6725/>             <1 0/>  ;
```

3/ let's find all lines of the table whose title begins with « z »

```
4 TABLE:wikimaster2.SELECT(<3 new/>, <5 title/>, <2 >=/>, <1 z/>)

4 1 1 2  1 0 ResultCount       1 0 NextPos
         <4 9507/>              <1 0/>  ;
```

A new RS with 9507 rowIds has been created on top of the RESULTS object stack

you can use the comparison operators in the table below in order to perform a select operation

Opérator	alias	code	# parameters	description
>	GT	0	1	Greater than op1
<	LT	1	1	Lower than op1
>=	GTE	2	1	Greater or Equal to op1
<=	LTE	3	1	Lower or Equal à op1
Between	BT	4	2	Between [op1, op2] (including op1 and op2)
=	EQ	5	1	Equal to op1

Streaming data from a variable size column.

Assuming session Id is 3

3 table:wikimaster2.get (<9 structure/>,)

```
3 1 12 3      8 0 Name              8 0 Type              1 0 Size
              <5 title/>            <4 CHAR/>             <3 200/>
              <11 link_wikifr/>     <4 CHAR/>             <3 250/>
              <12 link_wikieng/>    <4 CHAR/>             <3 250/>
              <8 date_maj/>         <10 SIMPLEDATE/>      <1 0/>
              <8 date_enr/>         <10 SIMPLEDATE/>      <1 0/>
              <5 texte/>            <6 STRING/>           <1 0/>
              <4 html/>             <6 STRING/>           <1 0/>
              <7 summary/>          <6 STRING/>           <1 0/>
              <12 KNW_ABSTRACT/>    <8 ABSTRACT/>         <1 0/>
              <11 KNW_LANGAGE/>     <5 INT32/> <1 0/>     <11 KNW_MEANING/>
              <5 INT32/> <1 0/>     <13 KNW_SEMORIGIN/>   <6 UINT32/> <1 0/>  ;
```

the table wikimaster2 has 12 columns.

The column **title** is of type **string**, eg is a **variable size** column.
In the following examples, we shall use this column.

***1/** reading 4096 bytes (by default) the column title of the line 819 of the table wikimaster2, from the beginning (byte #1 by default)*

/ uses default position and size to read, 1, 4096

```
3 table:wikimaster2.readblock ( <3 819/>, <5 texte/>,  )



3 1 1 4        2 0 TotalSize  2 0 ReadCount  2 0 NextPosition      8 0 Data
               <2 58/>        <2 58/>        <1 0/>                <58 This is a list of the
903 Top 20 number-one hits of 2007.
/>  ;
```

field is 58 bytes long, 58 bytes have been read, next position to read is 0, eg all the field has been read.

***2/** streaming the column title of the line 1471705 of the table wikimaster2, from the beginning, by 256 bytes blocks.*

/reads the firs 256 bytes bloc, starting at position #1

```
      3 table:wikimaster2.readblock ( <7 1471705/>, <5 texte/>, <1 1/>, <3 256/>,  )

      3 1 1 4  2 0 TotalSize  2 0 ReadCount  2 0 NextPosition      8 0 Data
               <4 4307/>      <3 256/>       <3 257/>              <256
      Allmusic  link
      Alternative Press  [1]
      The A.V. Club (favorable) 2005
      Robert Christgau  link
      Entertainment Weekly (A-) 2005
      Filter (93/100) [2]
      Pitchfork Media (7.6/10) 2005
      Mojo  [3]
      PopMatters  2005
      Rolling Stone  2005
      Yahoo! Music  2/>  ;
```

the field is **4307** bytes long, the next position to read is **257**.

```
      3 1 1 4  2 0 TotalSize  2 0 ReadCount  2 0 NextPosition      8 0 Data
               <4 4307/>      <3 256/>       <3 513/>              <256 005
      "Off the Record"
      Released: 2005
      Z is the fourth studio album by rock band My Morning Jacket. This collection features a
      much spacier and polished sound than previous MMJ releases, making heavy use of
      synthesizers throughout and incorporating small b/>  ;
```

next block has been read, the next position to read is **513**

reading next block.....

reading last block

```
3 table:wikimaster2.readblock ( <7 1471705/>, <5 texte/>, <4 4097/>, <3 256/>,  )

3 1 1 4         2 0 TotalSize  2 0 ReadCount  2 0 NextPosition      8 0 Data
                <4 4307/>      <3 211/>       <1 0/>                <211 "The Top 200
Albums of the 2000s: 200-151". Pitchfork Media. http://pitchfork.com/features/staff-
lists/7707-the-top-200-albums-of-the-2000s-150-101/. Retrieved October 1, 2009.
External links
Z at Metacritic
/>  ;
```

211 bytes has been read for the last block, and the next position to read is **0**, eg the streaming is finished

TABLE.GetInstances

description

Gets the name of all the database tables instanciated in the server

prototype

```
Table.GetInstances (int32 start, int32 count)
```

defaults

```
start = 1   (the first table)
count = -1 (all until the end is reached)
```

error messages

```
1 :    "start parameter out of range must be >= 1 ";
2 :    "count parameter anomaly";
3 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
Table.GetInstances ();
```

Tables
EuDetail
EuMaster

```
Table.GetInstances (2,1);
```

Tables
EuMaster

parameters **start** and **count** are usefull when the number of tables is high.

it is then possible to get all the connections by fetching them 10 by 10 in order to optimize the network and server load.

start is in base 1 eg, the first session in the list is at index 1

if the returned number of lines is less than the value of parameter **count**, all table names will have been fetched.

TABLE:xxx.GetLines

description
Retrieves the total number of lines in a database table

prototype
Table:*tablename*.GetLines (void)

or

Tbl:*tablename*.GetLines (void)

defaults
void

error messages

```
1 :    "unknown or deleted table";
2 :    "unable access table";
3 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
table:eudetail.GetLines (  )
```

lines
33238

```
table:eumaster.GetLines (  )
```

lines
33238

```
table:eudetail.delete ( "33237", " 33236", "33235 " ) ;
```

deleted
3

```
table:eumaster.delete ( " 33237"  ) ;
```

deleted
1

```
table:eudetail.GetLines (  )
```

lines
33235

```
table:eumaster.GetLines (  )
```

lines
33237

TABLE:xxx.GetStructure

description

Gets the structure of a given table.

Once created, the structure of a table cannot be altered

prototype

```
Table:tablename.GetStructure (void)
```

defaults

void

error messages

```
1 :    "unknown or deleted table";
2 :    "parameter start out of range";
3 :    "parameter count out of range";
4 :    "unable to acces table";
x :    "unknown error";
```

examples

```
table:eumaster.GetStructure ( );
```

name	type	size
id	INT32	0
titre	CHAR	106
soustitre	CHAR	141
rtf	STRING	0
texte	STRING	0
KNW_ABSTRACT	ABSTRACT	0
KNW_LANGAGE	INT32	0
KNW_MEANING	INT32	0

TABLE:xxx.GetBIndexes

description
Gets the names and the status of all Btree indexes linked to a given database table.

prototype
```
table:tablename.GetBindexes (void )
```

defaults
```
void
```

error messages
```
1 :    "unknown or deleted table";
2 :    "parameter start out of range";
3 :    "parameter count out of range";
4 :    "unable to acces table";
x :    "unknown error";
```

examples

```
table:eumaster.GetBindexes ( );
```

column	status	progress	unique
btree_eumaster_id	ready	100	false

TABLE:xxx.GetKIndexes

description

Gets the names and the status of all Ktree indexes linked to a given database table.

prototype

```
table:tablename.GetKindexes (void)
```

defaults

```
void
```

error messages

```
1 :    "unknown or deleted table";
2 :    "parameter start out of range";
3 :    "parameter count out of range";
4 :    "unable to access table";
x :    "unknown error";
```

examples

```
table:eumaster.GetKindexes ( );
```

column	status	progress
ktree_EuMaster_titre	ready	100
ktree_EuMaster_texte	ready	100

TABLE.Create

description

prototype

```
Table.Create (
```

string *name*,	the name of the table
string *location*,	if NULL in the server's repository, else a file path
int32 *previsional_size*,	NULL, or possible number of lines (for optimization)
string *type*,	<simple>, or <master>, default is <simple>
string *structure*	a descriptor of the columns : name, type, size ;

```
)
```

defaults

```
void
```

error messages

```
1 :    "system error unable to read parameters";
3 :    "illegal parameter name";
4 :    "illegal table descriptor";
5 :    "table allready exists";
6 :    "illegal name";
7 :    "unknown owner object";
9 :    "master table without mARC link";
10 :   "master table allready exists";
12 :   "table object registration failure";
14 :   "unable to create table structure";
15 :   "unable to create extra master table columns";
16 :   "unable to create composite data manager";
17 :   "unable to create variable size data managers";
18 :   "unable to update .cfg configuration file";
19 :   "unable to link master table knw_abstract column to the mARC";
x :    "unknown error";
```

examples

creating a master table **wikimaster2**, int the current repository, with a 1 000 000 lines initial capacity.

```
Table.Create (
      "wikimaster2",
      NULL,
      1000000,
      "master",
      " title CHAR 200, link_wikifr CHAR 250, link_wikieng CHAR 250,
      date_maj SIMPLEDATE , date_enr SIMPLEDATE , texte STRING , html
      STRING , summary STRING"
);
```

```
name =              "wikimaster2"      <11 wikimaster2/>
location =          NULL               NULL or <0 />
prev_size =         100000             <6 100000/>
```

```
type =            "master"          <6 master/>

structure =       "
                  title             char 200,
                  link_wikifr       char 250,
                  link_wikieng      char 250,
                  date_maj          simpledate,
                  date_enr          simpledate,
                  texte             string,
                  html              string,
                  summary           string
                  "
```

the command send to the server will be :

```
0 TABLE.CREATE(<11 wikimaster2/>, <0 />, <6 100000/>, <6 MASTER/>, <149 title CHAR 200,
link_wikifr CHAR 250, link_wikieng CHAR 250, date_maj SIMPLEDATE , date_enr SIMPLEDATE ,
texte STRING , html STRING , summary STRING />)
```

NB : types are writen in caps for comprehension purpose, but remember, types specification are not case sensitive, you could have used **string** instead of **STRING.**

TABLE.Kill

description
kills a table

prototype
```
Table.Kill (string tablename)
```

defaults
```
void
```

error messages
```
2 :    "unknown or allready deleted table";
3 :    "unable to delete name from repository";
4 :    "unable to delete section from cfg file";
5 :    "delete failure. may be bad extension";
6 :    "unable to kill associated indexes";
7 :    "unable to kill table instance";
8 :    "table allready used by another process";
x :    "unknown";
```

examples

TABLE:xxx.Insert

description

inserts a new line in a database table

prototype

```
Table.tablename.Insert (string colname1, string value1….string colnameN,
string valueN)
```

the rowid of the new line is returned

defaults

void

error messages

```
1 :    "invalid number of parameters";
2 :    "table lock";
3 :    "releasing memory string";
4 :    "session unlock";
5 :    "serialization";
6 :    "unknown insert";
7 :    "unknown column";
8 :    "invalid index";
9 :    "value already exists";
10 :   "insert failure";
11 :   "unable to find KNW_SEMORIGIN";
12 :   "btree indexation failure";
13 :   "maximum capacity reached";
x :    "unknown error";
```

examples

```
table:EuMaster.Insert (
                           "titre", "un titre",
                           "texte", "un texte",
                           "soustitre", NULL
                           );
```

rowid
33238

TABLE:xxx.Update

description

updates (modifies) one or several columns of a preexisting line in a table

prototype

```
Table.tablename.Update (
                                Int32 rowid,
                                string colname1, string value1,
                                …,
                                string colnameN, string valueN
                                )
```

defaults

```
void
```

error messages

```
1 :    "lock timeout";
2 :    "missing parameters";
3 :    "update failure";
4 :    "unable to recover parameters ressources";
5 :    "unknown column";
6 :    "value not compatible with the column type";
7 :    "value out of range";
8 :    "unlock failure";
9 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
table:EuMaster.Update (
                        33238,
                        "titre", "un nouveau titre",
                        "texte", "un nouveau texte"
                        );
```

rowid
33238

TABLE:xxx.Delete

description

deletes one or several lines of a given table of the database

no assumption must be made about the deleted Id's, since they can be reused during a next insert operation.

prototype

Table:tablename.Delete (rowid1,rowid2….,rowidN)

defaults

void

error messages

```
1 :    "missing parameters";
2 :    "lock failure. one or several lines are in use";
3 :    "unable to delete one or several lines";
4 :    "unlock failure";
5 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
table:eudetail.GetLines (  )
```

lines
33238

```
table:eumaster.GetLines (  )
```

lines
33238

```
table:eudetail.delete ( "33237", " 33236", "33235 " ) ;
```

deleted
3

```
table:eumaster.delete ( " 33237"  ) ;
```

deleted
1

```
table:eudetail.GetLines (  )
```

lines
33235

```
table:eumaster.GetLines (  )
```

lines
33237

TABLE:xxx.DataAdd

description

Adds data to a field of a line of a database Table.

this method is generally used to store large binary or text data block by block into a variable length field, instead of a one time commit that could overload the network, and slow multiple access

DataAdd can only operate on a one single column of a line

prototype

```
Table:tablename.DataAdd (int32 rowid, string colname, string value)


returns NULL
```

defaults

```
void
```

error messages

```
1 :    "unknown table";
2 :    "empty data string";
3 :    "value not compatible with the column type";
4 :    "lock timeout";
5 :    "DataAdd failure";
6 :    "unlock failure";
7 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
table:EuMaster.DataAdd (33238,"titre", " la suite du titre");
```

TABLE:xxx.Select

description

Use the Select method in order to select lines of a table according to certain criterion.

Select is a classical procedural method, **not** a mARC based method.

Select uses a column name in order to test which line will be selected.

The colum **must be indexed** via a Bindex (classical Btree index), or you can use the RowId column.

The selected lines RowId are stored in a Result Set Object (RS) on the RS stack of the RESULTS object. (notice that the max number of lines in a RS is limited by default to 100 000).

depending of the <mode> parameter,

it will create a new RS, <mode> = <3 new/>

add it's result inside the RS on top of the RESULTS stack, <mode> = <3 add/>

or clear the top RS of the stack before filling it , <mode> = <5 clear/>

you can use the comparison operators in the table below in order to perform a select operation

Operator	alias	code	# parameters	description
>	GT	0	1	Greater than op1
<	LT	1	1	Lower than op1
>=	GTE	2	1	Greater or Equal to op1
<=	LTE	3	1	Lower or Equal à op1
Between	BT	4	2	Between [op1, op2] (including op1 and op2)
=	EQ	5	1	Equal to op1

prototype

```
Table:tablename.Select (
                              string mode,
                              string colname,
                              string comparison,
                              string operand1,
                              string operand2
                              )
mode : [new, add, clear]
comparison : [ >, <, >=, <=, =, Between]
```

defaults

```
void
```

error messages

```
1 :   "unable to get paramater string";
2 :   "illegal comparison operator must be [ >, <, >=, <=, =, between]";
3 :   "bad or unknown column name";
4 :   "illegal mode parameter must be [ new, add, clear]";
5 :   "RS linked to a different table for parameter ADD";
6 :   "select failure";
```

```
7 :    "bad or unknown column name";
8 :    "no Btree defined for this column. column RowId does not need to be indexed";
9 :    "select failure";
10 :   "unable to update or create Result Set on the Results Stack";
11 :   "unable to access Result Set ";
12 :   "unable to generate response string";
x :    "unknown";
```

examples

Example : assuming session Id = 4

```
4 TABLE:wikimaster2.GetBIndexes( );   / gets the Btree existing on the table wikimaster2

4 1 1 4 8 0 Column 8 0 Status 8 0 Progress 11 0 Unique <23 btree_wikimaster2_title/> <5
ready/> <3 100/> <5 false/>  ;
```

there is only one, defined over the column **title** of the table **wikimaster2.**

One can only make a select upon that column

*1/ let's find all lines of the table whose title is between « **a** » and « **b** »*

```
4 TABLE:wikimaster2.SELECT(<3 new/>, <5 title/>, <7 Between/>, <1 a/>, <1 b/>)

4 1 1 2 1 0 ResultCount 1 0 NextPos <6 100000/> <6 100000/>  ;
```

100 000 (the max size of a RS) has been selected from the table

*2/ let's find all lines of the table whose title is between « **a** » and « **ab** »*

```
      4 TABLE:wikimaster2.SELECT(<3 new/>, <5 title/>, <7 Between/>, <1 a/>, <2 ab/>)

      4 1 1 2  1 0 ResultCount       1 0 NextPos
               <4 6725/>             <1 0/>  ;
```

3/ let's find all lines of the table whose title begins with « z »

```
      4 TABLE:wikimaster2.SELECT(<3 new/>, <5 title/>, <2 >=/>, <1 z/>)

      4 1 1 2  1 0 ResultCount       1 0 NextPos
               <4 9507/>              <1 0/>  ;
```

A new RS with 9507 rowIds has been created on top of the RESULTS object stack

prototype

```
Table:tablename.Select (
                              string mode,
                              string colname,
                              string operator,
                              string operand1,
                              string operand2
                        )
```

mode can be :

`"new"` :	a new result set is created on the result's stack
`"add"`	selection is added to the RS on top of result's stack
`"clear"`	top RS of result's stack is cleared, the selection is added it

defaults

void

error messages

example1

```
clear ( " results") ;
results.setProperties ("format = rowid);
table:eumaster.Select  ( "new", "rowid", "between", "31", "40") ;
table:eumaster.Select  ( "new", "rowid", "between", "21", "30") ;
table:eumaster.Select  ( "new", "rowid", "between", "11", "20") ;
table:eumaster.Select  ( "new", "rowid", "between", "1", "10") ;
```

```
results.fetch ( "10", "1" )
```

RowId
1
2
3
4
5
6
7
8
9
10

stack
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40

TABLE:xxx.ReadLine

description

Reads the content of one line of a database table.

for a variable length, only the 254 first byte are read. In this case, you must use the *ReadBlock* method in order to acces to all data of a variable length field

prototype

Table:tablename.ReadLine (int32 rowid, string colname1…., sting colnameN);

defaults

```
void
```

error messages

```
2 :    "missing parameter table line Id";
3 :    "not a valid, or deleted line";
4 :    "line read locked. allready in use";
5 :    "line write locked, allready in use";
6 :    "line locked";
7 :    "unable to lock line";
8 :    "lock timeout for the line";
9 :    "missing Id extraction method";
10 :   "error parsing data format";
11 :   "unable to read line";
12 :   "unable to generate response string";
```

examples

table:eumaster.readline ("2", " Rowid"," titre", "soustitre")

RowId	titre	soustitre
2	AÉROSPATIALE (INDUSTRIE)	2. Caractères généraux

TABLE:xxx.ReadFirstLine

description

Reads the first valid line, which RowId becomes the **current Id** stored in the Results.FetchID property.

since lines could have been previously deleted, valid rowid's are not guaranted being adjacent.

using *ReadFirstLine*, *ReadNextLine* methods can be used to build an iterator through all valid lines in a table.

The rowid of the last accessed valid line can be retrieved by using *session.GetLastDBInfo*

prototype

```
Table:tablename.ReadFirstLine (string colname1…., sting colnameN);
```

defaults

```
void
```

error messages

```
2 :    "missing parameter table line Id";
3 :    "not a valid, or deleted line";
4 :    "line read locked. allready in use";
5 :    "line write locked, allready in use";
6 :    "line locked";
7 :    "unable to lock line";
8 :    "lock timeout for the line";
9 :    "missing Id extraction method";
10 :   "error parsing data format";
11 :   "unable to read line";
12 :   "unable to generate response string";
```

examples

```
table:eumaster.ReadFirstLine ("rowid","titre","soustitre");
```

RowId	titre	soustitre
0	GABON	

```
GetLastDBInfo();
```

table	operation	id	status
EuMaster	read_line	0	ok

```
table:eumaster.ReadNextLine ("rowid","titre","soustitre");
```

RowId	titre	soustitre
1	BAYIN NAUNG	

```
table:eumaster.ReadNextLine ("rowid","titre","soustitre");
```

RowId	titre	soustitre
2	AÉROSPATIALE (INDUSTRIE)	2. Caractères généraux

```
table:eumaster.ReadNextLine ("rowid","titre","soustitre");
```

RowId	titre	soustitre
3	VICAIRE (G.)	

```
GetLastDBInfo();
```

table	operation	id	status
EuMaster	read_line	3	ok

TABLE:xxx.ReadNextLine

description

Reads the next valid rowid after a *ReadFirstLine* command, or the first valid rowid after session.DBCursor property.

setting the session.DBCursor is another way of iterating from another origin than the first valid line.

prototype

```
Table:tablename.ReadNextLine (string colname1…., sting colnameN);
```

defaults

```
void
```

error messages

```
2 :   "missing parameter table line Id";
3 :   "not a valid, or deleted line";
4 :   "line read locked. allready in use";
5 :   "line write locked, allready in use";
6 :   "line locked";
7 :   "unable to lock line";
8 :   "lock timeout for the line";
9 :   "missing Id extraction method";
10 :  "error parsing data format";
11 :  "unable to read line";
12 :  "unable to generate response string";
```

examples

```
table:eumaster.ReadFirstLine ("rowid","titre","soustitre");
```

RowId	titre	soustitre
0	GABON	

```
GetLastDBInfo();
```

table	operation	id	status
EuMaster	read_line	0	ok

```
table:eumaster.ReadNextLine ("rowid","titre","soustitre");
```

RowId	titre	soustitre
1	BAYIN NAUNG	

```
table:eumaster.ReadNextLine ("rowid","titre","soustitre");
```

RowId	titre	soustitre
2	AÉROSPATIALE (INDUSTRIE)	2. Caractères généraux

```
table:eumaster.ReadNextLine ("rowid","titre","soustitre");
```

RowId	titre	soustitre
3	VICAIRE (G.)	

```
GetLastDBInfo();
```

table	operation	id	status
EuMaster	read_line	3	ok

TABLE:xxx.ReadBlock

description

Stream reading of a variable size column of a line of a given table.

global and direct access to more than 254 byte data (through *ReadLine*) is not allowed for a variable size field.

prototype

```
Table:tablename.ReadBlock (
                                        Int32 rowid,
                                        string colname,
                                        int32 start,
                                        int32 count
                                        )
```

the method returns the *totalsize* of the field, the number of bytes read, *ReadCount*, and the *NextPosition* to read from.

Typically, an iterator will read blocks as long as Nextposition is different from 0

defaults

```
start = 1          (start is in base 1)
count = 4096       (block size in byte)
```

error messages

```
2 :    "missing parameter table line Id";
3 :    "not a valid, or deleted line";
4 :    "line read locked. allready in use";
5 :    "line write locked, allready in use";
6 :    "line locked";
7 :    "unable to lock line";
8 :    "lock timeout for the line";
9 :    "missing Id extraction method";
10 :   "error parsing data format";
11 :   "unable to find column";
12 :   "not a variable type column";
13 :   "unable to acquire block Id";
14 :   "empty or NULL data";
15 :   "unable link with variable file";
16 :   "internal data block read error";
17 :   "unable to generate response string";
x :    "unknown error";
```

examples

table:eumaster.ReadBlock (3, " texte") ;

TotalSize	ReadCount	NextPosition	Data
1011	1011	0	VICAIRE (G.) VICAIRE GABRIEL (1848-1900) Poète français, Gabriel Vicaire se met assez tardivement à écrire. En plein symbolisme, il s'oppose à toutes les recherches poétiques à la mode et fait figure d'homme heureux et sensuel auprès des décadents tourmentés par le sens de leur mission. Vicaire se rattache ainsi à une tradition de poésie facile, où se chante un accord profond avec la nature, avec la nourriture, le vin et l'amour. Originaire de la Bresse, ce sont les richesses de sa province qui l'inspirent dans Émaux bressans

			(1884), L'Heure enchantée (1890), Au bois joli (1893), Le Clos des fées (1897). Parfois sous la joie de vivre perce une mélancolie ou une amertume qui donnent à ses vers un ton de sincérité profonde. Mais, plus que par sa poésie, Vicaire est connu pour un pastiche de la poésie décadente. Sous le pseudonyme d'Adoré Floupette, il publie en 1885 Les Déliquescences , où il se livre à une parodie pleine d'humour des thèmes décadents.

table:eumaster.ReadBlock (10,"texte",1,256);

TotalSize	ReadCount	NextPosition	Data
3509	256	257	BABINSKI (J.) BABINSKI JOSEPH (1857-1932) Médecin français né à Paris de parents émigrés polonais, connu surtout pour ses travaux sur le système nerveux. Après des études secondaires à l'école polonaise des Batignolles à Paris, Babinski est interne en mé

table:eumaster.ReadBlock (10,"texte",257,256);

TotalSize	ReadCount	NextPosition	Data
3509	256	513	decine dans le service de Cornil à l'Hôtel-Dieu (1879), puis chef de clinique de Charcot à la Salpêtrière (1884). Il soutient en 1885 sa thèse de doctorat sur La Sclérose en plaques . Médecin des hôpitaux en 1890, sa carrière se déroule à la Pitié, dont il

…..

table:eumaster.ReadBlock (10,"texte",3400,256);

TotalSize	ReadCount	NextPosition	Data
3509	110	0	udie les séquelles nerveuses de l'épidémie d'encéphalite léthargique.

NextPosition = 0, all data have been retrieved.

TABLE:xxx.BIndexCreate

description

Creates a BTree index on the column of a table.

once created, such an index gives direct and fast access to the lines of a database table.

the column can be used with the TABLE:xxx.Select method in order to creat a ResultSet

Moreover, if the BIndex is created with the "unique" tag, a duplicated document will not be inserted in the table, and an error is issued.

Only one Bindex can be cretad for a column.

the name of the btree index will be :

btree_tablename_colname

Once created, a btree index will automatically begin the rebuild process

prototype

```
Table:tablename.BIndexCreate  (string colname, bool unique);
```

returns NULL;

defaults

```
unique = false;
```

error messages

```
2 :    "missing parameter column name";
3 :    "table lock timeout. table allready in use";
4 :    "unable to create Bindex instance";
5 :    "unknown column";
6 :    "variable size column type. no possible x";
7 :    "index allready exists";
8 :    "unable to create physical index";
9 :    "unable to synchronize table and index";
11 :   "index build stopped by user";
12 :   "index build stopped. if tag unique is used, data in the table is not unique ";
13 :   "index build failure";
14 :   "unable to generate response string";
x :    "unknown error";
```

examples

```
Table:eumaster.BIndexCreate ("Id", true);
```

```
table:eumaster.GetBIndexes (  );
```

column	status	progress	unique
btree_eumaster_id	ready	100	true

creates a BTree on the column "Id", so that these values are unique

TABLE:xxx.BIndexDelete

description
Deletes a BIndex

prototype
```
Table:tablename.BIndexDelete (string indexName)
```

defaults
```
void
```

error messages
```
2 :    "missing parameter index name";
3 :    "table lock timeout. table allready in use";
4 :    "unable to delete Bindex";
5 :    "unknown index";
6 :    "unknown column";
7 :    "index locked, allready in use";
8 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
table:eumaster.GetBIndexes (  );
```

column	status	progress	unique
btree_eumaster_id	ready	100	true

table:eumaster.BindexDelete ("btree_eumaster_id");

```
table:eumaster.GetBIndexes (  );
```

column	status	progress	unique

TABLE:xxx.BIndexRebuild

description

administration operation.

Rebuilds a given index.

Since the process can be long, issuing the same command, with the interrupt flag set to true, will abort the process.

If aborted, the index is no more valid, and must be deleted and then recreated

Table:tablename.GetBindexes command via another session will display the state and the completion of this task.

prototype

```
Table:tablename.BIndexRebuild (string indexName, bool interrupt)
```

defaults

```
interrupt = false
```

error messages

```
2 :    "missing parameter index name";
3 :    "table lock timeout. table allready in use";
4 :    "unable to rebuild Bindex";
5 :    "unknown column";
6 :    "variable size column type. no possible x";
7 :    "index allready exists";
8 :    "unable to create physical index";
9 :    "unable to synchronize table and index";
11 :   "index build stopped by user";
12 :   "index build stopped. if tag unique is used, data in the table is not unique ";
13 :   "index build failure";
14 :   "unable to generate response string";
x :    "unknown error";
```

examples

```
TABLE:EuMaster.BindexRebuild(" btree_eumaster_id");
```

column	status	progress	unique
btree_eumaster_id	building	37	true

TABLE:xxx.KIndexCreate

description

A KIndex is a special kind of index that can be used ONLY on a text column of the linked master table.

each time, a *table.insert,* or *table.update* occurs on a Kindexed column, the new content is automatically indexed.

this is a very simple way to design a back office application using the internal mARC server indexation engine.

Once created, the KTree will **NOT** begin to rebuild

the name of the ktree index will be :

ktree_tablename_colname

prototype

```
Table:tablename.KIndexCreate  (string colname);
```

colname **must** content text, otherwise, strange contexts will be detected by the mARC

defaults

```
void
```

error messages

```
1 :    "not a master table";
2 :    "missing parameter column name";
3 :    "table lock timeout. table allready in use";
4 :    "unable to create KIndex";
5 :    "unknown column";
6 :    "column type is not of type string or char";
7 :    "index allready exists";
8 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
Table:eumaster.KIndexCreate  ("titre");
Table:eumaster.KIndexCreate  ("texte");


table:eumaster.GetKIndexes (  );
```

column	status	progress
ktree_EuMaster_titre	ready	100
ktree_EuMaster_texte	ready	100

TABLE:xxx.KIndexDelete

description

Deletes a KIndex

prototype

```
Table:tablename.KIndexDelete (string indexName)
```

defaults

```
void
```

error messages

```
1 :    "not a master table";
2 :    "missing parameter column name";
3 :    "table lock timeout. table allready in use";
4 :    "unable to delete KIndex";
5 :    "unknown KIndex";
6 :    "unknown column";
7 :    "index locked, allready in use";
8 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
table:eumaster.GetKIndexes (  );
```

column	status	progress
ktree_EuMaster_titre	ready	100
ktree_EuMaster_texte	ready	100

table:eumaster.KindexDelete ("ktree_eumaster_titre");

```
table:eumaster.GetKIndexes (  );
```

column	status	progress
ktree_EuMaster_texte	ready	100

TABLE:xxx.KIndexRebuild

description

administration operation.

Rebuilds a given index.

Since the process can be long, issuing the same command, with the interrupt flag set to true, will abort the process.

If aborted, the index is no more valid, and must be deleted and then recreated

Table:tablename.GetKindexes command via another session will display the state and the completion of this task.

prototype

```
Table:tablename.KIndexRebuild (string indexName, bool interrupt)
```

defaults

```
interrupt = false
```

error messages

```
1 :    "not a master table";
2 :    "missing parameter column name";
3 :    "table lock timeout. table allready in use";
4 :    "unable to rebuild KIndex";
5 :    "unknown KIndex";
6 :    "unknown column";
7 :    "index locked, allready in use";
8 :    "unable to generate response string";
9 :    "rebuild allready running";
x :    "unknown error";
```

examples

```
TABLE:EuMaster.KindexRebuild(" ktree_eumaster_texte");
```

column	status	progress
ktree_eumaster_texte	building	52

Contexts

Handling the Contexts stack

Examples of the use of the **new**, **drop**, **dup**, **swap**, **rolldown**, **rollup** methods

1 creating 4 new empty contexts and making some stack operations

```
5 contexts.new ( ) ; Context #1
5 contexts.new ( ) ; Context #2
5 contexts.new ( ) ; Context #3
5 contexts.new ( ) ; Context #4
```

```
Context #4
Context #3
Context #2
Context #1
```

```
5 contexts.swap ( ) ;
```

```
Context #3
Context #4
Context #2
Context #1
```

```
5 contexts.dup ( ) ;

Context #5 = Context #3
```

```
Context #5
Context #3
Context #4
Context #2
Context #1
```

```
5 contexts.drop ( ) ;
```

```
Context #3
Context #4
Context #2
Context #1
```

```
5 contexts.rolldown ( <1 3/>)
```

```
Context #2
Context #3
Context #4
Context #1
```

```
5 contexts.rolldown ( <1 4/>)
```

```
Context #1
Context #2
Context #3
Context #4
```

```
5 contexts.rollup ( <1 4/>)
```

```
Context  #2
Context  #3
Context  #4
Context  #1
```

```
5 contexts.rollup ( <1 3/>)
```

```
Context  #3
Context  #4
Context  #2
Context  #1
```

```
5 contexts.swap ( )
```

```
Context  #4
Context  #3
Context  #2
Context  #1
```

```
5 contexts.clear ( )
```

```
Empty stack
```

Combining contexts.

Examples of the use of the **addelement**, **normalize**, **union, intersection** methods

1 creating 3 new empty contexts and making some stack operations

```
5 contexts.new (  ) ;
5 contexts.new (  ) ;
```

```
Empty context
Empty context
```

2 adding some words to each

```
5 contexts.addelement ( <10 metal iron/>)
5 contexts.swap (  ) ;
5 contexts.addelement ( <11 metal music/>,  )
```

```
metal 100 music 100
metal 100 iron 100
```

```
5 contexts.dup ()
5 contexts.rolldown ( <1 3/> )
5 contexts.dup ()
5 contexts.rolldown ( <1 3/> )
```

```
metal 100 music 100
metal 100 iron 100
metal 100 iron 100
metal 100 music 100
```

```
5 contexts.union ()
```

```
metal 100 music 100 metal 100 iron 100
metal 100 iron 100
metal 100 music 100
```

```
5 contexts.normalize()
```

makes the context with unique elements and consolidates activities to a normalized value of 100 %

```
metal 100 music 50 iron 50
metal 100 iron 100
metal 100 music 100
```

```
5 contexts.drop()
```

kills the upper context on the stack

```
metal 100 iron 100
metal 100 music 100
```

```
5 contexts.intersection()
```

intersection of the 2 upper contexts on the stack. Only common elements of both contexts are kept, anf their activities are consolidated.

```
metal 200
```

CONTEXTS.GetProperties

description

Gets one or several Context properties.

To access one property, use a directive as : propertyname
To access several properties values in one command, separate each directive with the character semi column (;)

if there are no parameter, all properties of the topmost context of the context's stack will be accessed

prototype

```
Contexts.GetProperties (string accessor, int32 index);
accessor = "propertyname1; …;propertynameN"
```

defaults

```
accessor = ""
index = 1   (base 1)    the index of the context on the stack
```

error messages

```
1 :    "empty context stack";
2 :    "unable to read parameters";
3 :    "unable to decode parameter string";
4 :    "unable to read properties";
5 :    "unable to generate response string";
x :    "unknown error";
```

examples

```
Contexts.GetProperties ( );
```

prop_name	prop_value	prop_type	prop_access
name	anonymous	string	rw
atom_count	1	int32	rw
activity	-1	int32	r
context_string	# mARC CONTEXT 100 atome 100	string	rw
fetch_size	48	uint32	rw
fetch_start	1	int32	rw

CONTEXTS.SetProperties

description

Access to context's properties

Trying to change the value of a Read Only property will **not** generate an error.

To see which properties are avalaible, see CONTEXTS.GetProperties

To change one property, use a directive as : propertyname = propertyvalue
To change several properties values in one command, separate each directive with the character semi column (;)

properties of the topmost context of the context's stack will be changed

prototype

```
Contexts.SetProperties ([int32 index], string accessor, …string accessorN )
index :     index of th context parameter on the stack. dfault 1
            optional, between  1 – stack_count
```

defaults

```
index = 1
```

error messages

error messages

```
0 :   "empty contexts stack";
1 :   "parameter error";
2 :   "unable to decode parameter string";
3 :   "unable to write properties";
4 :    custom messages depending of the property
5 :   "index out of range must be [1, stack_count]";
x :   "unknown";

custom messages
"error setting property context.XXX"
```

examples

```
contexts.SetProperties ( " context_string =  noyau atome électron " );
Contexts.GetProperties ( );
```

prop_name	prop_value	prop_type	prop_access
atom_count	3	int32	rw
activity	-1	int32	r
context_string	# mARC CONTEXT 300 noyau 100 atome 100 électron 100	string	rw

```
contexts.setproperties ( "atom_count = 2");
contexts.getproperties ( " atom_count ; context_string")
```

prop_name	prop_value	prop_type	prop_access
atom_count	2	int32	rw
context_string	# mARC CONTEXT 200 noyau 100 atome 100	string	rw

Warning

in the preceding example

these syntaxes are correct

```
1 contexts.SetProperties ( " context_string =  noyau atome électron " );
2 contexts.SetProperties ( " context_string =  "noyau atome électron" " );
```

```
but when using an extended accessor to set 2 or more properties, you will
have to use syntax #2
```

```
eg :
contexts.SetProperties (
      "context_string =  "noyau atome electron" ; atom_count = 2"
       );

contexts.SetProperties (
      "context_string =  noyau atome electron ; atom_count = 2"
       );
```

will fail

on the other hand

```
contexts.SetProperties (
      "context_string =  noyau atome electron, "atom_count = 2"
       );

contexts.SetProperties (
      "context_string =  "noyau atome electron", "atom_count = 2"
       );
```

will both run

For a property of type string, it is a good idea to put the data into quotes in the accessor.
but, if the data contents itself QUOTES character, you will first have to escape them before quoting it
example :

the data is :

```
this is an "example" of ambiguous string
```

you must transform it into

```
this is an \"example\" of ambiguous string
```

then you can build your accessor, and the data to send will be

```
context_string =  "this is an \"example\" of ambiguous string"
```

but for comfort, and to minimize boring string handling at client side, you can avoid that, by using multi parameters single accessors for the method.

the data is :

```
this is an "example" of ambiguous string
```

the accessor string will be

```
context_string = this is an "example" of ambiguous string
```

CONTEXTS.New

description
Creates a new context object on top of the Context's stack.

prototype
```
contexts.new(void)
```

defaults
```
void
```

error messages
```
x :    "unknown";
```

examples

```
clear ("contexts");                    //clears the context's stack
GetProperties (context_count);
```

prop_name	prop_value	prop_type	prop_access
context_count	0	int32	r

```
contexts.new ();
GetProperties (context_count);
```

prop_name	prop_value	prop_type	prop_access
context_count	1	int32	r

CONTEXTS.Drop

description

Drops the topmost context on the stack

ressource will be freed

prototype

```
Contexts.Drop (int32 count)
```

if *count* = -1, the whole stack will be dropped.

this is equivalent to `session.clear ("contexts")`

defaults

```
count = 1
```

error messages

```
1 :    "empty contexts stack";
2 :    "drop failure";
3 :    "unable to generate response string";
x :    "unknown";
```

examples

```
clear ("contexts");
contexts.new ( );
contexts.SetProperties( "context_string = Context01");
contexts.new ( );
context.SetProperties( "context_string = Context02");
contexts.new ( );
contexts.SetProperties( "context_string = Context03");
contexts.new ( );
context.SetProperties( "context_string = Context04");
```

Stack
Context04
Context03
Context02
Context01

```
contexts.Drop ( );
contexts.GetProperties ("context_string");
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT 100 context03 100	string	rw

Stack
Context03
Context02
Context01

```
contexts.Drop (2);
contexts.GetProperties ("context_string");
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT 100 context01 100	string	rw

Stack
Context01

CONTEXTS.Dup

description

Duplicates the topmost context, or context block of size *range*, on top of the contexts stack

prototype

```
Contexts.Dup (int32 range)
```

defaults

```
range = 1
```

error messages

```
1 :    "empty contexts stack";
2 :    "dup failure";
3 :    "unable to generate response string";
x :    "unknown";
```

examples

```
clear ("contexts");
contexts.new ( );
context.SetProperties( "context_string = Context01");
contexts.new ( );
context.SetProperties( "context_string = Context02");
```

Stack
Context02
Context01

```
contexts.Dup ( );
```

Stack
Context02
Context02
Context01

```
contexts.Dup (3);
```

Stack
Context02
Context02
Context01
Context02
Context02
Context01

CONTEXTS.Swap

description

Swaps the topmost context, or context block of size *range*, on top of the contexts stack

prototype

```
Contexts.Swap (int32 range)
```

defaults

```
range = 1
```

error messages

```
1 :   "range out of limits or bad contexts stack count. must be >=2*range";
2 :   "swap failure";
3 :   "unable to generate response string";
x :   "unknown";
```

examples

```
clear ("contexts);
contexts.new ( );
context.SetProperties( "context_string = Context01");
contexts.new ( );
context.SetProperties( "context_string = Context02");
contexts.new ( );
context.SetProperties( "context_string = Context03");
contexts.new ( );
context.SetProperties( "context_string = Context04");
```

Stack
Context04
Context03
Context02
Context01

```
contexts.Swap ( );
```

Stack
Context03
Context04
Context02
Context01

```
contexts.Swap (2);
```

Stack
Context02
Context01
Context03
Context04

CONTEXTS.OnTop

description

Selects a context of the context's stack and put it topmost on the stack.

the *selection* parameter is a vartype parameter, eg, it can be an int32, or a string depending of it's shape.

it is first evaluated as an integer numeric value.

if true, *OnTop* will put the context at range *selection*, on top of the stack. the range of the context will become 1.

if it is a string, *OnTop* will select the first context whose name property is *selection*, and put it on top of the stack.

prototype

```
contexts.OnTop (vartype selection)
```

defaults

```
void
```

error messages

```
1 :    "system error unable to read parameters";
2 :    "index out of range must be [1, stack_count]";
3 :    "unable to put context on top of the stack";
4 :    "unable to find context named " + name +" on the stack";
5 :    "unable to put context named " + name +" on the stack";
6 :    "system. unable to generate response";
x :     "unknown error";
```

examples

```
clear ("contexts");
contexts.new ( );
context.SetProperties( "context_string = Context01");
contexts.setProperties ( "name = test");
contexts.getProperties ( "name ");
```

prop_name	prop_value	prop_type	prop_access
name	test	string	rw

```
contexts.new ( );
context.SetProperties( "context_string = Context02");
contexts.new ( );
context.SetProperties( "context_string = Context03");
contexts.new ( );
contexts.SetProperties( "context_string = Context04");
```

Stack
Context04
Context03
Context02
Context01 name=test

```
contexts.OnTop (2);
```

Stack
Context02
Context04
Context03
Context01 name=test

```
contexts.OnTop (test);
```

Stack
Context01 name=test
Context02
Context04
Context03

CONTEXTS.Intersection

description

Intersection of 2, or *range*+1 contexts on the context's stack

Activities will be consolidated according to the value of parameter *consolidation*

prototype

```
contexts.Intersection (int 32 range, string consolidation)
consolidation in ["simple", "min","max", "mean","maxinc"]
```

defaults

```
consolidation = "simple"
range = 1;
```

error messages

```
1 :    "contexts stack underflow";
2 :    "intersection failure. mode must be [simple, min,max, mean,maxinc]";
3 :    "unable to generate response string";
x :    "unknown error";
```

examples

CONTEXTS.Union

description

Union of 2, or *range*+1 contexts on the context's stack

Activities will be consolidated according to the value of parameter *consolidation*

prototype

```
contexts.Union (int 32 range, string consolidation)
consolidation in ["simple", "min","max", "mean","maxinc"]
```

defaults

```
consolidation = "simple"
range = 1;
```

error messages

```
1 :   "contexts stack underflow";
2 :   "union failure. mode must be [simple, min,max, mean,maxinc]";
3 :   "unable to generate response string";
x :   "unknown error";
```

examples

CONTEXTS.Amplify

description

Performs a linear transform ont activities in the topmost context of the context's stack

for each activity of a particule in the context

*activity = a*activity + b*

prototype

```
contexts.amplify (double a, double b);
```

defaults

```
a = 1.0

b = 0.0
```

error messages

```
1 :   "contexts stack underflow";
2 :   "unable to access parameters";
3 :   "unable to generate response string";
x :   "unknown error";
```

examples

```
Clear();                        /clears all session ressources
DocToContext(887);              /gets the particle description of the document #887
SetSpectrum ("evaluate = true ; min_activity = 30; max_generality = 3; max_atom = 15")
ApplySpectrum();                /select the more contextually activated shapes
contexts.Normalize();
```

#	shape	activity	gen_class	generality	id
1	magnétohydrodynamique	100	3	68	587
2	adiabaticité	93	0	15	10522
3	2M	86	0	14	11121
4	mw	77	2	43	20857
5	TH	75	2	37	67905
6	section_droite	74	2	52	84172
7	équations_de_maxwell	72	3	60	88468
8	t_H	68	3	61	94974
9	champ_magnétique	60	3	71	11725
10	bobine	60	3	68	11690
11	radiale	53	3	72	1884
12	denses	14	3	72	34970
13	basse_fréquence	9	3	58	52953
14	quantitative	9	3	72	1855

#	act	atoms	name
1	850	14	

(to be continued on next page…)

```
contexts.dup();               /make a copy of the context on top of the stack
contexts.amplify (0.8, -3);   /modify the activities
```

#	shape	activity	gen_class	generality	id	#	act	atoms	name
2	adiabaticité	71	0	15	10522	1	634	14	
3	2M	65	0	14	11121	2	850	14	
4	mw	58	2	43	20857				
5	TH	57	2	37	67905				
6	section_droite	56	2	52	84172				
7	équations_de_maxwell	54	3	60	88468				
8	t_H	51	3	61	94974				
9	champ_magnétique	45	3	71	11725				
10	bobine	45	3	68	11690				
11	radiale	39	3	72	1884				
12	denses	8	3	72	34970				
13	basse_fréquence	4	3	58	52953				
14	quantitative	4	3	72	1855				

```
contexts.dup();
contexts.amplify (-2);
```

#	shape	activity	gen_class	generality	id	#	act	atoms	name
1	magnétohydrodynamique	-154	3	68	587	1	-1268	14	
2	adiabaticité	-142	0	15	10522	2	634	14	
3	2M	-130	0	14	11121	3	850	14	
4	mw	-116	2	43	20857				
5	TH	-114	2	37	67905				
6	section_droite	-112	2	52	84172				
7	équations_de_maxwell	-108	3	60	88468				
8	t_H	-102	3	61	94974				
9	champ_magnétique	-90	3	71	11725				
10	bobine	-90	3	68	11690				
11	radiale	-78	3	72	1884				
12	denses	-16	3	72	34970				
13	basse_fréquence	-8	3	58	52953				
14	quantitative	-8	3	72	1855				

CONTEXTS.Split

not yet implemented

description

prototype

```
Contexts.Split (void)
```

defaults

```
void
```

error messages

```
1 :    "contexts stack underflow";
2 :    "unable to split context";
3 :    "unable to generate response string";
x :    "unknown error";
```

examples

CONTEXTS.Fetch

description

Fetches the content of a context.

it is different from context_string, since it gives the state of the particles present in the contex, and not a textual signal.

an iterator can use a sequence of

Fetch (n,1);	//gets the first n particles, starting at 1
Fetch();	// gets the next n particles, starting at n+1
Fetch();	// gets the next n particles, starting at 2*n+1

properties *contexts.fetch_size* and *contexts.fetch_start* maintain the next Fetch parameters

if propertie *contexts.fetch_start* = 0, then the end of the context has been reached

prototype

```
Contexts.Fetch (int32 size, int32 start, int32 index)
```

defaults

```
size = 48
start = 1   (base 1)
index = 1   (base 1)    the index of the context on the stack
```

error messages

```
1 :   "empty contexts stack ";
2 :   "unable to fetch context";
3 :   "unable to generate response string";
x :   "unknown error";
```

examples

```
clear( );
contexts.new ( );
contexts.SetProperties ( " context_string = atome_d_hydrogène") ;
contextToContext ( );
contexts.Fetch ();
```

shape	activity	gen_class	generality	id
gluon	100	2	46	57623
Z_10	98	3	63	60424
différents_états	96	4	68	55470
électrodynamique_quantique	94	3	61	57414
atomes_d_hydrogène	92	3	60	62757
atome	90	5	89	2161
macromoléculaire	88	4	78	31811
hydrocarbure	86	4	67	7123
rutherford	84	4	70	31376
gluons	82	3	63	18895
électronvolts	80	4	67	50353
mégaélectronvolts	78	3	59	15152
fouet	70	2	33	18816
prédire	68	2	37	18702
pouvant	62	8	137	4939
mobile	60	3	51	301

déficient	58	1	32	6283
survenir	56	2	39	38227
peut_avoir	54	2	40	84541
vaut	44	5	92	353
imposa	42	2	39	5628

```
contexts.Fetch (5,1);
```

shape	activity	gen_class	generality	id
gluon	100	2	46	57623
Z_10	98	3	63	60424
différents_états	96	4	68	55470
électrodynamique_quantique	94	3	61	57414
atomes_d_hydrogène	92	3	60	62757

```
contexts.Fetch ();
```

shape	activity	gen_class	generality	id
atome	90	5	89	2161
macromoléculaire	88	4	78	31811
hydrocarbure	86	4	67	7123
rutherford	84	4	70	31376
gluons	82	3	63	18895

```
contexts.GetProperties ("fetch_size; fetch_start");
```

prop_name	prop_value	prop_type	prop_access
fetch_size	5	uint32	rw
fetch_start	11	int32	rw

```
contexts.Fetch ();
```

shape	activity	gen_class	generality	id
électronvolts	80	4	67	50353
mégaélectronvolts	78	3	59	15152
fouet	70	2	33	18816
prédire	68	2	37	18702
pouvant	62	8	137	4939

```
contexts.GetProperties ("fetch_size; fetch_start");
```

prop_name	prop_value	prop_type	prop_access
fetch_size	5	uint32	rw
fetch_start	16	int32	rw

```
contexts.Fetch ();
contexts.Fetch ();
contexts.GetProperties ("fetch_size; fetch_start");
```

prop_name	prop_value	prop_type	prop_access
fetch_start	**0**	int32	rw
fetch_size	5	uint32	rw

CONTEXTS.SortBy

description

Sorts the topmost context on the context's stack

criterion can be	["generality", "activity"]
order can be	["descending", "ascending"]

prototype

```
Contexts.SortBy (string criterion, string order)
```

defaults

```
criterion   = "generality"
order       = "ascending"
```

error messages

```
1 :    "empty contexts stack";
2 :    "illegal sort order";
3 :    "sort failure";
4 :    "illegal sort criterion";
5 :    "unable to generate response string";
x :    "unknown error";
```

examples

CONTEXTS.Learn

description

Learns and detects contexts from the topmost context of the context's stack.

This is an advanced causal (contextual) method.

It is used in order to develop over-learning strategies.

It must be used on "sure" contexts, eg, contexts that have been deducted by a preliminary contextual analysis

prototype

```
contexts.Learn(void)
```

defaults

error messages

```
1 :   "empty contexts stack ";
2 :   "unable to learn context";
3 :   "unable to generate response string";
x :   "unknown error";
```

examples

CONTEXTS.Normalize

description

Normalizes the activity of atoms in the topmost context of the context's stack.

The highest abs (activity) will become the reference for 100%

The topmost context remains sorted by activity, descending order.

if *behaviour* is "*absolute*", the reference of the 100% will be the highest activity OR 100 if the highest absolute value of activity is less than 100.

if *behaviour* is "*relative*", the reference of the 100% will allways be the highest absolute value of activity.

prototype

```
contexts.Normalize(string behaviour)
behaviour can be ["absolute", "relative", "constant","order"]
```

defaults

```
behaviour = "absolute"
```

error messages

```
1 :    "empty stack";
2 :    "unable to decode parameter string";
3 :    "illegal mode. must be [absolute, relative, constant, order]";
4 :    "normalization failure";
5 :    "unable to write response string";
x :    "unknown";
```

examples

```
contexts.getProperties ( "context_string");
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT 283 atome 200 molécule 62 quark 21	string	rw

```
contexts.Normalize ();
contexts.getProperties ( "context_string");
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT 146 atome 100 molécule 31 quark 15	string	rw

example 2

```
contexts.getProperties ( "context_string");
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT160 atome 80 molécule 50 quark 30	string	rw

```
contexts.Normalize ("absolute");
contexts.getProperties ( "context_string");
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT160 atome 80 molécule 50 quark 30	string	rw

```
contexts.Normalize ("relative");
contexts.getProperties ( "context_string");
```

prop_name	prop_value	prop_type	prop_access
context_string	# mARC CONTEXT160 atome 100 molécule 62 quark 37	string	rw

example 3

Results

RESULTS.GetProperties

description

Gets one or several Result Set (RS) properties.

To access one property, use a directive as : propertyname
To access several properties values in one command, separate each directive with the character semi column (;)

if there are no parameter, all properties of the topmost context of the context's stack will be accessed

prototype

```
Contexts.GetProperties (string accessor, int32 index);
accessor = "propertyname1; …;propertynameN"
```

defaults

```
accessor = ""
index = 1   (base 1)    the index of the RS on the stack
```

error messages

```
1 :    "empty result stack";
2 :    "unable to read parameters";
3 :    "unable to decode parameter string";
4 :    "unable to read properties";
5 :    "unable to format output string";
x :    "unknown";
```

example 1

```
Clear();
results.GetProperties();
```

since the Results stack is empty,an error will be logged and the following error string will be returned :
code number = 1 message = empty result stack in results.GetProperties <30 results.GetProperties ( ) .../>

example 2

```
Clear();
table:EuMaster.Select (new,rowid, between, 0,5);    /select the 6 first lines in a new RS
results.SetProperties (name = example on EuMaster); /name the RS
table:EuDetail.Select (new,rowid, between, 0,7);    /select the 8 first lines in a new RS
results.SetProperties (name = example on EuDetail); /name the RS
```

#	count	name	table
1	8	example on EuDetail	EuDetail
2	6	example on EuMaster	EuMaster

the Results Stack contains 2 RS, each linked to a different table.

```
results.GetProperties();      /get all properties of topmost RS
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	example on EuDetail	string	rw
count	8	int32	r
owner_table	EuDetail	string	rw
format	RowId	string	rw
fetch_size	10	int32	r
fetch_start	1	int32	r
fetch_id	-1	int32	r

to be continued on next page…

```
results.GetProperties(1);     /get all properties of first (topmost) RS
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	example on EuDetail	string	rw
count	8	int32	r
owner_table	EuDetail	string	rw
format	RowId	string	rw
fetch_size	10	int32	r
fetch_start	1	int32	r
fetch_id	-1	int32	r

this is equivalent to the previous instruction line

```
results.GetProperties(2);     /get all properties of the 2th RS
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	example on EuMaster	string	rw
count	6	int32	r
owner_table	EuMaster	string	rw
format	RowId	string	rw
fetch_size	10	int32	r
fetch_start	1	int32	r
fetch_id	-1	int32	r

```
results.GetProperties(name;count); /get properties name & count of the topmost RS
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	example on EuDetail	string	rw
count	8	int32	r

```
results.GetProperties(name,1); /get property name of the first (topmost) RS
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	example on EuDetail	string	rw

```
results.GetProperties(name,2); /get property name of the 2th RS
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	example on EuMaster	string	rw

```
results.GetProperties(2, name;count); or results.GetProperties(2, "name;count");
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	example on EuMaster	string	rw
count	6	int32	r

there is no context ambiguity so *name;count* and "*name;count*" are interpreted in an equivalent way type of **idx**, and **accessor** are different, AND an accessor cannot have the shape of a number

```
results.GetProperties(name;count, 2); or results.GetProperties("name;count",2);
```

string	string	string	string
prop_name	prop_value	prop_type	prop_access
name	example on EuMaster	string	rw
count	6	int32	r

there is no context ambiguity since type of **idx**, and **accessor** are different, AND an accessor cannot have the shape of a number

RESULTS.SetProperties

description

Access to result's properties

Trying to change the value of a Read Only property will **not** generate an error.

To see which properties are avalaible, see CONTEXTS.GetProperties

To change one property, use a directive as : propertyname = propertyvalue
To change several properties values in one command, separate each directive with the character semi column (;)

properties of the topmost ResultSet of the context's stack will be changed

an accessor is a string like :

"propertyname = value"

and can be extended like

"propertyname1 = value1, … propertynameN = valueN"

Depending of your client application using only one extended accessor, as a parameter, is equivalent as using several accessors as parameters

prototype

```
Results.SetProperties ( [int32 index], string accessor, …string accessorN )


accessor = "propertyname1 = value1; …;propertynameN = value2"
index, optional, between  1 – stack_count
```

defaults

```
index = 1
```

error messages

```
0 :    "empty results stack";
1 :    "parameter error";
2 :    "unable to decode parameter string";
3 :    "unable to write properties";
4 :     custom message depending of the property error

5 :    "index out of range must be [1, stack_count]";
x :    "unknown";


custom messages
"error setting property result.XXX";
```

examples

```
Clear();
TABLE:EuMaster.SELECT("new", " RowId", " Between", " 100", " 120");
results.GetProperties ( );
```

prop_name	prop_value	prop_type	prop_access
count	21	int32	r
owner_table	EuMaster	string	rw
format	RowId	string	rw
fetch_size	10	int32	r
fetch_start	1	int32	r
fetch_id	-1	int32	r

```
table:eumaster.GetStructure();
```

name	type	size
id	INT32	0
titre	CHAR	106
soustitre	CHAR	141
rtf	STRING	0
texte	STRING	0
KNW_ABSTRACT	ABSTRACT	0
KNW_LANGAGE	INT32	0
KNW_MEANING	INT32	0

```
results.SetProperties ( "format = " rowid act titre soustitre "" );
results.getproperties ( "format")
```

prop_name	prop_value	prop_type	prop_access
format	RowId Act titre soustitre	string	rw

RESULTS.New

description

Creates a new ResultSet (RS) on top of the Result's stack.

a RS must be linked to a table.

by default, the RS is linked to the master table, if it exists, or to NULL.

in such a situation, this link can be explicitely set using the RESULTS.SetProperties method over the property owner_table.

prototype

```
results.new ()
```

defaults

property `owner_table` set to the master table if exists, void otherwise

error messages

```
x :    "unknown";
```

examples

```
Clear();                              //equivalent to session.clear…
GetProperties ( "result_count");      //equivalent to session.GetProperties…
```

prop_name	prop_value	prop_type	prop_access
result_count	0	int32	r

```
Results.New ( );
```

prop_name	prop_value	prop_type	prop_access
result_count	1	int32	r

```
results.getproperties ("owner_table");
```

prop_name	prop_value	prop_type	prop_access
owner_table	EuMaster	string	rw

Since *eumaster* is the present master table, the newly created RS is linked to it

```
table.GetInstances (  )
```

tables
EuDetail
EuMaster

```
results.SetProperties ( "owner_table = eudetail"  ) ;
results.getproperties ("owner_table");
```

prop_name	prop_value	prop_type	prop_access
owner_table	EuDetail	string	rw

RESULTS.Drop

description

Drops the topmost RS on the stack

ressource will be freed

prototype

```
Results.Drop (int32 count)
```

defaults

```
count = 1
```

error messages

```
1 :    "empty results stack";
2 :    "stack access";
3 :    "unable to access Result Set";;
4 :    "drop failure";
5 :    "unable to generate response string";
x :    "unknown";
```

example 1

```
clear ( " results") ;
results.setProperties ("format = rowid);
table:eumaster.Select( "new", "rowid", "between", "31", "40");
table:eumaster.Select( "new", "rowid", "between", "21", "30");
table:eumaster.Select( "new", "rowid", "between", "11", "20");
table:eumaster.Select( "new", "rowid", "between", "1", "10");
```

GetProperties ("result_count");

prop_name	prop_value	prop_type	prop_access
result_count	4	int32	r

stack
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40

Results.Drop ( );

GetProperties ("result_count");

prop_name	prop_value	prop_type	prop_access
result_count	3	int32	r

stack
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40

Results.Drop ("-1");

GetProperties ("result_count");

prop_name	prop_value	prop_type	prop_access
result_count	0	int32	r

stack

RESULTS.Dup

description

Duplicates the topmost Result Set on top of the RS stack

prototype

```
Results.Dup (void)
```

defaults

```
void
```

error messages

```
1 :    "results stack underflow";
2 :    "maximum RS stack count reached";
3 :    "unable to allocate a new RS";
4 :    "unable to copy data to the RS";
5 :    "unable to put new RS on the stack";
6 :    "unable to generate response string";
x :    "unknown";
```

example 1

```
clear ( " results") ;
results.setProperties ("format = rowid);
table:eumaster.Select( "new", "rowid", "between", "1", "10");
```

GetProperties ("result_count");

prop_name	prop_value	prop_type	prop_access
result_count	1	int32	r

stack
1 2 3 4 5 6 7 8 9 10

Results.Dup ( );

GetProperties ("result_count");

prop_name	prop_value	prop_type	prop_access
result_count	2	int32	r

stack
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10

RESULTS.Swap

description
Swaps the topmost Result Set on top of the RS stack

prototype
```
Results.Swap (void)
```

defaults
```
void
```

error messages
```
1 :    "results stack underflow. count >= 2";
2 :    "swap failure";
3 :    "unable to generate response string";
x :    "unknown";
```

example 1

```
clear ( " results") ;
results.setProperties ("format = rowid);
table:eumaster.Select  ( "new", "rowid", "between", "11", "20") ;
table:eumaster.Select  ( "new", "rowid", "between", "1", "10") ;

results.fetch ( "10", "1" );
```

RowId
1
2
3
4
5
6
7
8
9
10

stack
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20

```
Results.Swap ( );
results.fetch ( "10", "1" );
```

RowId
11
12
13
14
15
16
17
18
19
20

stack
11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10

RESULTS.OnTop

description

Selects a RS (Result Set) of the RSt's stack and put it topmost on the stack.

the *selection* parameter is a vartype parameter, eg, it can be an int32, or a string depending of it's shape.

it is first evaluated as an integer numeric value.

if true, *OnTop* will put the RS at range *selection*, on top of the stack. the range of the RS will become 1.

if it is a string, *OnTop* will select the first RS whose name property is *selection*, and put it on top of the stack.

prototype

```
results.OnTop (vartype selection)
```

defaults

```
void
```

error messages

```
1 :   "system error unable to read parameters";
2 :   "index out of range must be [1, stack_count]";
3 :   "unable to put context on top of the stack";
4 :   "unable to find context named " + name +" on the stack";
5 :   "unable to put context named " + name +" on the stack";
6 :   "system. unable to generate response";
x :    "unknown error";
```

examples

```
clear ( " results") ;
results.setProperties  (format = rowid);
table:eumaster.Select  ( new, rowid, between, 31, 40) ;
table:eumaster.Select  ( new, rowid, between, 21, 30) ;
results.SetProperties   (name = example) ;
table:eumaster.Select  ( new, rowid, between, 11, 20) ;
table:eumaster.Select  ( new, rowid, between, 1, 10) ;

results.fetch ( "10", "1" )
```

RowId
1
2
3
4
5
6
7
8
9
10

stack
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40

```
results.onTop ("example" );
results.fetch ( "6", "1" )
```

RowId
21
22
23
24
25
26

stack
21 22 23 24 25 26 27 28 29 30
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
31 32 33 34 35 36 37 38 39 40

RESULTS.Intersection

description

Intersection of 2, topmost Result Set (RS) on the RS's stack

Activities will be consolidated by summing the activities

prototype

```
Results.Intersection (void)
```

defaults

```
void
```

error messages

```
1 :    "unable to read parameters";
2 :    "stack underflow. count must be >=2";
3 :    "unable to access RS 1";
4 :    "unable to access RS 2";
5 :    "unable to create destination RS";
6 :    "intersection failure";
7 :    "unable to kill RS 1";
8 :    "unable to transfer data in destination RS";
9 :    "unable to kill RS 2";
10 :   "unable to get destination RS size";
11 :   "unable to generate response string";
x :     "unknown error";
```

examples

RESULTS.Union

description

Union of 2, topmost Result Set (RS) on the RS's stack

Activities will be consolidated by summing the activities

prototype

```
Results.Union (void)
```

defaults

```
void
```

error messages

```
1 :    "stack underflow. count must be >=2";
2 :    "unable to access RS 1";
3 :    "unable to access RS 2";
4 :    "unable to create destination RS";
5 :    "union failure";
6 :    "unable to kill RS 1";
7 :    "unable to transfer data in destination RS";
8 :    "unable to kill RS 2";
9 :    "unable to get destination RS size";
10 :   "unable to generate response string";
x :     "unknown error";
```

examples

RESULTS.SelectBy

description

Selects the rowids of the topmost Result Set (RS) on top of the RS's stacks, according to a given rule

Operator	alias	code	parameters	description
>	GT	0	1	Greater than op1
<	LT	1	1	Lower than op1
>=	GTE	2	1	Greater or Equal to op1
<=	LTE	3	1	Lower or Equal à op1
Between	BT	4	2	Between [op1, op2] (including op1 and op2)
=	EQ	5	1	Equal to op1
!=	NEQ	6	1	Différent from op1
&	AND	7	1	Logical AND with op1, if result != 0, then true
|	OR	8	1	Logical OR with op1, if result != 0, then true
BeginWith	BW	9	1	String begins with op1
EndWith	EW	10	1	String ends with op1
Contains	CO	11	1	String contains op1

prototype

```
Results.SelectBy (
                           string column,
                           string operator,
                           string operand1,
                           string operand2
                           )
```

defaults

```
void
```

error messages

```
1 :    "stack underflow. count must be >=1";
2 :    "no table linked to RS";
3 :    "unable to access parameters";
4 :    "unknown comparison operator";
5 :    "unknown column";
7 :    "RS write locked";
8 :    "unable lock RS";
9 :    "unable to access table";
10 :   "comparison failure";
11 :   "unable to generate response string";
x :     "unknown error";
```

example 1

```
clear ( " results") ;
results.setProperties ("format = rowid);
table:eumaster.Select  ( "new", "rowid", "between", "31", "40") ;
table:eumaster.Select  ( "new", "rowid", "between", "21", "30") ;
table:eumaster.Select  ( "new", "rowid", "between", "11", "20") ;
```

stack

```
table:eumaster.Select  ( "new", "rowid",
"between", "1", "10") ;
```

```
results.fetch ( "10", "1" )
```

RowId
1
2
3
4
5
6
7
8
9
10

1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40

RESULTS.DeleteBy

description

Deletes the rowids of the topmost Result Set (RS) on top of the RS's stacks, according to a given rule

prototype

```
Results.DeleteBy (
                            string column,
                            string operator,
                            string operand1,
                            string operand2
                            )
```

Operator	alias	code	parameters	description
>	GT	0	1	Greater than op1
<	LT	1	1	Lower than op1
>=	GTE	2	1	Greater or Equal to op1
<=	LTE	3	1	Lower or Equal à op1
Between	BT	4	2	Between [op1, op2] (including op1 and op2)
=	EQ	5	1	Equal to op1
!=	NEQ	6	1	Différent from op1
&	AND	7	1	Logical AND with op1, if result != 0, then true
|	OR	8	1	Logical OR with op1, if result != 0, then true
BeginWith	BW	9	1	String begins with op1
EndWith	EW	10	1	String ends with op1
Contains	CO	11	1	String contains op1

defaults

```
void
```

error messages

```
1 :    "stack underflow. count must be >=1";
2 :    "no table linked to RS";
3 :    "unable to access parameters";
4 :    "unknown comparison operator";
5 :    "unknown column";
7 :    "RS write locked";
8 :    "unable lock RS";
9 :    "unable to access table";
10 :   "comparison failure";
11 :   "unable to generate response string";
x :     "unknown error";
```

examples

RESULTS.SortBy

description

Sorts the topmost Result Set (RS) of the RS's stack, according to the content of the specified column of the linked table .

the linked table of a RS is accessible through the property *results.owner_table*

prototype

Results.SortBy (string column, string order)

order can be ["descending", "ascending"]

defaults

order = "descending"

error messages

```
1 :    "stack underflow. count must be >=1";
2 :    "no table linked to RS";
3 :    "unable to access parameters";
4 :    "unknown column";
6 :    "RS write locked";
7 :    "unable lock RS";
8 :    "not a sortable column type (variable size)";
9 :    "unknown comuùn type";
10 :   "sort failure";
11 :   "unable to generate response string";
x :     "unknown error";
```

examples

RESULTS.UniqueBy

description

Makes the topmost Result Set (RS) of the RS's stack unique, according to the content of the specified column of the linked table .

prototype

```
Results.UniqueBy (string column);
```

defaults

```
void
```

error messages

```
1 :    "stack underflow. count must be >=1";
2 :    "no table linked to RS";
3 :    "unable to access parameters";
4 :    "unknown column";
6 :    "RS write locked";
7 :    "unable lock RS";
9 :    "RS write locked";
10 :   "unable lock RS";
11 :   "not a compatible column type (variable size)";
12 :   "unknown column type";
13 :   "sort failure";
14 :   "sunicity failure";
15 :   "unable to generate response string";
x :     "unknown error";
```

examples

RESULTS.SelectToTable

description

Transforms the topmost Result Set (RS) of the RS's stack, linked to an initial table, into anothe RS linked to another table, acording to the content of a specified column.

the *column* parameter must be a colum containing rowid's of the destination table, specified by the parameter *table* of the method.

Since the *column* is containing a rowid, it must be of type int32 at least, or int64, for future extensions of the database capacity.

SelectToTable is useful to solve relations between several tables.

prototype

```
Results.SelectToTable (string column, string table, bool unique)
```

defaults

```
unique = true;
```

error messages

```
1 :    "stack underflow. count must be >=1";
2 :    "no table linked to RS";
3 :    "unable to access parameters";
4 :    "unknown column";
5 :    "unknown destination table";
6 :    "joint failure";
7 :    "source table unlock failure";
8 :    "linking failure";
9 :    "unable to link destination table to RS";
10 :   "unable to lock destination table";
11 :   "unable to perform unicity";
12 :   "unable to generate response string";
x :     "unknown error";
```

examples

RESULTS.Fetch

description

Fetches the content of the topmost Result Set (RS) of the RS's stack.

The results are fetched according to the format defined in the property *results.format*

an iterator can use a sequence of

Fetch (n,1);	//gets the first n particles, starting at 1
Fetch();	// gets the next n particles, starting at n+1
Fetch();	// gets the next n particles, starting at 2*n+1

properties *results.fetch_size* and *results.fetch_start* maintain the next Fetch parameters

if propertie *results.fetch_start* = 0, then the end of the RS has been reached

prototype

```
Results.Fetch (int 32 size, int32 start, int32 index )
```

defaults

```
size = 10
start = 1
index = 1   (base 1)    the index of the RS on the stack
```

error messages

```
1 :   "stack underflow. count must be >=1";
2 :   "no table linked to RS";
3 :   "unable to access parameters";
5 :   "unable to access line";
6 :   "unable to access column";
7 :   "unable to recover buffer ressource";
8 :   "unable to get table format";
9 :   "unable to generate response string";
x :    "unknown error";
```

example 1

```
clear ( " results") ;
results.setProperties ("format = rowid);
table:eumaster.Select  ( "new", "rowid", "between", "1", "10") ;

results.fetch ( "6", "2" )
```

RowId
2
3
4
5
6
7

stack
1 2 3 4 5 6 7 8 9 10

RowId
8
9

stack
1 2 3 4 5 6 7 8 9 10

```
results.fetch ( )
```

10

RESULTS.Normalize

description

Normalizes the activity of Id's in the topmost Result Set of the RS's stack.

The highest abs (activity) will become the reference for 100%

The topmost context remains sorted by activity, descending order.

if *behaviour* is "*absolute*", the reference of the 100% will be the highest activity OR 100 if the highest absolute value of activity is less than 100.

if *behaviour* is "*relative*", the reference of the 100% will allways be the highest absolute value of activity.

prototype

```
Results.Normalize(string behaviour)


behaviour can be ["absolute", "relative", "constant", "order"]
```

defaults

```
behaviour = "absolute"
```

error messages

```
1 :    "stack underflow. count must be >=1";
2 :    "unable to read parameters";
3 :    "normalize failure";
4 :    "unable to generate response string";
x :     "unknown error";
```

examples

RESULTS.Amplify

description

Performs a linear transform ont activities in the topmost RS of the RS's stack

for each activity of a particule in the Result Set (RS)

*activity = a*activity + b*

prototype

```
results.amplify (double a, double b);
```

defaults

```
a = 1.0
b = 0.0
```

error messages

```
x :     "unknown error";
```

examples

#,count,name,table

1,5,,EuMaster

	int32	int32	char
#	RowId	Act	titre
0	887	497	PLASMAS
1	32791	219	MAGNÉTOHYDRODYNAMIQUE
2	1252	186	MAGNÉTOHYDRODYNAMIQUE
3	19911	154	MAGNÉTOHYDRODYNAMIQUE
4	232	100	MAGNÉTOHYDRODYNAMIQUE