

Function Reference A-Z

Complete callable catalogue with small examples



LEARN	BUILD	REFERENCE
Small examples	Real Kokum interfaces	Searchable A-Z entries

Linux 1.1.0 • VM 1.7 • GUI contract 1.9
September 2026

The language designed and developed by Anil Jagtap.
Email: anil.softx@gmail.com

Includes DatePicker.GetDate, TRIM / LTRIM / RTRIM,
REST requests, TOTP and the current Linux distribution workflow.

Contents

Use the linked entries below or the document navigation pane. The A-Z index at the end links directly to individual entries.

About this edition	3
Read and run the examples	4
Reusable example setups	6
Built-in functions A-Z — 110 entries	12
Native resource methods — 112 entries	53
Control-proxy methods — 22 entries	91
Native resource property directory	99
HTTP, REST and TOTP details	103
Language and service recipes	105
Sources, scope and verification	108
A-Z callable index	110

About this edition

This is the current **Kokum Linux 1.1 developer edition**, including string trimming, the Designer rename correction, DatePicker and DatePicker.GetDate. It is written for users of the supplied Linux distribution; you do not need to compile Kokum itself.

Two books, complementary purposes

The GUI Designer Guide explains building interfaces and every entry in the current widget schema. The Function Reference is an A–Z catalogue of all 110 public global built-ins, 112 resource methods and 22 control-proxy methods, with a separate property directory. Neither book treats internal C entry points as user-facing functions.

Value / artifact	Meaning in this edition
1.1.0	Product label. Do not use this label alone as a runtime-compatibility check.
VM ABI 1.7	Needed by the new trimming built-ins; older-only applications retain their lower requirements.
GUI contract 1.9	Newly compiled DatePicker forms, including GetDate support.
.kokum / .kform	Language source / form source.
.kbc / .kres / .kapp	Compiled module / compiled form / complete application bundle.

Example conventions

A **Main body** listing goes inside FUNCTION Main AS INTEGER and ends with RETURN 0 / ENDFUNC. A **slot body** goes inside a bound procedure with sender, event and form parameters. SET-* labels identify reusable declarations, controls or external configuration. The reference supplies each setup explicitly; a context-dependent fragment is not presented as an independent console program.

Square brackets in a syntax signature mean optional arguments; they are not literal characters to copy. In executable snippets, brackets create or index real Kokum arrays. Kokum array indices are one-based; many GUI selection and TableView method indices are zero-based. Empty STRING and NULL are different values.

Known limitations are kept visible

The current shared GUI findings are TableView automatic-array-to-method handoff, programmatic Splitter position, one TextBox Visible=false path, and dynamic Image refresh. ModalDialog and SubFormHost are documented as restricted/legacy schema entries, not newly certified widgets. See the companion GUI guide's troubleshooting chapter.

Source references such as [S02] identify the authoritative files listed at the end. Screenshots from the earlier Designer manual are labelled as retained interface examples; they illustrate layout and workflow, not a new all-widget acceptance test.

Read and run the examples

The A-Z global section has one entry for each of the 110 public built-ins. The 112 resource methods are qualified by receiver type so that DATABASE.Close, REGEX.Close and WEBSOCKET.Close cannot be confused. The 22 selected control-proxy methods cover the actual DatePicker, ComboBox, TableView and PropertyGrid native surface. Public properties are listed separately; they are not additional global functions.

No invented functions

INT already converts a NUMBER to INTEGER. TRIM, LTRIM and RTRIM are present; RTREM is not an alias. There is no public SEEK() or EOF() in this catalogue. DATE() takes no argument. Internal KOKUMFIND is deliberately excluded. Planned financial indicators are not silently added to this reference.

Run a Main-body fragment

Kokum example

```
FUNCTION Main AS INTEGER
  && Insert one MAIN example here.
  ? INT(VAL("123"))
  RETURN 0
ENDFUNC
```

Linux terminal

```
kokumc check example.kokum
kokumc run-bytecode example.kokum
kokumc compile example.kokum -o example.kbc
kokumvm example.kbc
```

All four executed paths are useful for verification, but an ordinary user needs only the application run/compile commands provided by their installed tools. The compiler and runtime must meet the program's actual minimum ABI. These instructions never rebuild Kokum itself.

Run a GUI-slot fragment

Kokum example

```
EXPORT PROCEDURE btnSample_clicked

PROCEDURE btnSample_clicked(sender, event, form)
  && Insert a GUI example after its SET-* preparation.
  form.lblStatus.Text = "Ready"
ENDPROC
```

Create/configure the named controls in the Designer and bind btnSample.clicked to this routine. The generated variables are supplied by the form/runtime. Do not create a DATABASE, GRAPH, IMAGE or WEBSOCKET merely by declaring its type in a standalone console program.

Label	Meaning
MAIN	Paste into Main; no additional declarations unless named in the entry.
SET-REFLECTION / THREAD / TASK	Add the reusable declaration before Main.
SET-HTTP / FLATDB	Start the stated local service and use the matching private configuration.
SET-DATABASE / GRAPH / IMAGE / WEBSOCKET	Configure the native resource and run in its owning GUI session.
SET-TABLE / COMBO / GRID / DATE	Use the actual control MAP, normally form.controlId.
SET-REMOTE-DATABASE	Requires a real permitted remote server; source/compiler checked, not live-server executed for these books.

Types, missing values and side effects

NUMBER is binary floating point; INTEGER is a signed integral type; DECIMAL preserves exact decimal intent. Do not silently use fractional IDs or route large integer text through VAL before conversion. INT truncates toward zero, so validate whole-number intent. NULL is not zero, an empty STRING or an empty ARRAY. Check nullable query/receive results before indexing.

AADD/ASET/AINS/ADEL and map setters mutate their target collections. File modes w/w+ can truncate files. SQL examples write only a disposable lab database. Network credentials are placeholders for local demonstrations; no real credentials appear in the books. Timeout, task completion, cancellation and transaction rollback are distinct contracts.

Reusable example setups

SET-REFLECTION — a small class

Kokum example

```
CLASS Person
  PROPERTY Name AS STRING = "Anil"
  METHOD Greet AS STRING
    RETURN "Hello " + THIS.Name
  ENDMETHOD
ENDCLASS
```

Place this class before Main. Reflection examples create NEW Person(). PROPERTY/METHOD names and return types remain exactly as shown.

SET-THREAD — named RUN routines

Kokum example

```
FUNCTION Work AS INTEGER
  RETURN 7
ENDFUNC

PROCEDURE SlowWork
  SLEEP(10)
ENDPROC
```

Named RUN returns a task-like handle identified by the chosen name. WAIT FOR coordinates it. THREADSTATUS reports uppercase task states; task.Status is lowercase. A first observed state depends on scheduling.

SET-TASK — asynchronous function

Kokum example

```
ASYNC FUNCTION Double(value AS INTEGER) AS INTEGER
  RETURN value * 2
ENDFUNC
```

Place this before Main. Calling Double(21) returns a TASK in the asynchronous interface; Result/Await/Wait read the same cached result. The bytecode VM provides the worker-backed execution. The ordinary AST/IR ASYNC reference paths may finish inline; do not infer background GUI responsiveness from those paths alone.

SET-COMBO, SET-TABLE and SET-GRID

Add ComboBox cmbRole, TableView tblData and PropertyGrid gridInfo. Before each independent method example, run the relevant preparation in the same slot or restart the form with a clean state.

SET-COMBO

```
form.cmbRole.SetItems(["Analyst", "Developer"])
```

SET-TABLE — native-method-owned rows

```
form.tblData.ClearColumns()
form.tblData.AddColumn("id", "ID", 80)
form.tblData.AddColumn("name", "Name", 180)
form.tblData.AddRow(["1", "Anil"])
form.tblData.AddRow(["2", "Sara"])
```

Do not replace tblData through its automatic ARRAY and then use later native methods in this example. The array-to-method handoff remains a known separate defect. Native row/column positions are zero-based; item/array positions elsewhere may be one-based.

SET-GRID

```
form.gridInfo.Clear()
form.gridInfo.AddProperty( ;
    "General", "title", "Title", "Kokum", "string")
```

SET-DATE — DatePicker control proxy

Add DatePicker dtAdmission. Its generated variable is STRING; the GetDate method belongs to the form.dtAdmission MAP. Use the updated Linux GUI1.9 runtime.

Kokum example

```
form.dtAdmission.MinDate = ""
form.dtAdmission.MaxDate = ""
form.dtAdmission.Value = "2026-09-20"
```

SET-GRAPH and SET-IMAGE

Add Graph graphSales and Image imgLogo. Their generated native resources are addressed directly, not through a guessed JavaScript interface.

SET-GRAPH — prepare before each independent sample

```
graphSales.Clear()
graphSales.SetType("line")
graphSales.SetData(["Jan","Feb","Mar"], [10,20,30], "Sales")
graphSales.AddSeries("Target", [12,18,25])
```

For IMAGE.SetImage, place an actual PNG at images/kokum.png relative to the runtime working directory. A bundled web image is not automatically extracted as a host file. For portable designed imagery, set Source to images/kokum.png and bundle it with target web/images/kokum.png. Native loading and stable rendered refresh are distinct; the latter retains an open issue.

SET-DATABASE — private local SQLite lab

Add SQLite dbLocal. Use the actual slots-module name in external configuration. Every example assumes a fresh disposable lab, not an existing business database. Start the application with KOKUM_DB_ENV pointing to this private file.

Configuration fragment

```
[database:dbLocal]
provider=sqlite
module=book.slots
file=/absolute/path/to/private-lab/reference.sqlite3
auto_open=true
open_mode=readwrite-create
create_parent_directory=true
journal_mode=DELETE
synchronous=FULL
foreign_keys=true
busy_timeout_ms=1000
```

Linux terminal

```
export KOKUM_DB_ENV=/absolute/path/to/private-lab/db_env.conf
kokumvm /absolute/path/to/reference.kapp
```

Run this preparation before independent database examples

```
dbLocal.Execute("CREATE TABLE IF NOT EXISTS people " + ;
    "(id INTEGER PRIMARY KEY, name TEXT)")
dbLocal.Execute("CREATE TABLE IF NOT EXISTS log (message TEXT)")
dbLocal.Execute("INSERT OR REPLACE INTO people(id,name) " + ;
    "VALUES (?,?)", [1,"Anil"])
```

The test harness recreated the lab tables before each independent example. In your own fresh lab, the code above prepares people and log. CreateTableFromJson uses a fresh imported_people name; its displayed DROP affects only that disposable lab table. Do not run destructive examples against a live application database.

SET-REMOTE-DATABASE — operator-supplied server

DATABASE.Ping, UseDatabase and Raw are not portable SQLite operations. Configure a permitted TigerDB component dbTiger (or the specific PostgreSQL Ping alternative) with a dedicated test server. Keep credentials in the external config/environment, require verified TLS for untrusted networks and use an account limited to the intended lab database.

The three remote-specific entries were compiler-checked but not executed against live TigerDB/PostgreSQL servers for this book. No fake result is substituted. Ordinary local SQLite Query/Execute/transaction examples were executed through real compiled slots.

SET-FORMS — an additional form

Create and declare a second Window named CustomerView in the same GUI project. A Button on MainWindow can call SHOWFORM/HIDEFORM/CLOSEFORM with "CustomerView". These three independent calls must run in separate actions to observe their effects. The functions request existing form actions; they do not create an undeclared form.

Configuration fragment

```
[[form]]
name = "CustomerView"
source = "ui/customer.kform"
slots = "src/slots.kokum"
```

SET-FLATDB — local Remote FlatDB server

Use a publisher-supplied kokum-flatdb-server. In a private lab directory, generate a 64-byte key, protect it, and start the server. The openssl command below is optional lab setup tooling, not a dependency of normal Kokum application execution.

Linux terminal

```
mkdir -p "$HOME/kokum-flatdb-lab"
cd "$HOME/kokum-flatdb-lab"
openssl rand -out company.kkey 64
chmod 600 company.kkey
kokum-flatdb-server --database company.kfdb \
  --key company.kkey --listen 127.0.0.1 --port 9437 --create
```

In the application working directory, place a protected copy of that same key and kdb.conf below. Keep the database/key names in agreement. The examples use one local lab endpoint; do not copy this secret into a distributable KAPP.

kdb.conf

```
[connection]
host=127.0.0.1
port=9437
database=company.kfdb
key_file=company.kkey
connect_timeout_ms=2000
request_timeout_ms=15000
max_message_bytes=16777216
```

KCONNECT establishes the current context's connection; KEXECUTE/KQUERY accept a single SQL STRING, not an ARRAY of parameters. Only use the fixed synthetic SQL in these examples. A wrong key is denied without a successful authenticated session. The client protocol does not turn the database file into an encrypted-at-rest vault. [S07]

SET-HTTP — a deterministic local endpoint

Start the following small Python3 service in a separate terminal. This is test support for the URL examples, not a Kokum runtime dependency and not a public production server. It binds loopback only and returns a fixed greeting or the JSON body sent to it.

`http_lab.py`

```
import http.server, json

class Handler(http.server.BaseHTTPRequestHandler):
    def log_message(self, *args):
        pass
    def reply(self, value):
        body = json.dumps(value).encode("utf-8")
        self.send_response(200)
        self.send_header("Content-Type", "application/json")
        self.send_header("Content-Length", str(len(body)))
        self.end_headers()
        self.wfile.write(body)
    def do_GET(self):
        self.reply({"message": "Hello from Kokum"})
    def do_POST(self):
        n = int(self.headers.get("Content-Length", "0"))
        if n > 65536:
            self.send_error(413)
            return
        raw = self.rfile.read(n)
        try:
            value = json.loads(raw or b"{}")
        except (ValueError, UnicodeDecodeError):
```

`http_lab.py — continued`

```
            self.send_error(400)
            return
        self.reply(value)
    do_PUT = do_POST
    do_PATCH = do_POST
    do_DELETE = do_POST
    def do_HEAD(self):
        self.send_response(200)
        self.send_header("Content-Length", "0")
        self.end_headers()
    def do_OPTIONS(self):
        self.reply({"message": "options"})

with http.server.ThreadingHTTPServer(
    ("127.0.0.1", 18080), Handler) as server:
    print("Local HTTP lab: 127.0.0.1:18080", flush=True)
    server.serve_forever()
```

Linux terminal

```
python3 http_lab.py
```

The three URL examples were executed against this same response contract. The service has no authentication and must not receive real credentials. The service-writing chapter also demonstrates a Kokum handler, with separate application configuration.

SET-WEBSOCKET — local echo lab

Add WebSocketClient wsFeed. Set URL to `ws://127.0.0.1:18082/echo`, Auto Connect=false and Reconnect=false. Save the next consecutive blocks as `echo_lab.py` and run it in another terminal. This small test peer accepts only unfragmented, masked lab traffic up to 64 KiB; it is not an RFC-complete or production WebSocket server.

echo_lab.py

```

import base64, hashlib, socketserver, struct

def readn(sock, size):
    data = b""
    while len(data) < size:
        part = sock.recv(size - len(data))
        if not part:
            raise EOFError()
        data += part
    return data

def send_frame(sock, opcode, payload):
    size = len(payload)
    header = bytes([0x80 | opcode])
    header += (bytes([size]) if size < 126 else
               b"\x7e" + struct.pack("!H", size))
    sock.sendall(header + payload)

class Echo(socketserver.BaseRequestHandler):
    def handle(self):
        sock = self.request
        sock.settimeout(30)
        try:

```

echo_lab.py — continued

```

        header = b""
        while not header.endswith(b"\r\n\r\n"):
            header += readn(sock, 1)
            if len(header) > 16384:
                return
        fields = {}
        for line in header.decode("ascii").split("\r\n")[1:]:
            if ":" in line:
                key, value = line.split(":", 1)
                fields[key.lower()] = value.strip()
        key = fields["sec-websocket-key"]
        seed = key + "258EAF5E914-47DA-95CA-C5AB0DC85B11"
        accept = base64.b64encode(
            hashlib.shal(seed.encode()).digest()).decode()
        reply = "HTTP/1.1 101 Switching Protocols\r\n"
        reply += "Upgrade: websocket\r\nConnection: Upgrade\r\n"
        reply += "Sec-WebSocket-Accept: " + accept + "\r\n"
        if fields.get("sec-websocket-protocol"):
            protocol = fields["sec-websocket-protocol"].split(",")[0]
            reply += "Sec-WebSocket-Protocol: " + protocol.strip()
            reply += "\r\n"
        sock.sendall((reply + "\r\n").encode())
        while True:

```

echo_lab.py — continued

```

            first, second = readn(sock, 2)
            opcode, size = first & 15, second & 127
            if not (first & 0x80 and second & 0x80):
                return
            if first & 0x70 or size == 127:
                return
            if size == 126:
                size = struct.unpack("!H", readn(sock, 2))[0]
            if opcode >= 8 and size > 125:
                return
            mask = readn(sock, 4)
            data = readn(sock, size)
            data = bytes(v ^ mask[i % 4]
                         for i, v in enumerate(data))
            if opcode in (1, 2):
                send_frame(sock, opcode, data)
            elif opcode == 9:
                send_frame(sock, 10, data)
            elif opcode == 8:
                send_frame(sock, 8, data)
            return
        except (OSError, EOFError, ValueError, KeyError):
            return

```

echo_lab.py — continued

```
class Server(socketserver.ThreadingTCPServer):
    allow_reuse_address = True
    daemon_threads = True

with Server(("127.0.0.1", 18082), Echo) as server:
    print("Local echo: ws://127.0.0.1:18082/echo", flush=True)
    server.serve_forever()
```

Linux terminal

```
python3 echo_lab.py
```

Reset before independent configuration examples

```
wsFeed.Close()
wsFeed.SetUrl("ws://127.0.0.1:18082/echo")
wsFeed.SetProtocols("json")
wsFeed.SetReconnect(.F., 1000, 0)
wsFeed.ClearMessages()
```

For sending/receiving examples: connect, then queue a small echo

```
wsFeed.Connect()
wsFeed.ClearMessages()
wsFeed.SendJSON({"message": "hello"})
```

Allow the native queue to receive the echo before inspecting it, or use a bounded Receive timeout. Different event/data methods filter messages differently; NULL is a legitimate no-message result. Do not reuse a stale assumed queue state across independent examples. A successful Connect does not mean every queued Send was delivered. Synthetic token/cookie strings in the reference must never be replaced with live secrets on this plaintext lab connection. [S09]

Built-in functions A-Z

110 public global functions. Stable BI identifiers follow the source catalogue; the private KOKUMFIND entry is excluded. Aliases with separate public IDs retain individual entries. The result type is the catalogue return category; ANY can represent multiple valid types, including NULL where the entry specifies it.

AADD

BI-017 / COLLECTIONS AND CODEBLOCKS

Syntax

```
AADD(array, value)
```

Appends one value to an array.

Return: ANY. **Arguments:** 2.

Main body

```
LOCAL a AS ARRAY
a = []
AADD(a, "first")
? a[1]
```

Result / effect: Prints first.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

ABS

BI-005 / TEXT, CONVERSION AND JSON

Syntax

```
ABS(number)
```

Returns the absolute numeric value.

Return: ANY. **Arguments:** 1.

Main body

```
? ABS(-12.5)
```

Result / effect: Prints 12.5.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

ADEL

BI-020 / COLLECTIONS AND CODEBLOCKS

Syntax

```
ADEL(array, index)
```

Removes and returns a one-based array element.

Return: ANY. **Arguments:** 2.

Main body

```
LOCAL a AS ARRAY
a = ["a", "b"]
? ADEL(a, 1), LEN(a)
```

Result / effect: Prints a and 1.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

AINS

BI-019 / COLLECTIONS AND CODEBLOCKS

Syntax

```
AINS(array, index, value)
```

Inserts a value at a one-based array position.

Return: ANY. **Arguments:** 3.

Main body

```
LOCAL a AS ARRAY
a = ["a", "c"]
AINS(a, 2, "b")
? JSONENCODE(a)
```

Result / effect: Prints ["a","b","c"].

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

APPEND

BI-064 / FILE I/O

Syntax

```
APPEND(file, text)
```

Writes UTF-8 text at end of file and returns bytes written.

Return: INTEGER. **Arguments:** 2.

Main body

```
LOCAL file AS FILE
file = OPEN("sample.txt", "a")
APPEND(file, "Second line\n")
CLOSEFILE(file)
```

Result / effect: Adds Second line without replacing existing data.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

ASET

BI-018 / COLLECTIONS AND CODEBLOCKS

Syntax

```
ASET(array, index, value)
```

Replaces an existing one-based array element.

Return: ANY. **Arguments:** 3.

Main body

```
LOCAL a AS ARRAY
a = [10, 20]
ASET(a, 2, 25)
? a[2]
```

Result / effect: Prints 25.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

ATOMICADD

BI-084 / TASKS AND SYNCHRONIZATION

Syntax

```
ATOMICADD(counter, delta)
```

Adds a delta atomically and returns the new value.

Return: INTEGER. **Arguments:** 2.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(10)
? ATOMICADD(counter, 5)
```

Result / effect: Prints 15.

SET returns the previous integer; ADD/INC/DEC return the new integer. CompareExchange returns a LOGICAL success flag. Ordinary ARRAY/MAP mutation is not made thread-safe by these functions.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICCOMPAREEXCHANGE

BI-087 / TASKS AND SYNCHRONIZATION

Syntax

```
ATOMICCOMPAREEXCHANGE(counter, expected, desired)
```

Replaces the value only when it equals expected.

Return: LOGICAL. **Arguments:** 3.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? ATOMICCOMPAREEXCHANGE(counter, 5, 9), counter.Value
```

Result / effect: Prints .T. and 9.

SET returns the previous integer; ADD/INC/DEC return the new integer. CompareExchange returns a LOGICAL success flag. Ordinary ARRAY/MAP mutation is not made thread-safe by these functions.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICDEC

BI-086 / TASKS AND SYNCHRONIZATION

Syntax

```
ATOMICDEC(counter)
```

Decrements atomically and returns the new value.

Return: INTEGER. **Arguments:** 1.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(2)
? ATOMICDEC(counter)
```

Result / effect: Prints 1.

SET returns the previous integer; ADD/INC/DEC return the new integer. CompareExchange returns a LOGICAL success flag. Ordinary ARRAY/MAP mutation is not made thread-safe by these functions.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICGET

BI-082 / TASKS AND SYNCHRONIZATION

Syntax

```
ATOMICGET(counter)
```

Reads an atomic integer.

Return: INTEGER. **Arguments:** 1.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(10)
? ATOMICGET(counter)
```

Result / effect: Prints 10.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINC

BI-085 / TASKS AND SYNCHRONIZATION

Syntax

```
ATOMICINC(counter)
```

Increments atomically and returns the new value.

Return: INTEGER. **Arguments:** 1.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(0)
? ATOMICINC(counter)
```

Result / effect: Prints 1.

SET returns the previous integer; ADD/INC/DEC return the new integer. CompareExchange returns a LOGICAL success flag. Ordinary ARRAY/MAP mutation is not made thread-safe by these functions.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINTEGER

BI-081 / TASKS AND SYNCHRONIZATION

Syntax

```
ATOMICINTEGER([initial])
```

Creates an atomic signed integer, optionally with an initial value.

Return: ATOMICINTEGER. **Arguments:** 0..1.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(10)
? counter.Value
```

Result / effect: Prints 10.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICSET

BI-083 / TASKS AND SYNCHRONIZATION

Syntax

```
ATOMICSET(counter, value)
```

Sets an atomic value and returns the previous value.

Return: INTEGER. **Arguments:** 2.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(10)
? ATOMICSET(counter, 20), ATOMICGET(counter)
```

Result / effect: Prints 10 and 20.

SET returns the previous integer; ADD/INC/DEC return the new integer. CompareExchange returns a LOGICAL success flag. Ordinary ARRAY/MAP mutation is not made thread-safe by these functions.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

AVERAGE

BI-054 / MATHEMATICS AND STATISTICS

Syntax

```
AVERAGE(values)
```

Alias of MEAN.

Return: ANY. **Arguments:** 1.

Main body

```
? AVERAGE([10, 20, 30])
```

Result / effect: Prints 20.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

CEIL

BI-042 / MATHEMATICS AND STATISTICS

Syntax

```
CEIL(number)
```

Returns the least integer not below the input.

Return: ANY. **Arguments:** 1.

Main body

```
? CEIL(3.1)
```

Result / effect: Prints 4.

An integral-looking result need not have INTEGER type. These functions preserve the numeric family where practical; use INT explicitly when an INTEGER is required.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

CEILING

BI-043 / MATHEMATICS AND STATISTICS

Syntax

```
CEILING(number)
```

Alias of CEIL.

Return: ANY. **Arguments:** 1.

Main body

```
? CEILING(3.1)
```

Result / effect: Prints 4.

An integral-looking result need not have INTEGER type. These functions preserve the numeric family where practical; use INT explicitly when an INTEGER is required.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

CHANNEL

BI-088 / TASKS AND SYNCHRONIZATION

Syntax

```
CHANNEL(capacity)
```

Creates a bounded transferable message channel.

Return: CHANNEL. **Arguments:** 1.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
? queue.Capacity
```

Result / effect: Prints 2.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNELCAPACITY

BI-093 / TASKS AND SYNCHRONIZATION

Syntax

```
CHANNELCAPACITY(channel)
```

Returns the configured channel capacity.

Return: INTEGER. **Arguments:** 1.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(3)
? CHANNELCAPACITY(queue)
```

Result / effect: Prints 3.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNELCLOSE

BI-091 / TASKS AND SYNCHRONIZATION

Syntax

```
CHANNELCLOSE(channel)
```

Closes a channel and wakes blocked senders/receivers.

Return: LOGICAL. **Arguments:** 1.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(1)
? CHANNELCLOSE(queue)
```

Result / effect: Prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNELCLOSED

BI-094 / TASKS AND SYNCHRONIZATION

Syntax

```
CHANNELCLOSED(channel)
```

Tests whether a channel is closed.

Return: LOGICAL. **Arguments:** 1.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(1)
CHANNELCLOSE(queue)
? CHANNELCLOSED(queue)
```

Result / effect: Prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNELCOUNT

BI-092 / TASKS AND SYNCHRONIZATION

Syntax

```
CHANNELCOUNT(channel)
```

Returns the current queued-message count.

Return: INTEGER. **Arguments:** 1.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
CHANNELSEND(queue, 1)
? CHANNELCOUNT(queue)
```

Result / effect: Prints 1.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNELRECEIVE

BI-090 / TASKS AND SYNCHRONIZATION

Syntax

```
CHANNELRECEIVE(channel [, timeoutMs])
```

Receives one value, optionally with a timeout; closed and drained returns NULL.

Return: ANY. **Arguments:** 1..2.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(1)
CHANNELSEND(queue, 42)
? CHANNELRECEIVE(queue)
```

Result / effect: Prints 42.

Timeouts are milliseconds and have operation-specific outcomes. A wait timeout is not task cancellation. Never block the GUI owner indefinitely.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNELSEND

BI-089 / TASKS AND SYNCHRONIZATION

Syntax

```
CHANNELSEND(channel, value [, timeoutMs])
```

Sends one copied value, optionally with a timeout.

Return: LOGICAL. **Arguments:** 2..3.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(1)
? CHANNELSEND(queue, "hello", 1000)
```

Result / effect: Prints .T.

Timeouts are milliseconds and have operation-specific outcomes. A wait timeout is not task cancellation. Never block the GUI owner indefinitely.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNELSTATUS

BI-095 / TASKS AND SYNCHRONIZATION

Syntax

```
CHANNELSTATUS(channel)
```

Returns capacity, count, waiters, and counters.

Return: MAP. **Arguments:** 1.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
? JSONENCODE(CHANNELSTATUS(queue))
```

Result / effect: Prints a status map.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CLAMP

BI-051 / MATHEMATICS AND STATISTICS

Syntax

```
CLAMP(value, minimum, maximum)
```

Constrains a value to an inclusive minimum/maximum range.

Return: ANY. **Arguments:** 3.

Main body

```
? CLAMP(15, 0, 10)
```

Result / effect: Prints 10.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

CLASSNAME

BI-026 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
CLASSNAME(object)
```

Returns the effective class name of an object.

Return: STRING. **Arguments:** 1.

Main body + SET-REFLECTION

```
LOCAL p
p = NEW Person()
? CLASSNAME(p)
```

Result / effect: Prints Person.

Use the Person class from SET-REFLECTION.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

CLOSEFILE

BI-065 / FILE I/O

Syntax

```
CLOSEFILE(file)
```

Closes a FILE resource idempotently; NULL is safe.

Return: NULL. No useful return value is required. **Arguments:** 1.

Main body

```
LOCAL file AS FILE
file = OPEN("close-demo.txt", "w")
CLOSEFILE(file)
CLOSEFILE(file)
? "closed"
```

Result / effect: Prints closed. Repeated closure is safe.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

CLOSEFORM

BI-037 / FORMS

Syntax

```
CLOSEFORM(formName)
```

Closes a compiled form by identifier.

Return: NULL. No useful return value is required. **Arguments:** 1.

Slot body + SET-FORMS

```
CLOSEFORM("CustomerView")
```

Result / effect: CustomerView closes.

Verification: Executed in a compiled Linux slot; setup required. Source: [S01]; current signature also checked against [S01].

COUNTNUM

BI-059 / MATHEMATICS AND STATISTICS

Syntax

```
COUNTNUM(values)
```

Counts numeric, non-NULL elements in an array.

Return: INTEGER. **Arguments:** 1.

Main body

```
? COUNTNUM([10, NULL, 20])
```

Result / effect: Prints 2.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

DATE

BI-010 / TEXT, CONVERSION AND JSON

Syntax

```
DATE()
```

Returns the current local date.

Return: DATE. **Arguments:** 0.

Main body

```
? TYPEOF(DATE())
```

Result / effect: Prints DATE.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

DATETIME

BI-011 / TEXT, CONVERSION AND JSON

Syntax

```
DATETIME()
```

Returns the current local date and time.

Return: DATETIME. **Arguments:** 0.

Main body

```
? TYPEOF(DATETIME())
```

Result / effect: Prints DATETIME.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

DECIMAL

BI-008 / TEXT, CONVERSION AND JSON

Syntax

```
DECIMAL(value)
```

Converts compatible input to exact DECIMAL.

Return: DECIMAL. **Arguments:** 1.

Main body

```
? DECIMAL("19.95") + 0.05M
```

Result / effect: Prints 20; the result remains an exact DECIMAL.

Use quoted decimal text or an M-suffixed numeric literal to preserve base-10 intent. Converting an already rounded NUMBER does not recreate lost digits. DECIMAL and NUMBER are not silently interchangeable.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

EVAL

BI-016 / COLLECTIONS AND CODEBLOCKS

Syntax

```
EVAL(block [, arguments ...])
```

Invokes a CODEBLOCK with optional arguments.

Return: ANY. **Arguments:** 1..many.

Main body

```
? EVAL({|x| x * 2}, 5)
```

Result / effect: Prints 10.

The first argument must be a CODEBLOCK. No runtime source-evaluation facility is implied.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

EXP

BI-047 / MATHEMATICS AND STATISTICS

Syntax

```
EXP (number)
```

Returns e raised to the supplied power.

Return: NUMBER. **Arguments:** 1.

Main body

```
? EXP (0)
```

Result / effect: Prints 1.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

FLOOR

BI-041 / MATHEMATICS AND STATISTICS

Syntax

```
FLOOR (number)
```

Returns the greatest integer not above the input.

Return: ANY. **Arguments:** 1.

Main body

```
? FLOOR (3.9)
```

Result / effect: Prints 3.

An integral-looking result need not have INTEGER type. These functions preserve the numeric family where practical; use INT explicitly when an INTEGER is required.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

GCCOLLECT

BI-033 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
GCCOLLECT ()
```

Runs explicit cycle collection and returns the number of reclaimed nodes.

Return: INTEGER. **Arguments:** 0.

Main body

```
LOCAL cycle
cycle = []
AADD(cycle, cycle)
cycle = NULL
? TYPEOF(GCCOLLECT())
```

Result / effect: Prints INTEGER. The reclaimed-node count is runtime-dependent.

Explicit collection is diagnostic; do not depend on a particular reclamation count.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

GCSTATS

BI-034 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
GCSTATS()
```

Returns runtime memory and garbage-collection diagnostics.

Return: MAP. **Arguments:** 0.

Main body

```
? TYPEOF(GCSTATS())
```

Result / effect: Prints MAP. Inspect the map for runtime-specific counters.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

GETPROPERTY

BI-031 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
GETPROPERTY(object, name)
```

Reads an object property by name.

Return: ANY. **Arguments:** 2.

Main body + SET-REFLECTION

```
LOCAL p
p = NEW Person()
? GETPROPERTY(p, "Name")
```

Result / effect: Prints Anil.

Use the Person class from SET-REFLECTION.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

HASKEY

BI-022 / COLLECTIONS AND CODEBLOCKS

Syntax

```
HASKEY(map, key)
```

Tests whether a map contains a key.

Return: LOGICAL. **Arguments:** 2.

Main body

```
? HASKEY({"id": 7}, "id")
```

Result / effect: Prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

HASMETHOD

BI-030 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
HASMETHOD(object, name)
```

Tests whether an object exposes a method.

Return: LOGICAL. **Arguments:** 2.

Main body + SET-REFLECTION

```
LOCAL p
p = NEW Person()
? HASMETHOD(p, "Greet")
```

Result / effect: Prints .T.

Use the Person class from SET-REFLECTION.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

HASPROPERTY

BI-029 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
HASPROPERTY(object, name)
```

Tests whether an object exposes a property.

Return: LOGICAL. **Arguments:** 2.

Main body + SET-REFLECTION

```
LOCAL p
p = NEW Person()
? HASPROPERTY(p, "Name")
```

Result / effect: Prints .T.

Use the Person class from SET-REFLECTION.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

HIDEFORM

BI-036 / FORMS

Syntax

```
HIDEFORM(formName)
```

Hides a compiled form by identifier.

Return: NULL. No useful return value is required. **Arguments:** 1.

Slot body + SET-FORMS

```
HIDEFORM("CustomerView")
```

Result / effect: CustomerView becomes hidden.

Verification: Executed in a compiled Linux slot; setup required. Source: [S01]; current signature also checked against [S01].

INT

BI-006 / TEXT, CONVERSION AND JSON

Syntax

```
INT (number)
```

Converts a numeric value to an integer by truncation toward zero.

Return: INTEGER. **Arguments:** 1.

Main body

```
LOCAL rowId AS INTEGER
rowId = INT(VAL("123"))
? rowId
? TYPEOF(rowId)
```

Result / effect: 123, followed by INTEGER.

Truncates toward zero, rather than rounding: INT(7.9)=7 and INT(-7.9)=-7. Validate integral input before treating a fractional numeric value as a database identifier.

For large integer text, use INT(DECIMAL("9007199254740993")); a value already rounded by VAL/NUMBER cannot be recovered by INT. Results must fit the signed 64-bit INTEGER range.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

JSONDECODE

BI-013 / TEXT, CONVERSION AND JSON

Syntax

```
JSONDECODE (text)
```

Decodes JSON text into Kokum values.

Return: JSON. **Arguments:** 1.

Main body

```
LOCAL data AS JSON
data = JSONDECODE('{"count":2}')
```

```
? data["count"]
```

Result / effect: Prints 2.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

JSONENCODE

BI-012 / TEXT, CONVERSION AND JSON

Syntax

```
JSONENCODE (value)
```

Encodes supported Kokum values as JSON text. Native handles and cyclic graphs are not JSON values.

Return: STRING. **Arguments:** 1.

Main body

```
? JSONENCODE({"name": "Kokum", "active": .T.})
```

Result / effect: Prints a JSON object string.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

KCONNECT

BI-067 / REMOTE FLATDB

Syntax

```
KCONNECT(path) or KCONNECT USING path
```

Connects the Remote FlatDB client using a configuration-file path.

Return: LOGICAL. **Arguments:** 1.

Main body + FlatDB connection

```
KCONNECT USING kdb.conf
? KCONNECTED()
```

Result / effect: Prints .T. when the configured server accepts the connection.

Requires the Remote FlatDB setup. KEXECUTE and KQUERY take exactly one SQL string; their signatures have no parameter-array argument.

Verification: Executed with the local Kokum FlatDB server. Source: [S07]; current signature also checked against [S01].

KCONNECTED

BI-068 / REMOTE FLATDB

Syntax

```
KCONNECTED()
```

Reports whether the process-level Remote FlatDB client is connected.

Return: LOGICAL. **Arguments:** 0.

Main body + FlatDB connection

```
KCONNECT USING kdb.conf
? KCONNECTED()
KDISCONNECT()
```

Result / effect: Prints .T. after a successful authenticated connection.

Requires the Remote FlatDB setup. KEXECUTE and KQUERY take exactly one SQL string; their signatures have no parameter-array argument.

Verification: Executed with the local Kokum FlatDB server. Source: [S07]; current signature also checked against [S01].

KDISCONNECT

BI-071 / REMOTE FLATDB

Syntax

```
KDISCONNECT()
```

Closes the Remote FlatDB connection.

Return: LOGICAL. **Arguments:** 0.

Main body + FlatDB connection

```
KCONNECT USING kdb.conf
IF KCONNECTED()
    ? KDISCONNECT()
    ? KCONNECTED()
ENDIF
```

Result / effect: Prints .T. then .F. after a successful initial connection.

Requires the Remote FlatDB setup. KEXECUTE and KQUERY take exactly one SQL string; their signatures have no parameter-array argument.

Verification: Executed with the local Kokum FlatDB server. Source: [S07]; current signature also checked against [S01].

KEXECUTE

BI-069 / REMOTE FLATDB

Syntax

```
KEXECUTE(sql)
```

Executes a non-query SQL statement through the Remote FlatDB connection.

Return: INTEGER. **Arguments:** 1.

Main body + FlatDB connection

```
KCONNECT USING kdb.conf
IF KCONNECTED()
    KEXECUTE("CREATE TABLE IF NOT EXISTS items (id INTEGER)")
    ? KEXECUTE("INSERT INTO items(id) VALUES (1)")
    KDISCONNECT()
ENDIF
```

Result / effect: Prints 1 affected row after connecting.

Requires the Remote FlatDB setup. KEXECUTE and KQUERY take exactly one SQL string; their signatures have no parameter-array argument.

Verification: Executed with the local Kokum FlatDB server. Source: [S07]; current signature also checked against [S01].

KEYS

BI-024 / COLLECTIONS AND CODEBLOCKS

Syntax

```
KEYS (map)
```

Returns map keys in insertion order.

Return: ARRAY. **Arguments:** 1.

Main body

```
? JSONENCODE(KEYS({"a": 1, "b": 2}))
```

Result / effect: Prints ["a","b"].

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

KQUERY

BI-070 / REMOTE FLATDB

Syntax

```
KQUERY(sql)
```

Executes a query and returns an ARRAY of row MAPs.

Return: ARRAY. **Arguments:** 1.

Main body + FlatDB connection

```
KCONNECT USING kdb.conf
IF KCONNECTED()
  KEXECUTE("CREATE TABLE IF NOT EXISTS items (id INTEGER)")
  KEXECUTE("DELETE FROM items")
  KEXECUTE("INSERT INTO items(id) VALUES (7)")
  ? JSONENCODE(KQUERY("SELECT id FROM items"))
  KDISCONNECT()
ENDIF
```

Result / effect: Prints [{"id":7}]. Uses a disposable test database.

Requires the Remote FlatDB setup. KEXECUTE and KQUERY take exactly one SQL string; their signatures have no parameter-array argument.

Verification: Executed with the local Kokum FlatDB server. Source: [S07]; current signature also checked against [S01].

LEN

BI-002 / TEXT, CONVERSION AND JSON

Syntax

```
LEN(value)
```

Returns the length of a string, array, or map.

Return: INTEGER. **Arguments:** 1.

Main body

```
? LEN("Kokum"), LEN([10, 20, 30])
```

Result / effect: Prints 5 and 3.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

LOCK

BI-078 / TASKS AND SYNCHRONIZATION

Syntax

```
LOCK(mutex [, timeoutMs])
```

Function form: acquires a MUTEX, optionally with timeout, and returns .T..

Return: LOGICAL. **Arguments:** 1..2.

Main body

```
LOCAL gate AS MUTEX
gate = MUTEX()
? LOCK(gate, 1000)
UNLOCK(gate)
```

Result / effect: Prints .T. when the mutex is acquired.

Timeouts are milliseconds and have operation-specific outcomes. A wait timeout is not task cancellation. Never block the GUI owner indefinitely.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

LOG

BI-048 / MATHEMATICS AND STATISTICS

Syntax

```
LOG(number)
```

Returns the natural logarithm.

Return: NUMBER. **Arguments:** 1.

Main body

```
? ROUND(LOG(EXP(1)), 6)
```

Result / effect: Prints 1.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

LOG10

BI-049 / MATHEMATICS AND STATISTICS

Syntax

```
LOG10(number)
```

Returns the base-10 logarithm.

Return: NUMBER. **Arguments:** 1.

Main body

```
? LOG10(1000)
```

Result / effect: Prints 3.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

LOWER

BI-004 / TEXT, CONVERSION AND JSON

Syntax

```
LOWER(text)
```

Converts text to lowercase.

Return: STRING. **Arguments:** 1.

Main body

```
? LOWER("KOKUM")
```

Result / effect: Prints kokum.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

LTRIM

BI-110 / TEXT, CONVERSION AND JSON

Syntax

```
LTRIM(text AS STRING)
```

Return new text with supported ASCII whitespace removed from the left edge only.

Return: STRING. **Arguments:** 1.

Main body

```
? "[" + LTRIM(" Kokum ") + "]"
```

Result / effect: [Kokum].

Same six ASCII whitespace characters as TRIM; trailing and interior whitespace remain unchanged.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S12]; current signature also checked against [S01].

MAPREMOVE

BI-023 / COLLECTIONS AND CODEBLOCKS

Syntax

```
MAPREMOVE(map, key)
```

Removes a map key and returns its previous value or NULL.

Return: ANY. **Arguments:** 2.

Main body

```
LOCAL m AS MAP
m = {"id": 7}
? MAPREMOVE(m, "id"), HASKEY(m, "id")
```

Result / effect: Prints 7 and .F.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

MAPSET

BI-021 / COLLECTIONS AND CODEBLOCKS

Syntax

```
MAPSET(map, key, value)
```

Sets a map key and returns the stored value.

Return: ANY. **Arguments:** 3.

Main body

```
LOCAL m AS MAP
m = {}
MAPSET(m, "name", "Kokum")
? m["name"]
```

Result / effect: Prints Kokum.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

MAX

BI-015 / COLLECTIONS AND CODEBLOCKS

Syntax

```
MAX(values AS ARRAY)
```

Returns the largest element of one nonempty array. Elements must be mutually comparable.

Return: ANY. **Arguments:** 1.

Main body

```
? MAX([7, 3, 9])
```

Result / effect: Prints 9.

Pass one array, not multiple scalar arguments. Empty or incomparable arrays raise a catchable error.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

MEAN

BI-053 / MATHEMATICS AND STATISTICS

Syntax

```
MEAN(values)
```

Returns the arithmetic mean of numeric array values.

Return: ANY. **Arguments:** 1.

Main body

```
? MEAN([10, 20, 30])
```

Result / effect: Prints 20.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

MEDIAN

BI-055 / MATHEMATICS AND STATISTICS

Syntax

```
MEDIAN(values)
```

Returns the middle value or midpoint after sorting a copy.

Return: ANY. **Arguments:** 1.

Main body

```
? MEDIAN([4, 1, 3, 2])
```

Result / effect: Prints 2.5.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

Sorts a copy, not the caller array. Even-sized data uses the midpoint of the two central values.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

METHODS

BI-028 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
METHODS(object)
```

Returns the public method names of an object.

Return: ARRAY. **Arguments:** 1.

Main body + SET-REFLECTION

```
LOCAL p
p = NEW Person()
? JSONENCODE(METHODS(p))
```

Result / effect: Prints an array containing Greet.

Use the Person class from SET-REFLECTION.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

MIN

BI-014 / COLLECTIONS AND CODEBLOCKS

Syntax

```
MIN(values AS ARRAY)
```

Returns the smallest element of one nonempty array. Elements must be mutually comparable.

Return: ANY. **Arguments:** 1.

Main body

```
? MIN([7, 3, 9])
```

Result / effect: Prints 3.

Pass one array, not multiple scalar arguments. Empty or incomparable arrays raise a catchable error.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

MOD

BI-050 / MATHEMATICS AND STATISTICS

Syntax

```
MOD(dividend, divisor)
```

Returns the numeric remainder.

Return: ANY. **Arguments:** 2.

Main body

```
? MOD(17, 5)
```

Result / effect: Prints 2.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

MUTEX

BI-077 / TASKS AND SYNCHRONIZATION

Syntax

```
MUTEX()
```

Creates a MUTEX native resource.

Return: MUTEX. **Arguments:** 0.

Main body

```
LOCAL gate AS MUTEX
gate = MUTEX()
? TYPEOF(gate)
```

Result / effect: Prints MUTEX.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

MUTEXSTATUS

BI-080 / TASKS AND SYNCHRONIZATION

Syntax

```
MUTEXSTATUS(mutex)
```

Returns mutex ownership and contention diagnostics.

Return: MAP. **Arguments:** 1.

Main body

```
LOCAL gate AS MUTEX
gate = MUTEX()
? JSONENCODE(MUTEXSTATUS(gate))
```

Result / effect: Prints a status map.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

OPEN

BI-061 / FILE I/O

Syntax

```
OPEN(path, mode)
```

Opens a UTF-8 text file and returns a FILE resource.

Return: FILE. **Arguments:** 2.

Main body

```
LOCAL file AS FILE
file = OPEN("sample.txt", "w")
? TYPEOF(file)
CLOSEFILE(file)
```

Result / effect: Prints FILE and creates/truncates sample.txt.

Modes r, w, rw/r+, w+, a and a+ have distinct create/truncate/read/write rules. All examples write only files in a disposable working directory; w truncates an existing file.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

PERCENTILE

BI-058 / MATHEMATICS AND STATISTICS

Syntax

```
PERCENTILE(values, percentile)
```

Returns an R-7 linearly interpolated percentile from 0 through 100.

Return: ANY. **Arguments:** 2.

Main body

```
? PERCENTILE([1, 2, 3, 4], 25)
```

Result / effect: Prints 1.75.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

Percentile is between 0 and 100. Uses R-7 linear interpolation, not a nearest-item percentile.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

POW

BI-046 / MATHEMATICS AND STATISTICS

Syntax

```
POW(base, exponent)
```

Raises a number to a power.

Return: NUMBER. **Arguments:** 2.

Main body

```
? POW(2, 8)
```

Result / effect: Prints 256.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

PROPERTIES

BI-027 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
PROPERTIES(object)
```

Returns the public property names of an object.

Return: ARRAY. **Arguments:** 1.

Main body + SET-REFLECTION

```
LOCAL p
p = NEW Person()
? JSONENCODE(PROPERTIES(p))
```

Result / effect: Prints an array containing Name.

Use the Person class from SET-REFLECTION.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

RANGE

BI-060 / MATHEMATICS AND STATISTICS

Syntax

```
RANGE(values)
```

Returns maximum minus minimum for numeric array values.

Return: ANY. **Arguments:** 1.

Main body

```
? RANGE([10, 30, 20])
```

Result / effect: Prints 20.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

READ

BI-062 / FILE I/O

Syntax

```
READ(file [, byteCount])
```

Reads the remainder or a bounded byte count as UTF-8 text.

Return: STRING. **Arguments:** 1..2.

Main body

```
LOCAL file AS FILE
file = OPEN("sample.txt", "w")
WRITE(file, "Alpha\nBeta\n")
CLOSEFILE(file)
file = OPEN("sample.txt", "r")
? READ(file, 5)
CLOSEFILE(file)
```

Result / effect: Prints Alpha. READ counts bytes and advances the file position.

byteCount counts UTF-8 bytes, not characters. A returned chunk that splits an encoded character is invalid. This is text I/O, not a raw-byte interface.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

READLINE

BI-066 / FILE I/O

Syntax

```
READLINE(file)
```

Reads one line and advances the file pointer; returns NULL at EOF.

Return: ANY. **Arguments:** 1.

Main body

```
LOCAL file AS FILE
LOCAL line
file = OPEN("lines.txt", "w")
WRITE(file, "Alpha\n\nBeta\n")
CLOSEFILE(file)
file = OPEN("lines.txt", "r")
DO WHILE .T.
    line = READLINE(file)
    IF line = NULL
        EXIT
    ENDIF
    ? line
ENDDO
CLOSEFILE(file)
```

Result / effect: Prints Alpha, a blank line, then Beta. NULL signals EOF; an empty string is a real blank line.

No EOF() or SEEK() built-in exists in this catalogue. A read loop stops on READLINE() = NULL.

Distinguish a blank STRING from NULL. Do not write WHILE line as an EOF condition: a legitimate empty line must not stop the loop.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEXCOMPILE

BI-097 / REGULAR EXPRESSIONS

Syntax

```
REGEXCOMPILE(pattern [, flags])
```

Compiles a reusable REGEX resource. PATTERN is its annotation alias.

Return: REGEX. **Arguments:** 1..2.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[A-Z]+", "i")
? rx.IsMatch("kokum")
rx.Close()
```

Result / effect: Prints .T.

Compile once for repeated matching; close the resource after use. REGEX/PATTERN is a native resource, not a MAP.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEXESCAPE

BI-103 / REGULAR EXPRESSIONS

Syntax

```
REGEXESCAPE(text)
```

Escapes metacharacters so input is matched literally.

Return: STRING. **Arguments:** 1.

Main body

```
LOCAL pattern AS STRING
pattern = "^" + REGEXESCAPE("a+b") + "$"
? REGEXMATCH(pattern, "a+b")
```

Result / effect: Prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEXFIND

BI-099 / REGULAR EXPRESSIONS

Syntax

```
REGEXFIND(patternOrRegex, textOrFile [, flags])
```

Returns the first matched STRING, or NULL. Detailed capture maps are obtained through natural FIND ... WITH DETAILS, not this function.

Return: ANY. **Arguments:** 2..3.

Main body

```
LOCAL hit
hit = REGEXFIND("[0-9]+", "ID 42")
IF hit != NULL
    ? JSONENCODE(hit)
ENDIF
```

Result / effect: Prints the JSON string "42".

The result is not a match MAP. Test NULL before using it.

A FILE subject is consumed from its current position through EOF and remains open. Direct regex is bounded whole-subject work, not the streaming line FIND path. A compiled REGEX already owns its flags.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEXFINDALL

BI-100 / REGULAR EXPRESSIONS

Syntax

```
REGEXFINDALL(patternOrRegex, textOrFile [, flags])
```

Returns an ARRAY of matched strings in non-overlapping order, not match maps.

Return: ARRAY. **Arguments:** 2..3.

Main body

```
? LEN(REGEXFINDALL("[0-9]+", "A1 B22"))
```

Result / effect: Prints 2.

A FILE subject is consumed from its current position through EOF and remains open. Direct regex is bounded whole-subject work, not the streaming line FIND path. A compiled REGEX already owns its flags.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEXMATCH

BI-098 / REGULAR EXPRESSIONS

Syntax

```
REGEXMATCH(patternOrRegex, textOrFile [, flags])
```

Tests whether any match exists; anchors are needed for whole-string validation.

Return: LOGICAL. **Arguments:** 2..3.

Main body

```
? REGEXMATCH("[A-Z]{2}[0-9]{2}$", "AB12")
```

Result / effect: Prints .T.

A FILE subject is consumed from its current position through EOF and remains open. Direct regex is bounded whole-subject work, not the streaming line FIND path. A compiled REGEX already owns its flags.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEXREPLACE

BI-101 / REGULAR EXPRESSIONS

Syntax

```
REGEXREPLACE(patternOrRegex, textOrFile, replacement [, flags])
```

Returns replacement text; does not modify an input FILE.

Return: STRING. **Arguments:** 3..4.

Main body

```
? REGEXREPLACE("[0-9]+", "A12 B34", "#")
```

Result / effect: Prints A# B#.

A FILE subject is consumed from its current position through EOF and remains open. Direct regex is bounded whole-subject work, not the streaming line FIND path. A compiled REGEX already owns its flags.

Returns transformed text. It never overwrites an input file; write the returned string explicitly.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGXSPLIT

BI-102 / REGULAR EXPRESSIONS

Syntax

```
REGXSPLIT(patternOrRegex, textOrFile [, flags])
```

Splits the subject around regular-expression delimiters.

Return: ARRAY. **Arguments:** 2..3.

Main body

```
? JSONENCODE(REGXSPLIT("[,;]", "one,two;three"))
```

Result / effect: Prints ["one","two","three"].

A FILE subject is consumed from its current position through EOF and remains open. Direct regex is bounded whole-subject work, not the streaming line FIND path. A compiled REGEX already owns its flags.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

ROUND

BI-040 / MATHEMATICS AND STATISTICS

Syntax

```
ROUND(number [, places])
```

Rounds a number, optionally to a decimal-place count.

Return: ANY. **Arguments:** 1..2.

Main body

```
? ROUND(12.345, 2)
```

Result / effect: Prints 12.35.

An integral-looking result need not have INTEGER type. These functions preserve the numeric family where practical; use INT explicitly when an INTEGER is required.

places is an integral count from -18 through 18. Positive values address decimal places; negative values address powers of ten. ROUND uses halfway-away-from-zero; TRUNC truncates toward zero.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

RTRIM

BI-111 / TEXT, CONVERSION AND JSON

Syntax

```
RTRIM(text AS STRING)
```

Return new text with supported ASCII whitespace removed from the right edge only.

Return: STRING. **Arguments:** 1.

Main body

```
? "[" + RTRIM(" Kokum ") + "]"
```

Result / effect: [Kokum].

Same six ASCII whitespace characters as TRIM; leading and interior whitespace remain unchanged. RTREM is not an alias.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S12]; current signature also checked against [S01].

SETPROPERTY

BI-032 / REFLECTION AND RUNTIME DIAGNOSTICS

Syntax

```
SETPROPERTY(object, name, value)
```

Writes an object property by name.

Return: ANY. **Arguments:** 3.

Main body + SET-REFLECTION

```
LOCAL p
p = NEW Person()
SETPROPERTY(p, "Name", "Kokum")
? p.Name
```

Result / effect: Prints Kokum.

Use the Person class from SET-REFLECTION.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

SHOWFORM

BI-035 / FORMS

Syntax

```
SHOWFORM(formName)
```

Shows a compiled form by identifier.

Return: NULL. No useful return value is required. **Arguments:** 1.

Slot body + SET-FORMS

```
SHOWFORM("CustomerView")
```

Result / effect: CustomerView becomes visible.

Verification: Executed in a compiled Linux slot; setup required. Source: [S01]; current signature also checked against [S01].

SIGN

BI-039 / MATHEMATICS AND STATISTICS

Syntax

```
SIGN(number)
```

Returns -1, 0, or 1 according to a number's sign.

Return: INTEGER. **Arguments:** 1.

Main body

```
? SIGN(-5), SIGN(0), SIGN(8)
```

Result / effect: Prints -1, 0, and 1.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

SLEEP

BI-038 / TASKS AND SYNCHRONIZATION

Syntax

```
SLEEP(milliseconds)
```

Blocks the current execution for the requested milliseconds.

Return: NULL. No useful return value is required. **Arguments:** 1.

Main body

```
? "before"
SLEEP(50)
? "after"
```

Result / effect: Prints before, pauses briefly, then prints after.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

SQRT

BI-045 / MATHEMATICS AND STATISTICS

Syntax

```
SQRT (number)
```

Returns the square root.

Return: NUMBER. **Arguments:** 1.

Main body

```
? SQRT(81)
```

Result / effect: Prints 9.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

STDDEV

BI-057 / MATHEMATICS AND STATISTICS

Syntax

```
STDDEV(values [, sample])
```

Returns the square root of the corresponding variance.

Return: ANY. **Arguments:** 1..2.

Main body

```
? STDDEV([2, 4, 4, 4, 5, 5, 7, 9])
```

Result / effect: Prints 2.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

sample defaults to .F. (population). Pass .T. for sample variance/deviation; fewer than two numeric sample values returns NULL.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

STR

BI-001 / TEXT, CONVERSION AND JSON

Syntax

```
STR(value [, width [, decimals]])
```

Converts a value to text; optional width/decimals format numeric output.

Return: STRING. **Arguments:** 1..3.

Main body

```
? STR(125)
```

Result / effect: Prints 125.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

SUM

BI-052 / MATHEMATICS AND STATISTICS

Syntax

```
SUM(values)
```

Returns the sum of numeric array values while ignoring NULL.

Return: ANY. **Arguments:** 1.

Main body

```
? SUM([10, NULL, 20, 30])
```

Result / effect: Prints 60.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

SUM of an empty/all-NULL numeric collection is zero. The input array is not sorted or changed.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

SYNCSTATS

BI-096 / TASKS AND SYNCHRONIZATION

Syntax

```
SYNCSTATS([resource])
```

Returns synchronization diagnostics globally or for one sync resource.

Return: MAP. **Arguments:** 0..1.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(0)
? TYPEOF(SYNCSTATS(counter))
```

Result / effect: Prints MAP.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

THREADDONE

BI-073 / TASKS AND SYNCHRONIZATION

Syntax

```
THREADDONE(task)
```

Tests whether a named task has finished.

Return: LOGICAL. **Arguments:** 1.

Main body + SET-THREAD

```
RUN SlowWork() AS worker NOWAIT
? THREADDONE(worker)
WAIT FOR worker
? THREADDONE(worker)
```

Result / effect: The final result is .T.

Use SlowWork from SET-THREAD; the final check follows WAIT FOR.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

THREADERROR

BI-074 / TASKS AND SYNCHRONIZATION

Syntax

```
THREADERROR(task)
```

Returns a named task error message, or an empty string when none exists.

Return: STRING. **Arguments:** 1.

Main body + SET-THREAD

```
RUN Work() AS worker NOWAIT
WAIT FOR worker
? THREADERROR(worker)
```

Result / effect: Prints an empty string after successful work.

Use Work from SET-THREAD. An empty message means no task fault.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

THREADID

BI-075 / TASKS AND SYNCHRONIZATION

Syntax

```
THREADID(task)
```

Returns Kokum's worker-thread identifier for the named task.

Return: INTEGER. **Arguments:** 1.

Main body + SET-THREAD

```
RUN Work() AS worker NOWAIT
WAIT FOR worker
? THREADID(worker)
```

Result / effect: Prints a positive worker identifier after the task has run.

Use Work from SET-THREAD. This is a Kokum platform thread identifier, not a unique task ID; tasks can reuse a worker.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

THREADPOOLSIZE

BI-076 / TASKS AND SYNCHRONIZATION

Syntax

```
THREADPOOLSIZE()
```

Returns the configured bounded worker-pool size.

Return: INTEGER. **Arguments:** 0.

Main body

```
? THREADPOOLSIZE()
```

Result / effect: Prints a positive integer.

All tasks in the process share the configured bounded worker pool.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

THREADSTATUS

BI-072 / TASKS AND SYNCHRONIZATION

Syntax

```
THREADSTATUS(task)
```

Returns the named task status string.

Return: STRING. **Arguments:** 1.

Main body + SET-THREAD

```
RUN SlowWork() AS worker NOWAIT
? THREADSTATUS(worker)
WAIT FOR worker
```

Result / effect: Prints a state such as QUEUED, RUNNING or COMPLETED; the first state depends on scheduling.

Use SlowWork from SET-THREAD. THREADSTATUS uses uppercase state names; task.Status uses lowercase names.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

TOTP

BI-108 / HTTP, REST AND TOTP

Syntax

```
TOTP(secret AS STRING [, options AS MAP])
```

Generate a zero-padded time-based one-time password STRING from a Base32 secret.

Return: STRING. **Arguments:** 1..2.

Main body

```
? TOTP( ;
  "GEZDGNBVG3TQ0JQGEZDGNBVG3TQ0JQ", ;
  {"Timestamp":59, "Digits":8})
```

Result / effect: 94287082.

Public test secret only. Real enrollment secrets and OTPs must not be stored in source, examples, screenshots or persistent logs.

Defaults: SHA1, 6 digits, 30 seconds and StartTime=0. Algorithm, Digits, PeriodSeconds, Timestamp and StartTime are the supported option keys. Numeric options must be INTEGER.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S11]; current signature also checked against [S01].

TRIM

BI-109 / TEXT, CONVERSION AND JSON

Syntax

```
TRIM(text AS STRING)
```

Return new text with supported ASCII whitespace removed from both ends.

Return: STRING. **Arguments:** 1.

Main body

```
LOCAL original AS STRING
original = " Kokum "
? "[" + TRIM(original) + "]"
? "[" + original + "]"
```

Result / effect: [Kokum], then [Kokum].

Removes space, tab, LF, vertical tab, form feed and CR at the selected edge. Interior whitespace and non-ASCII spacing characters are preserved.

Exactly one STRING argument; NULL and numeric values are errors. Does not mutate the input.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S12]; current signature also checked against [S01].

TRUNC

BI-044 / MATHEMATICS AND STATISTICS

Syntax

```
TRUNC(number [, places])
```

Truncates a number, optionally to a decimal-place count.

Return: ANY. **Arguments:** 1..2.

Main body

```
? TRUNC(12.987, 2)
```

Result / effect: Prints 12.98.

An integral-looking result need not have INTEGER type. These functions preserve the numeric family where practical; use INT explicitly when an INTEGER is required.

places is an integral count from -18 through 18. Positive values address decimal places; negative values address powers of ten. ROUND uses halfway-away-from-zero; TRUNC truncates toward zero.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

TYPEOF

BI-009 / TEXT, CONVERSION AND JSON

Syntax

```
TYPEOF(value)
```

Returns the runtime type name.

Return: STRING. **Arguments:** 1.

Main body

```
? TYPEOF({"id": 1})
```

Result / effect: Prints MAP.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

UNLOCK

BI-079 / TASKS AND SYNCHRONIZATION

Syntax

```
UNLOCK(mutex)
```

Function form: releases a MUTEX owned by the current execution.

Return: LOGICAL. **Arguments:** 1.

Main body

```
LOCAL gate AS MUTEX
gate = MUTEX()
LOCK(gate)
? UNLOCK(gate)
```

Result / effect: Prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

UPPER

BI-003 / TEXT, CONVERSION AND JSON

Syntax

```
UPPER(text)
```

Converts text to uppercase.

Return: STRING. **Arguments:** 1.

Main body

```
? UPPER("Kokum")
```

Result / effect: Prints KOKUM.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

URLGET

BI-105 / HTTP, REST AND TOTP

Syntax

```
URLGET(url [, options])
```

Performs an HTTP or HTTPS GET and returns a response MAP.

Return: MAP. **Arguments:** 1..2.

Main body + local HTTP service

```
LOCAL response AS MAP
response = URLGET("http://127.0.0.1:18080/hello")
? response["Status"], response["Json"]["message"]
```

Result / effect: Prints 200 and Hello from Kokum.

Start the complete Kokum test service from the HTTP chapter first. HTTP 4xx/5xx are response maps, not transport exceptions.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S10]; current signature also checked against [S01].

URLPOST

BI-106 / HTTP, REST AND TOTP

Syntax

```
URLPOST(url, body [, options])
```

Posts text, JSON, an encoded form or an empty body. MAP/ARRAY bodies default to JSON.

Return: MAP. **Arguments:** 2..3.

Main body + local HTTP service

```
LOCAL response AS MAP
response = URLPOST( ;
  "http://127.0.0.1:18080/echo", {"name":"Anil"})
? response["Json"]["name"]
```

Result / effect: Prints Anil.

Start the complete Kokum test service from the HTTP chapter first. HTTP 4xx/5xx are response maps, not transport exceptions.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S10]; current signature also checked against [S01].

URLREQUEST

BI-107 / HTTP, REST AND TOTP

Syntax

```
URLREQUEST(method, url, body [, options])
```

Execute a general HTTP/HTTPS request and return its response MAP. Supported methods are GET, HEAD, OPTIONS, POST, PUT, PATCH and DELETE.

Return: MAP. **Arguments:** 3..4.

Main body + local HTTP service

```
LOCAL response AS MAP
response = URLREQUEST( ;
  "PUT", "http://127.0.0.1:18080/echo", ;
  {"name":"Kokum"})
? response["Status"]
? response["Json"]["name"]
```

Result / effect: 200, followed by Kokum.

The third argument is mandatory. Use NULL for GET, HEAD and OPTIONS; they do not accept content. Options must be a MAP when supplied.

FollowRedirects defaults to .F. for URLREQUEST; URLGET/URLPOST default to .T. HTTP failure status is a normal response, not a transport exception.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S10]; current signature also checked against [S01].

VAL

BI-007 / TEXT, CONVERSION AND JSON

Syntax

```
VAL(text)
```

Parses numeric text as a NUMBER.

Return: NUMBER. **Arguments:** 1.

Main body

```
? VAL("12.5") + 1
```

Result / effect: Prints 13.5.

The result remains NUMBER even when the text is "123". Use INT(VAL(text)) for a bounded numeric-to-integer conversion, or INT(DECIMAL(text)) when exact large integer text must be preserved.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

VALUES

BI-025 / COLLECTIONS AND CODEBLOCKS

Syntax

```
VALUES(map)
```

Returns map values in insertion order.

Return: ARRAY. **Arguments:** 1.

Main body

```
? JSONENCODE(VALUES({"a": 1, "b": 2}))
```

Result / effect: Prints [1,2].

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S01]; current signature also checked against [S01].

VARIANCE

BI-056 / MATHEMATICS AND STATISTICS

Syntax

```
VARIANCE(values [, sample])
```

Returns population variance, or sample variance when sample is .T..

Return: ANY. **Arguments:** 1..2.

Main body

```
? VARIANCE([2, 4, 4, 4, 5, 5, 7, 9])
```

Result / effect: Prints 4.

Input must be one numeric ARRAY. NULL elements are ignored; other nonnumeric elements are rejected. Empty/insufficient data follows the function-specific contract; do not assume every statistic returns zero.

sample defaults to .F. (population). Pass .T. for sample variance/deviation; fewer than two numeric sample values returns NULL.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S04]; current signature also checked against [S01].

WRITE

BI-063 / FILE I/O

Syntax

```
WRITE(file, text)
```

Writes UTF-8 text at the current stream position and returns bytes written.

Return: INTEGER. **Arguments:** 2.

Main body

```
LOCAL file AS FILE  
file = OPEN("sample.txt", "w")  
? WRITE(file, "Kokum\n")  
CLOSEFILE(file)
```

Result / effect: Writes Kokum plus newline and prints the byte count.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

Native resource methods A-Z

112 methods, qualified by resource type. A property has no call parentheses; a zero-argument method still does. REGEX.Pattern(), Flags() and IsClosed() are methods. Resource instances require their stated setup. The method name alone does not create the resource.

ATOMICINTEGER methods

ATOMICINTEGER.Add

API-092 / ATOMICINTEGER

Syntax

```
Add(delta AS INTEGER)
```

Add a delta and return the new value.

Return: INTEGER. **Arguments:** see signature.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? counter.Add(5)
```

Result / effect: Adds 5 and prints the new value.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINTEGER.CompareExchange

API-095 / ATOMICINTEGER

Syntax

```
CompareExchange(expected AS INTEGER, desired AS INTEGER)
```

Replace the value only when it equals expected.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? counter.CompareExchange(5, 9)
```

Result / effect: Prints .T. and stores 9 only when current value is 5.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINTEGER.Decrement

API-094 / ATOMICINTEGER

Syntax

```
Decrement()
```

Decrement and return the new value.

Return: INTEGER. **Arguments:** see signature.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? counter.Decrement()
```

Result / effect: Decrements and prints the new value.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINTEGER.Exchange

API-091 / ATOMICINTEGER

Syntax

```
Exchange(value AS INTEGER)
```

Atomically replace the value and return the previous value.

Return: INTEGER. **Arguments:** see signature.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? counter.Exchange(20)
```

Result / effect: Prints the previous value and stores 20.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINTEGER.Get

API-089 / ATOMICINTEGER

Syntax

```
Get()
```

Read the current value.

Return: INTEGER. **Arguments:** see signature.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? counter.Get()
```

Result / effect: Prints the current atomic value.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINTEGER.Increment

API-093 / ATOMICINTEGER

Syntax

```
Increment()
```

Increment and return the new value.

Return: INTEGER. **Arguments:** see signature.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? counter.Increment()
```

Result / effect: Increments and prints the new value.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINTEGER.Set

API-090 / ATOMICINTEGER

Syntax

```
Set(value AS INTEGER)
```

Set a value and return the previous value.

Return: INTEGER. **Arguments:** see signature.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? counter.Set(10)
```

Result / effect: Prints the previous value and stores 10.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

ATOMICINTEGER.Status

API-096 / ATOMICINTEGER

Syntax

```
Status()
```

Return value and operation diagnostics.

Return: MAP. **Arguments:** see signature.

Main body

```
LOCAL counter AS ATOMICINTEGER
counter = ATOMICINTEGER(5)
? JSONENCODE(counter.Status())
```

Result / effect: Prints value and operation diagnostics.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNEL methods

CHANNEL.Close

API-101 / CHANNEL

Syntax

```
Close()
```

Close the channel and wake blocked senders and receivers.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
? queue.Close()
```

Result / effect: Closes the channel and prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNEL.Receive

API-099 / CHANNEL

Syntax

```
Receive([timeoutMs AS INTEGER])
```

Receive one message, optionally with a timeout.

Return: ANY. **Arguments:** see signature.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
queue.Send(42)
? queue.Receive(1000)
```

Result / effect: Prints one received value, NULL after closed/drained, or times out.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNEL.Send

API-097 / CHANNEL

Syntax

```
Send(value AS ANY [, timeoutMs AS INTEGER])
```

Send one copied message, waiting for capacity when needed.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
? queue.Send("hello", 1000)
```

Result / effect: Copies and sends hello; prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNEL.Status

API-102 / CHANNEL

Syntax

```
Status()
```

Return capacity, queue depth, waiters, and counters.

Return: MAP. **Arguments:** see signature.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
? JSONENCODE(queue.Status())
```

Result / effect: Prints capacity, depth, waiters, and counters.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNEL.TryReceive

API-100 / CHANNEL

Syntax

```
TryReceive()
```

Return {Received, Value} without waiting.

Return: MAP. **Arguments:** see signature.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
queue.Send("hello")
? JSONENCODE(queue.TryReceive())
```

Result / effect: Prints a map with Received and Value.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

CHANNEL.TrySend

API-098 / CHANNEL

Syntax

```
TrySend(value AS ANY)
```

Attempt an immediate send.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL queue AS CHANNEL
queue = CHANNEL(2)
? queue.TrySend("hello")
```

Result / effect: Attempts an immediate send; prints .T./.F.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

DATABASE methods

DATABASE.Begin

API-003 / DATABASE

Syntax

```
Begin()
```

Begin a transaction.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-DATABASE

```
dbLocal.Begin()
? "transaction started"
dbLocal.Rollback()
```

Result / effect: Starts then rolls back an otherwise empty transaction.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.Close

API-002 / DATABASE

Syntax

```
Close()
```

Close the database connection.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-DATABASE

```
dbLocal.Close()  
? dbLocal.IsOpen
```

Result / effect: Prints .F.; repeated Close() is safe.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.Commit

API-004 / DATABASE

Syntax

```
Commit()
```

Commit the active transaction.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-DATABASE

```
dbLocal.Begin()  
dbLocal.Execute("INSERT INTO log(message) VALUES (?)", ["ok"])  
dbLocal.Commit()
```

Result / effect: Commits the inserted row.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.CreateTableFromJson

API-009 / DATABASE

Syntax

```
CreateTableFromJson(tableName AS STRING, source AS JSON [, schemaOnly AS LOGICAL])
```

Create a table from JSON and return the inserted row count.

Return: INTEGER. **Arguments:** see signature.

Slot body + SET-DATABASE

```
dbLocal.Execute("DROP TABLE IF EXISTS imported_people")  
? dbLocal.CreateTableFromJson(  
  "imported_people", [{"name": "Anil", "age": 55}], .F.)
```

Result / effect: Creates imported_people and prints 1. Use a fresh table name.

This example deliberately recreates a sample table; use only the disposable lab database.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.Execute

API-010 / DATABASE

Syntax

```
Execute(sql AS STRING [, parameters AS ARRAY])
```

Execute parameterized SQL.

Return: INTEGER. **Arguments:** see signature.

Slot body + SET-DATABASE

```
? dbLocal.Execute("INSERT INTO log(message) VALUES (?)", ["Anil"])
```

Result / effect: Prints 1. User data is supplied through a parameter array.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.Open

API-001 / DATABASE

Syntax

```
Open()
```

Open the configured database connection.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-DATABASE

```
? dbLocal.Open()
```

Result / effect: Prints .T. and opens the configured connection.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.Ping

API-006 / DATABASE

Syntax

```
Ping()
```

Ping a TigerDB or PostgreSQL server.

Return: ANY. **Arguments:** see signature.

Slot body + SET-REMOTE-DATABASE

```
? JSONENCODE(dbTiger.Ping())
```

Result / effect: Prints the TigerDB/PostgreSQL ping response.

Requires a configured TigerDB or PostgreSQL server; do not use the SQLite dbLocal setup for this call.

Verification: Compiler checked; external server acceptance not executed. Source: [S07]; current signature also checked against [S01].

DATABASE.Query

API-011 / DATABASE

Syntax

```
Query(sql AS STRING [, parameters AS ARRAY])
```

Return all result rows as maps.

Return: ARRAY. **Arguments:** see signature.

Slot body + SET-DATABASE

```
LOCAL rows AS ARRAY
rows = dbLocal.Query("SELECT * FROM people ORDER BY id")
? TYPEOF(rows), LEN(rows)
```

Result / effect: Prints ARRAY and the row count.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.QueryOne

API-012 / DATABASE

Syntax

```
QueryOne(sql AS STRING [, parameters AS ARRAY])
```

Return the first row or NULL.

Return: MAP. **Arguments:** see signature.

Slot body + SET-DATABASE

```
LOCAL row
row = dbLocal.QueryOne("SELECT name FROM people WHERE id = ?", [1])
IF row != NULL
  ? row["name"]
ENDIF
```

Result / effect: Prints Anil when id 1 exists. Checks for NULL before reading a field.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.QueryTable

API-014 / DATABASE

Syntax

```
QueryTable(sql AS STRING [, parameters AS ARRAY [, headers AS LOGICAL]])
```

Return a two-dimensional TableView-compatible array.

Return: ARRAY. **Arguments:** see signature.

Slot body + SET-DATABASE

```
LOCAL tableData AS ARRAY
tableData = dbLocal.QueryTable( ;
  "SELECT id, name FROM people", NULL, .T.)
? JSONENCODE(tableData)
```

Result / effect: Returns a two-dimensional array including its header row.

Only configured local SQLite examples were executed for this book. Ping, UseDatabase and Raw entries require an external server and operator-supplied connection settings. DATABASE.Raw takes a JSON STRING, not a MAP.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.QueryValue

API-013 / DATABASE

Syntax

```
QueryValue(sql AS STRING [, parameters AS ARRAY])
```

Return the first column of the first row.

Return: ANY. **Arguments:** see signature.

Slot body + SET-DATABASE

```
? dbLocal.QueryValue("SELECT COUNT(*) FROM people")
```

Result / effect: Prints the scalar count.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.Raw

API-008 / DATABASE

Syntax

```
Raw(request AS STRING)
```

Send a raw TigerDB protocol request.

Return: ANY. **Arguments:** see signature.

Slot body + SET-REMOTE-DATABASE

```
? JSONENCODE(dbTiger.Raw(JSONENCODE({"action": "ping"})))
```

Result / effect: Prints the raw TigerDB protocol response.

TigerDB-specific. Raw takes a JSON STRING, so encode a MAP first. It is not a provider-neutral arbitrary-call API.

Verification: Compiler checked; external server acceptance not executed. Source: [S07]; current signature also checked against [S01].

DATABASE.Rollback

API-005 / DATABASE

Syntax

```
Rollback()
```

Roll back the active transaction.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-DATABASE

```
dbLocal.Begin()  
dbLocal.Execute("INSERT INTO log(message) VALUES (?)", ["cancelled"])  
dbLocal.Rollback()
```

Result / effect: The inserted row is not committed.

Verification: Executed in a compiled Linux slot; setup required. Source: [S07]; current signature also checked against [S01].

DATABASE.UseDatabase

API-007 / DATABASE

Syntax

```
UseDatabase(name AS STRING)
```

Select a TigerDB database.

Return: ANY. **Arguments:** see signature.

Slot body + SET-REMOTE-DATABASE

```
? JSONENCOD(dbTiger.UseDatabase("sales"))
```

Result / effect: Selects the TigerDB sales database.

TigerDB-specific; use an existing permitted database name.

Verification: Compiler checked; external server acceptance not executed. Source: [S07]; current signature also checked against [S01].

GRAPH methods

GRAPH.AddCandle

API-031 / GRAPH

Syntax

```
AddCandle(label AS ANY, open AS NUMBER, high AS NUMBER, low AS NUMBER, close AS NUMBER)
```

Append one candlestick record.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetType("candle")
graphSales.AddCandle("02 Sep", 108, 115, 103, 105)
```

Result / effect: Appends one OHLC candle.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.AddPoint

API-027 / GRAPH

Syntax

```
AddPoint(series AS STRING, label AS ANY, value AS NUMBER)
```

Append one point to a series.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.AddPoint("Sales", "Apr", 35)
```

Result / effect: Appends label Apr and value 35.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.AddSeries

API-025 / GRAPH

Syntax

```
AddSeries(name AS STRING, values AS ARRAY [, color AS STRING])
```

Add a named series.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.AddSeries("Target", [12,18,25])
```

Result / effect: Loads the Target series values.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.AddSlice

API-029 / GRAPH

Syntax

```
AddSlice(label AS ANY, value AS NUMBER)
```

Append one pie slice.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetType("pie")
graphSales.SetPie(["A","B"], [60,40])
graphSales.AddSlice("Cash", 10)
```

Result / effect: Appends one pie slice.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.Clear

API-015 / GRAPH

Syntax

```
Clear()
```

Remove all graph data.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.Clear()
```

Result / effect: All labels, series, and candle data are removed.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.ClearData

API-016 / GRAPH

Syntax

```
ClearData()
```

Remove labels, series, and candle data.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.ClearData()
```

Result / effect: Same data-clearing result as Clear().

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.Refresh

API-017 / GRAPH

Syntax

```
Refresh()
```

Force a browser redraw.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.Refresh()
```

Result / effect: The browser redraws the graph.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.RemoveSeries

API-026 / GRAPH

Syntax

```
RemoveSeries(name AS STRING)
```

Remove a named series.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-GRAPH

```
? graphSales.RemoveSeries("Target")
```

Result / effect: Prints .T. when Target existed and was removed.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetAnimation

API-037 / GRAPH

Syntax

```
SetAnimation(enabled AS LOGICAL)
```

Enable or disable redraw animation.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetAnimation(.T.)
```

Result / effect: Enables redraw animation.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetCandles

API-030 / GRAPH

Syntax

```
SetCandles(candles AS ARRAY)
```

Replace candlestick data.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetType("candle")
graphSales.SetCandles([["01 Sep",102,111,98,108]])
```

Result / effect: Loads one OHLC candle.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetData

API-021 / GRAPH

Syntax

```
SetData(labels AS ARRAY, values AS ARRAY [, seriesName AS STRING [, color AS STRING]])
```

Replace labels and one complete data series; candle graphs accept SetData(candles).

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetData(["Q1","Q2"], [120,150], "Sales")
```

Result / effect: Replaces labels and one complete series.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetEmptyText

API-046 / GRAPH

Syntax

```
SetEmptyText(text AS STRING)
```

Set the message shown when no data exists.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetEmptyText("No data available")
```

Result / effect: Shows this message when no data exists.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetGraphType

API-019 / GRAPH

Syntax

```
SetGraphType(type AS STRING)
```

Set line, bar, pie, or candle mode.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetGraphType("bar")
```

Result / effect: Graph mode becomes bar.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetGrid

API-035 / GRAPH

Syntax

```
SetGrid(visible AS LOGICAL)
```

Show or hide Cartesian grid lines.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetGrid(.T.)
```

Result / effect: Shows Cartesian grid lines.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetLabels

API-023 / GRAPH

Syntax

```
SetLabels(labels AS ARRAY)
```

Replace graph labels.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetLabels(["Jan", "Feb", "Mar"])
```

Result / effect: Graph labels are replaced.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetLegend

API-033 / GRAPH

Syntax

```
SetLegend(visible AS LOGICAL)
```

Show or hide the graph legend.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetLegend(.T.)
```

Result / effect: Shows the legend.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetLineWidth

API-040 / GRAPH

Syntax

```
SetLineWidth(width AS NUMBER)
```

Set line and candle stroke width.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetLineWidth(2.5)
```

Result / effect: Sets stroke width to 2.5.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetMaximumPoints

API-042 / GRAPH

Syntax

```
SetMaximumPoints(count AS INTEGER)
```

Set the maximum retained point count.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetMaximumPoints(100)
```

Result / effect: Alias of SetMaxPoints.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetMaxPoints

API-041 / GRAPH

Syntax

```
SetMaxPoints(count AS INTEGER)
```

Set the maximum retained point count.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetMaxPoints(100)
```

Result / effect: Retains at most 100 points.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetOption

API-032 / GRAPH

Syntax

```
SetOption(name AS STRING, value AS ANY)
```

Change one graph option.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetOption("showLegend", .T.)
```

Result / effect: The legend option becomes .T.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetPalette

API-045 / GRAPH

Syntax

```
SetPalette(palette AS STRING)
```

Select theme, classic, cool, warm, market, or monochrome.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetPalette("cool")
```

Result / effect: Uses the cool palette.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetPie

API-028 / GRAPH

Syntax

```
SetPie(labels AS ARRAY, values AS ARRAY)
```

Replace pie-chart data.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetType("pie")
graphSales.SetPie(["A", "B"], [60, 40])
```

Result / effect: Displays two pie slices.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetSeries

API-024 / GRAPH

Syntax

```
SetSeries(series AS ARRAY)
```

Replace all graph series.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetSeries([{"name": "Sales", "values": [10, 20, 30]}])
```

Result / effect: All graph series are replaced.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetSmooth

API-038 / GRAPH

Syntax

```
SetSmooth(enabled AS LOGICAL)
```

Enable or disable smoothed line segments.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetSmooth(.T.)
```

Result / effect: Uses smoothed line segments.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetStacked

API-039 / GRAPH

Syntax

```
SetStacked(enabled AS LOGICAL)
```

Enable or disable stacked bars.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetStacked(.T.)
```

Result / effect: Enables stacking for bar-mode rendering.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetTitle

API-020 / GRAPH

Syntax

```
SetTitle(title AS STRING)
```

Set the graph title.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetTitle("Quarterly Sales")
```

Result / effect: The title is displayed above the graph.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetType

API-018 / GRAPH

Syntax

```
SetType(type AS STRING)
```

Set line, bar, pie, or candle mode.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetType("line")
```

Result / effect: Graph mode becomes line.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetXAxisTitle

API-043 / GRAPH

Syntax

```
SetXAxisTitle(title AS STRING)
```

Set the horizontal-axis title.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetXAxisTitle("Quarter")
```

Result / effect: Displays the horizontal-axis title.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SetYAxisTitle

API-044 / GRAPH

Syntax

```
SetYAxisTitle(title AS STRING)
```

Set the vertical-axis title.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SetYAxisTitle("Revenue")
```

Result / effect: Displays the vertical-axis title.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.ShowGrid

API-036 / GRAPH

Syntax

```
ShowGrid(visible AS LOGICAL)
```

Show or hide Cartesian grid lines.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.ShowGrid(.F.)
```

Result / effect: Hides Cartesian grid lines.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.ShowLegend

API-034 / GRAPH

Syntax

```
ShowLegend(visible AS LOGICAL)
```

Show or hide the graph legend.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.ShowLegend(.F.)
```

Result / effect: Hides the legend.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.Snapshot

API-047 / GRAPH

Syntax

```
Snapshot()
```

Return a copy of the current graph state.

Return: MAP. **Arguments:** see signature.

Slot body + SET-GRAPH

```
LOCAL state AS MAP
state = graphSales.Snapshot()
? state["type"], state["title"]
```

Result / effect: Prints the current graph type and title.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

GRAPH.SupplyData

API-022 / GRAPH

Syntax

```
SupplyData(labels AS ARRAY, values AS ARRAY [, seriesName AS STRING [, color AS STRING]])
```

Alias for SetData.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-GRAPH

```
graphSales.SupplyData(["Q1", "Q2"], [120, 150], "Sales")
```

Result / effect: Alias of SetData; the series is loaded.

Verification: Executed in a compiled Linux slot; setup required. Source: [S08]; current signature also checked against [S01].

IMAGE methods

IMAGE.Clear

API-049 / IMAGE

Syntax

```
Clear()
```

Remove the current dynamic image.

Return: VOID. No useful return value is required. **Arguments:** see signature.

Slot body + SET-IMAGE

```
imgLogo.Clear()
```

Result / effect: Removes the current dynamic image; a configured static source may be restored.

IMAGE is the generated native resource (imgLogo), not a generic form MAP. Dynamic pixel refresh has a recorded GUI limitation; successful native loading is not proof of every rendered update.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

IMAGE.SetImage

API-048 / IMAGE

Syntax

```
SetImage(path AS STRING, keepAspectRatio AS LOGICAL)
```

Load a JPG, JPEG, or PNG image.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-IMAGE

```
? imgLogo.SetImage("images/kokum.png", .T.)
```

Result / effect: Prints .T. and displays the PNG while preserving aspect ratio.

Copy a real PNG to images/kokum.png in the application runtime working directory. For portable designed imagery, use a web/images resource and Source="images/kokum.png" instead.

IMAGE is the generated native resource (imgLogo), not a generic form MAP. Dynamic pixel refresh has a recorded GUI limitation; successful native loading is not proof of every rendered update.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

IMAGE.Snapshot

API-050 / IMAGE

Syntax

```
Snapshot()
```

Return the current Image state.

Return: MAP. **Arguments:** see signature.

Slot body + SET-IMAGE

```
? JSONENCODE(imgLogo.Snapshot())
```

Result / effect: Prints a portable Image state map.

IMAGE is the generated native resource (imgLogo), not a generic form MAP. Dynamic pixel refresh has a recorded GUI limitation; successful native loading is not proof of every rendered update.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

MUTEX methods

MUTEX.Lock

API-085 / MUTEX

Syntax

```
Lock([timeoutMs AS INTEGER])
```

Acquire the mutex, optionally with a timeout.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL gate AS MUTEX
gate = MUTEX()
? gate.Lock(1000)
gate.Unlock()
```

Result / effect: Prints .T. when gate is acquired.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

MUTEX.Status

API-088 / MUTEX

Syntax

```
Status()
```

Return mutex ownership and contention diagnostics.

Return: MAP. **Arguments:** see signature.

Main body

```
LOCAL gate AS MUTEX
gate = MUTEX()
? JSONENCODE(gate.Status())
```

Result / effect: Prints ownership and contention diagnostics.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

MUTEX.TryLock

API-086 / MUTEX

Syntax

```
TryLock([timeoutMs AS INTEGER])
```

Attempt to acquire the mutex without an unbounded wait.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL gate AS MUTEX
gate = MUTEX()
? gate.TryLock(0)
gate.Unlock()
```

Result / effect: Immediately attempts acquisition and prints .T./.F.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

MUTEX.Unlock

API-087 / MUTEX

Syntax

```
Unlock()
```

Release a mutex owned by the current Kokum execution.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL gate AS MUTEX
gate = MUTEX()
gate.Lock()
? gate.Unlock()
```

Result / effect: Prints .T. when the current execution releases gate.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

REGEX methods

REGEX.Close

API-112 / REGEX

Syntax

```
Close()
```

Releases compiled pattern resources.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
rx.Close()
? rx.IsClosed()
```

Result / effect: Prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.Find

API-105 / REGEX

Syntax

```
Find(subject AS STRING|FILE)
```

Returns the first matched STRING, or NULL.

Return: ANY. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? JSONENCODE(rx.Find("A12"))
rx.Close()
```

Result / effect: Prints the JSON string "12".

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.FindAll

API-106 / REGEX

Syntax

```
FindAll(subject AS STRING|FILE)
```

Returns all non-overlapping matched strings.

Return: ARRAY. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? LEN(rx.FindAll("A12 B3"))
rx.Close()
```

Result / effect: Prints 2.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.Flags

API-110 / REGEX

Syntax

```
Flags()
```

Returns the normalized flag string.

Return: STRING. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? TYPEOF(rx.Flags())
rx.Close()
```

Result / effect: Prints STRING.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.IsClosed

API-111 / REGEX

Syntax

```
IsClosed()
```

Tests whether the compiled resource is closed.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? rx.IsClosed()
rx.Close()
```

Result / effect: Prints .F. before Close().

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.IsMatch

API-103 / REGEX

Syntax

```
IsMatch(subject AS STRING|FILE)
```

Tests for a match.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? rx.IsMatch("A12")
rx.Close()
```

Result / effect: Prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.Pattern

API-109 / REGEX

Syntax

```
Pattern()
```

Returns the source pattern string.

Return: STRING. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? rx.Pattern()
rx.Close()
```

Result / effect: Prints [0-9]+.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.Replace

API-107 / REGEX

Syntax

```
Replace(subject AS STRING|FILE, replacement AS STRING)
```

Replaces matches and returns new text.

Return: STRING. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? rx.Replace("A12 B3", "#")
rx.Close()
```

Result / effect: Prints A# B#.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.Split

API-108 / REGEX

Syntax

```
Split(subject AS STRING|FILE)
```

Splits around this compiled pattern.

Return: ARRAY. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? JSONENCODE(rx.Split("A12B3C"))
rx.Close()
```

Result / effect: Prints ["A","B","C"].

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

REGEX.Test

API-104 / REGEX

Syntax

```
Test(subject AS STRING|FILE)
```

Alias of IsMatch.

Return: LOGICAL. **Arguments:** see signature.

Main body

```
LOCAL rx AS REGEX
rx = REGEXCOMPILE("[0-9]+")
? rx.Test("A12")
rx.Close()
```

Result / effect: Prints .T.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S05]; current signature also checked against [S01].

TASK methods

TASK.Await

API-082 / TASK

Syntax

```
Await([timeoutMs AS INTEGER])
```

Wait for completion and return or rethrow the result.

Return: ANY. **Arguments:** see signature.

Main body + SET-TASK

```
LOCAL task AS TASK
task = Double(21)
? task.Await(3000)
```

Result / effect: Waits up to 3 seconds and prints the task result.

Result/Await/Wait return the same cached result and rethrow a captured fault. A timeout does not cancel work, repeat a request or refresh a TOTP.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

TASK.Result

API-084 / TASK

Syntax

```
Result([timeoutMs AS INTEGER])
```

Alias for Await.

Return: ANY. **Arguments:** see signature.

Main body + SET-TASK

```
LOCAL task AS TASK
task = Double(21)
? task.Result(3000)
```

Result / effect: Alias of Await; prints the result.

Result/Await/Wait return the same cached result and rethrow a captured fault. A timeout does not cancel work, repeat a request or refresh a TOTP.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

TASK.Wait

API-083 / TASK

Syntax

```
Wait([timeoutMs AS INTEGER])
```

Alias for Await.

Return: ANY. **Arguments:** see signature.

Main body + SET-TASK

```
LOCAL task AS TASK
task = Double(21)
? task.Wait(3000)
```

Result / effect: Alias of Await; prints the result.

Result/Await/Wait return the same cached result and rethrow a captured fault. A timeout does not cancel work, repeat a request or refresh a TOTP.

Verification: Executed in source, IR, bytecode and external Linux VM. Source: [S06]; current signature also checked against [S01].

WEBSOCKET methods

WEBSOCKET.AddQueryParameter

API-063 / WEBSOCKET

Syntax

```
AddQueryParameter(name AS STRING, value AS STRING)
```

Set or replace a URL query parameter.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.AddQueryParameter("token", "abc")
```

Result / effect: Adds or replaces the token query parameter.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.ClearHeaders

API-056 / WEBSOCKET

Syntax

```
ClearHeaders()
```

Remove all custom upgrade headers.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.ClearHeaders()
```

Result / effect: Removes all custom upgrade headers.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.ClearMessages

API-080 / WEBSOCKET

Syntax

```
ClearMessages()
```

Discard queued incoming messages.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.ClearMessages()  
? wsFeed.PendingMessages
```

Result / effect: Prints 0.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.ClearQueryParameters

API-065 / WEBSOCKET

Syntax

```
ClearQueryParameters()
```

Remove all configured query parameters.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.ClearQueryParameters()
```

Result / effect: Removes all configured URL query parameters.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Close

API-068 / WEBSOCKET

Syntax

```
Close([code AS INTEGER [, reason AS STRING [, timeoutMs AS INTEGER]])
```

Close the WebSocket connection.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.Close(1000, "Application shutdown", 3000)
```

Result / effect: Performs a normal close and prints success.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Connect

API-067 / WEBSOCKET

Syntax

```
Connect([url AS STRING])
```

Connect to the configured WebSocket endpoint, optionally replacing its URL.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.Connect()
```

Result / effect: Prints .T. after the initial connection opens; a failed connection raises an error.

Initial Connect can block during DNS, connection and handshake. Subsequent messaging is worker-backed; keep GUI responsiveness in mind.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Disconnect

API-069 / WEBSOCKET

Syntax

```
Disconnect([code AS INTEGER [, reason AS STRING [, timeoutMs AS INTEGER]])
```

Alias for Close.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.Disconnect(1000, "Done", 3000)
```

Result / effect: Alias of Close.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Drain

API-079 / WEBSOCKET

Syntax

```
Drain([limit AS INTEGER])
```

Return and clear queued messages, optionally up to a limit.

Return: ARRAY. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
LOCAL batch AS ARRAY
batch = wsFeed.Drain(250)
? LEN(batch)
```

Result / effect: Returns and clears up to 250 queued events.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Ping

API-073 / WEBSOCKET

Syntax

```
Ping([payload AS STRING])
```

Send an optional WebSocket ping payload.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.Ping("health")
```

Result / effect: Queues a ping frame.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Poll

API-076 / WEBSOCKET

Syntax

```
Poll([timeoutMs AS INTEGER])
```

Alias for Receive.

Return: ANY. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
LOCAL message
message = wsFeed.Poll(0)
```

Result / effect: Alias of Receive; performs a nonblocking poll.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Receive

API-075 / WEBSOCKET

Syntax

```
Receive([timeoutMs AS INTEGER])
```

Receive the next queued event or NULL.

Return: ANY. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
LOCAL message
message = wsFeed.Receive(500)
IF message != NULL
  ? JSONENCODE(message)
ENDIF
```

Result / effect: Prints the next event map when available.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.ReceiveJSON

API-078 / WEBSOCKET

Syntax

```
ReceiveJSON([timeoutMs AS INTEGER])
```

Receive and decode the next JSON message or return NULL.

Return: JSON. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
LOCAL data AS JSON
data = wsFeed.ReceiveJSON(500)
? JSONENCODE(data)
```

Result / effect: Prints the next decoded JSON value or NULL.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.ReceiveText

API-077 / WEBSOCKET

Syntax

```
ReceiveText([timeoutMs AS INTEGER])
```

Receive the next text message or NULL.

Return: STRING. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
LOCAL text
text = wsFeed.ReceiveText(500)
IF text != NULL
    ? text
ENDIF
```

Result / effect: Prints the next text payload when available; a timeout produces no output.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.RemoveHeader

API-055 / WEBSOCKET

Syntax

```
RemoveHeader(name AS STRING)
```

Remove an HTTP upgrade header.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.RemoveHeader("X-Client-Code")
```

Result / effect: Prints .T. when the header existed.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.RemoveQueryParameter

API-064 / WEBSOCKET

Syntax

```
RemoveQueryParameter(name AS STRING)
```

Remove a URL query parameter.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.RemoveQueryParameter("token")
```

Result / effect: Prints .T. when the parameter existed.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SendBinary

API-072 / WEBSOCKET

Syntax

```
SendBinary(data AS ARRAY)
```

Queue an ARRAY of byte values 0..255.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.SendBinary([1,2,3,4])
```

Result / effect: Queues four binary bytes.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SendJSON

API-071 / WEBSOCKET

Syntax

```
SendJSON(value AS JSON)
```

Encode and queue a JSON message.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.SendJSON({"action":"subscribe","symbols":["NIFTY"]})
```

Result / effect: Encodes and queues a JSON frame.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SendText

API-070 / WEBSOCKET

Syntax

```
SendText(text AS STRING)
```

Queue a UTF-8 text message.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.SendText("hello")
```

Result / effect: Queues a UTF-8 text frame and prints .T.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetApiKey

API-059 / WEBSOCKET

Syntax

```
SetApiKey(token AS STRING [, header AS STRING])
```

Set an API key, optionally using a custom header.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetApiKey("api-key", "X-API-Key")
```

Result / effect: Sets the API key header.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetBasicAuth

API-060 / WEBSOCKET

Syntax

```
SetBasicAuth(username AS STRING, password AS STRING)
```

Set HTTP Basic authentication.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetBasicAuth("user", "secret")
```

Result / effect: Configures HTTP Basic authentication.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetBearerToken

API-057 / WEBSOCKET

Syntax

```
SetBearerToken(token AS STRING [, header AS STRING])
```

Set a Bearer token, optionally using a custom header.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetBearerToken("access-token")
```

Result / effect: Sets Authorization: Bearer access-token.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetCookie

API-061 / WEBSOCKET

Syntax

```
SetCookie(cookie AS STRING)
```

Set the Cookie header.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetCookie("session=abc123")
```

Result / effect: Sets the Cookie upgrade header.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetFeedToken

API-058 / WEBSOCKET

Syntax

```
SetFeedToken(token AS STRING [, header AS STRING])
```

Set a feed token, optionally using a custom header.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetFeedToken("feed-token", "X-Feed-Token")
```

Result / effect: Sets the feed token header.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetHeader

API-054 / WEBSOCKET

Syntax

```
SetHeader(name AS STRING, value AS STRING)
```

Set an HTTP upgrade header.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetHeader("X-Client-Code", "ABC123")
```

Result / effect: Adds the custom upgrade header.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetProtocols

API-052 / WEBSOCKET

Syntax

```
SetProtocols(protocols AS STRING)
```

Set the requested comma-separated subprotocol list.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetProtocols("json,market-v1")
```

Result / effect: Requests the two subprotocols.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetQueryParam

API-062 / WEBSOCKET

Syntax

```
SetQueryParam(name AS STRING, value AS STRING)
```

Set a URL query parameter.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetQueryParam("client", "kokum")
```

Result / effect: Adds or replaces ?client=kokum.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetReconnect

API-066 / WEBSOCKET

Syntax

```
SetReconnect(enabled AS LOGICAL [, delayMs AS INTEGER [, maxAttempts AS INTEGER]])
```

Configure automatic reconnection.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetReconnect(.T., 1000, 10)
```

Result / effect: Enables reconnect with 1-second delay and 10 attempts.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetSubprotocols

API-053 / WEBSOCKET

Syntax

```
SetSubprotocols(protocols AS STRING)
```

Alias for SetProtocols.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetSubprotocols("market-v1")
```

Result / effect: Alias of SetProtocols.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.SetUrl

API-051 / WEBSOCKET

Syntax

```
SetUrl(url AS STRING)
```

Change the WebSocket URL while disconnected.

Return: WEBSOCKET. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
wsFeed.SetUrl("ws://127.0.0.1:18082/echo")
```

Result / effect: Stores the URL while disconnected.

Loopback-only lab URL; use verified wss:// for untrusted networks.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Snapshot

API-081 / WEBSOCKET

Syntax

```
Snapshot()
```

Return connection counters and state.

Return: MAP. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? JSONENCODE(wsFeed.Snapshot())
```

Result / effect: Prints state/counter information without secrets.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

WEBSOCKET.Wait

API-074 / WEBSOCKET

Syntax

```
Wait([timeoutMs AS INTEGER])
```

Wait for connection or queued activity.

Return: LOGICAL. **Arguments:** see signature.

Slot body + SET-WEBSOCKET

```
? wsFeed.Wait(500)
```

Result / effect: Waits up to 500 ms for connection or queued activity.

Connect first. Run receive examples after SendJSON({"message":"hello"}) against the local echo lab.

Verification: Executed in a compiled Linux slot; setup required. Source: [S09]; current signature also checked against [S01].

Control-proxy methods A-Z

These 22 native callable members are accessed through a control MAP such as `form.cmbRole` or `form.tblData`. They are distinct from the automatically generated STRING/ARRAY value. The documented surface is not the larger browser-only JavaScript object API.

ComboBox.AddItem

CTL-002 / COMBOBOX

Syntax

```
AddItem(value AS STRING)
```

Append an item; return the resulting item count.

Return: INTEGER.

Slot body + SET-COMBO

```
form.cmbRole.AddItem("Tester")
```

Result / effect: Tester is appended.

Item insertion/removal positions are one-based; SelectedIndex is zero-based and -1 means no selection.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

ComboBox.ClearItems

CTL-006 / COMBOBOX

Syntax

```
ClearItems()
```

Clear items and selection.

Return: NULL. No useful return value is required.

Slot body + SET-COMBO

```
form.cmbRole.ClearItems()
```

Result / effect: The option list becomes empty.

Item insertion/removal positions are one-based; SelectedIndex is zero-based and -1 means no selection.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

ComboBox.ContainsItem

CTL-007 / COMBOBOX

Syntax

```
ContainsItem(value AS STRING)
```

Test for an exact stored item.

Return: LOGICAL.

Slot body + SET-COMBO

```
? form.cmbRole.ContainsItem("Analyst")
```

Result / effect: True after SET-COMBO preparation.

Item insertion/removal positions are one-based; SelectedIndex is zero-based and -1 means no selection.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

ComboBox.InsertItem

CTL-003 / COMBOBOX

Syntax

```
InsertItem(index AS INTEGER, value AS STRING)
```

Insert at a one-based position; existing selections are reconciled.

Return: INTEGER.

Slot body + SET-COMBO

```
form.cmbRole.InsertItem(2, "Support")
```

Result / effect: Support is inserted at position 2.

Item insertion/removal positions are one-based; SelectedIndex is zero-based and -1 means no selection.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

ComboBox.RemoveItem

CTL-004 / COMBOBOX

Syntax

```
RemoveItem(value AS STRING)
```

Remove the first matching text value.

Return: LOGICAL.

Slot body + SET-COMBO

```
? form.cmbRole.RemoveItem("Analyst")
```

Result / effect: True when the matching item existed.

Item insertion/removal positions are one-based; SelectedIndex is zero-based and -1 means no selection.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

ComboBox.RemoveItemAt

CTL-005 / COMBOBOX

Syntax

```
RemoveItemAt(index AS INTEGER)
```

Remove and return a one-based item.

Return: STRING.

Slot body + SET-COMBO

```
? form.cmbRole.RemoveItemAt(1)
```

Result / effect: The first option text is returned.

Item insertion/removal positions are one-based; SelectedIndex is zero-based and -1 means no selection.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

ComboBox.SetItems

CTL-001 / COMBOBOX

Syntax

```
SetItems(items AS ARRAY)
```

Replace all items; return the resulting count.

Return: INTEGER.

Slot body + SET-COMBO

```
form.cmbRole.SetItems(["Analyst", "Developer"])
```

Result / effect: Two choices are available.

Item insertion/removal positions are one-based; SelectedIndex is zero-based and -1 means no selection.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

DatePicker.GetDate

CTL-008 / DATEPICKER

Syntax

```
GetDate(mask AS STRING)
```

Format the selected ISO date without changing its stored value.

Return: STRING.

Slot body + SET-DATE

```
form.dtAdmission.Value = "2026-09-20"
? form.dtAdmission.GetDate("dd-mm-YYYY")
```

Result / effect: 20-09-2026.

Supports YYYY/YY, mm and dd once each, with ASCII punctuation/space separators. Empty selection returns "". Invalid mask: KFC5401; invalid Value: KFC5402. Requires Linux GUI contract 1.9.

Verification: Executed in a compiled Linux slot; setup required. Source: [S13]; current signature also checked against [S01].

PropertyGrid.AddProperty

CTL-018 / PROPERTYGRID

Syntax

```
AddProperty(category, name, caption, value [, type [, readOnly]])
```

Add one uniquely named property under a category.

Return: LOGICAL.

Slot body + SET-GRID

```
? form.gridInfo.AddProperty( ;
    "General", "city", "City", "Ratnagiri", "string")
```

Result / effect: Adds the City field.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

PropertyGrid.Clear

CTL-022 / PROPERTYGRID

Syntax

```
Clear()
```

Remove all categories and properties.

Return: LOGICAL.

Slot body + SET-GRID

```
form.gridInfo.Clear()
```

Result / effect: The grid is empty.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

PropertyGrid.GetValue

CTL-020 / PROPERTYGRID

Syntax

```
GetValue(name AS STRING)
```

Read an existing value; absent names return NULL.

Return: ANY.

Slot body + SET-GRID

```
? form.gridInfo.GetValue("title")
```

Result / effect: Kokum after SET-GRID.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

PropertyGrid.RemoveProperty

CTL-021 / PROPERTYGRID

Syntax

```
RemoveProperty(name AS STRING)
```

Remove the named property.

Return: LOGICAL.

Slot body + SET-GRID

```
? form.gridInfo.RemoveProperty("title")
```

Result / effect: Returns true when removed.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

PropertyGrid.SetValue

CTL-019 / PROPERTYGRID

Syntax

```
SetValue(name AS STRING, value)
```

Change an existing property value.

Return: LOGICAL.

Slot body + SET-GRID

```
? form.gridInfo.SetValue("title", "Customer view")
```

Result / effect: Changes the title property.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.AddColumn

CTL-009 / TABLEVIEW

Syntax

```
AddColumn(id AS STRING [, caption AS STRING [, width AS NUMBER [, alignment AS STRING]])
```

Append a column; the native result is the resulting column count.

Return: INTEGER.

Slot body + SET-TABLE

```
form.tblData.AddColumn("score", "Score", 100)
```

Result / effect: Adds a third column after SET-TABLE.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

The native method accepts up to four arguments. This example tests identity/caption and count, not rendered width/alignment behavior.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.AddRow

CTL-010 / TABLEVIEW

Syntax

```
AddRow(values AS ARRAY) or AddRow(values AS MAP) or AddRow(value1 [, value2 ...])
```

Append a row using the current columns.

Return: INTEGER.

Slot body + SET-TABLE

```
form.tblData.AddRow(["3", "Mira"])
```

Result / effect: Adds a row; returns the resulting row count.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.ClearColumns

CTL-017 / TABLEVIEW

Syntax

```
ClearColumns()
```

Remove columns and their row data.

Return: LOGICAL.

Slot body + SET-TABLE

```
form.tblData.ClearColumns()
```

Result / effect: The method-owned table is empty.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.ClearRows

CTL-016 / TABLEVIEW

Syntax

```
ClearRows()
```

Remove all rows but retain the columns.

Return: LOGICAL.

Slot body + SET-TABLE

```
form.tblData.ClearRows()
```

Result / effect: RowCount becomes zero.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.GetCell

CTL-015 / TABLEVIEW

Syntax

```
GetCell(row AS INTEGER, columnIndexOrName)
```

Read a cell without changing it.

Return: STRING or NULL.

Slot body + SET-TABLE

```
? form.tblData.GetCell(0, "name")
```

Result / effect: Anil after SET-TABLE.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.RemoveColumn

CTL-011 / TABLEVIEW

Syntax

```
RemoveColumn(indexOrName)
```

Remove the column identified by a zero-based index or key.

Return: LOGICAL.

Slot body + SET-TABLE

```
? form.tblData.RemoveColumn("name")
```

Result / effect: Removes the name column.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.RemoveRow

CTL-012 / TABLEVIEW

Syntax

```
RemoveRow(row AS INTEGER)
```

Remove a zero-based row.

Return: LOGICAL.

Slot body + SET-TABLE

```
? form.tblData.RemoveRow(0)
```

Result / effect: Removes the first row.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.SetCell

CTL-014 / TABLEVIEW

Syntax

```
SetCell(row AS INTEGER, columnIndexOrName, value)
```

Replace one cell; native row/column indices are zero-based.

Return: LOGICAL.

Slot body + SET-TABLE

```
? form.tblData.SetCell(0, "name", "Ameya")
```

Result / effect: The first name cell becomes Ameya.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView.UpdateRow

CTL-013 / TABLEVIEW

Syntax

```
UpdateRow(row AS INTEGER, values AS ARRAY)
```

Replace one zero-based row.

Return: LOGICAL.

Slot body + SET-TABLE

```
? form.tblData.UpdateRow(0, ["1", "Ameya"])
```

Result / effect: The first row is replaced.

Use method-owned rows for this example. Do not switch from an automatic ARRAY assignment to later native row/cell methods: the documented handoff defect remains open.

Verification: Executed in a compiled Linux slot; setup required. Source: [S02]; current signature also checked against [S01].

TableView property-style accessors

RowCount, ColumnCount and SelectedRow are also native accessor names. The implementation supports reading them through the control property/member path; Use property-style reads such as `form.tblData.RowCount` and `form.gridInfo.PropertyCount`; ComboBox also supplies `ItemCount` and `SelectedValue`. These are accessor properties rather than additional method entries. Their row/column interpretation remains subject to the documented data-ownership boundary. Do not use a method name on the automatic `tblData` ARRAY.

Native resource property directory

This directory covers all 90 property entries advertised by the current language service. Some descriptions are last-known local state, not a new query of a remote server. Properties are read without parentheses. Where mutation is supported, use the documented setting method rather than guessing that any property is writable.

ATOMICINTEGER properties

Property	Type	Meaning
Operations	INTEGER	Number of atomic operations performed.
Value	INTEGER	Current atomic integer value.

CHANNEL properties

Property	Type	Meaning
Capacity	INTEGER	Fixed bounded channel capacity.
Closed	LOGICAL	True after the channel has been closed.
Count	INTEGER	Current number of queued messages.
Received	INTEGER	Total successfully received messages.
Sent	INTEGER	Total successfully sent messages.
WaitingReceivers	INTEGER	Current blocked receiver count.
WaitingSenders	INTEGER	Current blocked sender count.

DATABASE properties

Property	Type	Meaning
ApplicationName	STRING	Provider application name.
BackendPid	INTEGER	PostgreSQL backend process identifier when available.
Changes	INTEGER	Rows changed by the latest Execute operation.
Database	STRING	Selected database name.
Endpoint	STRING	Resolved provider endpoint.
File	STRING	SQLite database path when applicable.
Host	STRING	Remote database host when applicable.
Id	STRING	Stable database component identifier.
IsOpen	LOGICAL	True while the database connection is open.
LastElapsedMs	NUMBER	Duration of the latest provider request.
LastInsertId	INTEGER	Latest generated row identifier when available.
Port	INTEGER	Remote database port when applicable.

Property	Type	Meaning
Provider	STRING	Canonical provider name: sqlite, tigerdb, or postgresql.
ReadOnly	LOGICAL	Whether the component is configured read-only.
RequestCount	INTEGER	Number of provider requests issued.
Schema	STRING	Selected PostgreSQL schema.
ServerVersion	INTEGER	Connected PostgreSQL server version number when available.
SSLMode	STRING	PostgreSQL SSL mode.
TLS	LOGICAL	Whether TLS is enabled for the provider.

GRAPH properties

Property	Type	Meaning
Animation	LOGICAL	Current redraw-animation setting.
Candles	ARRAY	Current candlestick data.
EmptyText	STRING	Message shown when no graph data exists.
GraphType	STRING	Alias for Type.
Id	STRING	Stable Graph control identifier.
Labels	ARRAY	Current graph labels.
LineWidth	NUMBER	Current line and candle stroke width.
MaximumPoints	INTEGER	Maximum retained point count.
MaxPoints	INTEGER	Alias for MaximumPoints.
Options	MAP	Current rendering options.
Palette	STRING	Current graph colour palette.
PointCount	INTEGER	Number of graph points or candles.
Revision	INTEGER	Current graph-state revision.
Series	ARRAY	Current graph series.
SeriesCount	INTEGER	Number of graph series.
ShowGrid	LOGICAL	True when Cartesian grid lines are visible.
ShowLegend	LOGICAL	True when the legend is visible.
Smooth	LOGICAL	Current line-smoothing setting.
Stacked	LOGICAL	Current stacked-series setting.
Title	STRING	Displayed graph title.
Type	STRING	line, bar, pie, or candle.
XAxisTitle	STRING	Horizontal-axis title.
YAxisTitle	STRING	Vertical-axis title.

IMAGE properties

Property	Type	Meaning
KeepAspectRatio	LOGICAL	Current aspect-ratio policy.
Loaded	LOGICAL	True after a dynamic image has been loaded.
MimeType	STRING	image/jpeg or image/png.
Path	STRING	Current local source path.
Revision	INTEGER	Current private-image revision.
ScaleMode	STRING	fit or stretch.
Size	INTEGER	Loaded image size in bytes.
Source	STRING	Current browser-facing image source.

MUTEX properties

Property	Type	Meaning
Acquisitions	INTEGER	Successful lock-acquisition count.
Locked	LOGICAL	True while the mutex is owned by one Kokum execution.
OwnerThreadId	INTEGER	Logical owner execution identifier, or zero while unlocked.
Waiters	INTEGER	Current number of native waiters.

TASK properties

Property	Type	Meaning
ElapsedMs	INTEGER	Task execution duration in milliseconds.
ErrorMessage	STRING	Captured worker error, or an empty string when no error exists.
IsCompleted	LOGICAL	True after successful or failed completion.
IsFaulted	LOGICAL	True when the worker ended with an error.
IsRunning	LOGICAL	True for pending or running work in the current implementation; use IsDone for completion.
Status	STRING	pending, running, completed, or faulted.
ThreadId	INTEGER	Native KokumVM worker identifier, or zero before the worker starts.

TASK.Status uses lowercase values; THREADSTATUS returns uppercase names. Timing-dependent states can change before the next expression. Error/Result access does not cancel or rerun work.

WEBSOCKET properties

Property	Type	Meaning
Connected	LOGICAL	Alias for IsConnected.
DroppedMessages	INTEGER	Dropped incoming message count.

Property	Type	Meaning
Id	STRING	Stable WebSocket component identifier.
IsConnected	LOGICAL	True while connected.
IsRunning	LOGICAL	True while the worker is active.
LastCloseCode	INTEGER	Latest close status code.
LastCloseReason	STRING	Latest close reason.
LastError	STRING	Latest WebSocket error.
LastMessage	STRING	Latest received text message.
PendingMessages	INTEGER	Queued incoming message count.
PendingSends	INTEGER	Queued outgoing message count.
Protocol	STRING	Negotiated subprotocol.
ReceivedBytes	INTEGER	Total received bytes.
ReceivedMessages	INTEGER	Total received-message count.
Reconnect	LOGICAL	Whether automatic reconnection is enabled.
ReconnectCount	INTEGER	Reconnect-attempt count.
SentBytes	INTEGER	Total sent bytes.
SentMessages	INTEGER	Total sent-message count.
State	STRING	Current connection state.
Url	STRING	Current ws:// or wss:// URL.

GUI scalar properties, schema defaults and permitted child types are covered in the companion GUI Designer Guide. Native FILE functions use global operations rather than an invented `file.Read()/Seek()` object API.

HTTP, REST and TOTP details

Choose the request function

Function	Purpose	Redirect default
URLGET(url [, options])	GET a URL	FollowRedirects = .T.
URLPOST(url, body [, options])	POST a body	FollowRedirects = .T.
URLREQUEST(method, url, body [, options])	GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS	FollowRedirects = .F.

The URLREQUEST body argument is mandatory. Pass NULL for GET, HEAD and OPTIONS, which do not accept a request body in this contract. Options, when supplied, must be a MAP. Success at the transport level can return HTTP 400/500 normally: inspect Ok and Status. DNS, TLS, timeout and malformed-response failures raise errors. [S10]

Option	Type	Meaning
Headers	MAP	Explicit request headers; protect credentials and avoid contradictory framing.
Query	MAP	Query values; use the helper instead of manual concatenation.
ConnectTimeoutMs	INTEGER	Connection phase timeout.
TimeoutMs	INTEGER	Overall configured request timeout.
ReadTimeoutMs	INTEGER	Read operation timeout.
MaxResponseBytes	INTEGER	Response-size ceiling.
FollowRedirects	LOGICAL	Enable/disable redirect handling.
MaxRedirects	INTEGER	Maximum followed redirects.
VerifyTLS	LOGICAL	Keep verification enabled for production.
CAFile	STRING	Explicit trust-anchor file when required.
TLSServerName	STRING	TLS identity/SNI override; not a reason to disable verification.
ContentType	STRING	Explicit content type.
BodyType	STRING	Body encoding policy, such as JSON/text/form.
BearerToken	STRING	Bearer authentication helper.
BasicAuth	MAP	Username/password credentials using the supported map fields.
Cookie	STRING	Explicit caller-managed cookie value.
UserAgent	STRING	Request user agent.
CookiePolicy	STRING	Cookie forwarding policy; omit or same-origin.
AllowInsecureAuth	LOGICAL	Explicit insecure-auth exception; do not use in production examples.
AllowCrossOriginBody	LOGICAL	Explicit permission for preserving body on cross-origin replay.

Use conservative timeouts and response bounds. Cross-origin redirection can remove credentials or refuse body replay. Cookie handling is a caller-controlled policy, not a browser cookie jar. No automatic REST retry or durable delivery guarantee is introduced. A task timeout is not a transport cancellation. Consult the supplied source contract for deployment-specific defaults rather than relying on a provider's sample code.

Response field	Type	Meaning
Ok	LOGICAL	HTTP success status interpretation.
Status	INTEGER	HTTP status code.
Reason	STRING	HTTP reason phrase.
HttpVersion	STRING	Parsed protocol version.
Url	STRING	Effective request URL.
Headers	MAP	Normalized header values.
HeaderValues	MAP	Multiple values for repeated headers.
Body	STRING	Returned response text.
Json	ANY / JSON	Decoded JSON value when supported; may be NULL.
JsonError	STRING	JSON decoding diagnostic when applicable.
ContentType	STRING	Response content type.
ContentLength	INTEGER	Returned body byte count; HEAD has an empty Body.
RedirectCount	INTEGER	Number of followed redirects.
ElapsedMs	INTEGER	Observed request elapsed time.

TOTP settings and safe use

Option	Type	Default / range
Algorithm	STRING	SHA1; SHA1, SHA256 or SHA512
Digits	INTEGER	6; from 6 through 8
PeriodSeconds	INTEGER	30; from 1 through 86400
Timestamp	INTEGER	Current Unix seconds if absent; 0..INT64_MAX
StartTime	INTEGER	0; no later than the resolved timestamp

TOTP accepts Base32 text, not an otpauth URI, QR image or hexadecimal string. Option keys and supported algorithm names are ASCII case-insensitive, but algorithm spellings are not trimmed; SHA-1 is not an alias for SHA1. Unknown/case-duplicate keys and non-INTEGER numeric settings are rejected. The result is a STRING preserving leading zeroes. [S11]

Public deterministic reference: `output 94287082`

```
LOCAL code AS STRING
code = TOTP( ;
    "GEZDGNBVG3TQ0JQGEZDGNBVG3TQ0JQ", ;
    {"Timestamp": 59, "Digits": 8})
? code
```

An explicit timestamp reads no clock; an omitted timestamp samples time when the call executes. Generation alone does not enroll an account, authenticate a broker, compare OTPs or enforce replay protection. Do not log real secrets or codes. A task caches its calculated result; retrieving it again is not a fresh code.

Diagnostic	Meaning
KTOTP1000–1007	Invocation, type, Base32, options, algorithm, digit/period or time-range error.
KTOTP1900	Native allocation failure in the guarded TOTP paths.
KTOTP2001	Clock unavailable or unrepresentable.
KTOTP2002 / KTOTP2003	Provider algorithm refusal / HMAC calculation failure.

Language and service recipes

This chapter provides the language context around the callable reference. Statements and syntax are not counted as built-in functions. Keep routines, modules, loops and tasks distinct from resource methods. The listings use ordinary current Kokum syntax and supplied application/tooling conventions.

Variables, arrays, maps and validation

Kokum example

```
FUNCTION Main AS INTEGER
  LOCAL values AS ARRAY
  LOCAL record AS MAP
  LOCAL id AS INTEGER
  values = [10, 20, 30]
  record = {"id": 123, "name": "  Kokum  "}
  id = INT(record["id"])
  ? values[1]
  ? TRIM(record["name"])
  ? id
  RETURN 0
ENDFUNC
```

Expected values are 10, Kokum and 123. For user IDs supplied as NUMBER, reject a fractional value before treating INT truncation as validation. For very large integer text use an exact representation, for example INT(DECIMAL("9007199254740993")); a prior conversion through VAL can already lose precision.

Conditions, loops and routines

Kokum example

```
FUNCTION Main AS INTEGER
  LOCAL total AS INTEGER
  LOCAL i AS INTEGER
  total = 0
  FOR i = 1 TO 3
    total = total + i
  NEXT
  IF total = 6
    ? "sum is six"
  ELSE
    ? "unexpected"
  ENDIF
  RETURN 0
ENDFUNC
```

FOR EACH iterates a collection; DO WHILE repeats while its condition holds. EXIT/BREAK leave the loop; LOOP advances to its next iteration. Use TRY/CATCH/FINALLY around operations with a meaningful recovery path. A routine with an INTEGER result must actually return an INTEGER, not a NUMBER that happens to display without decimals.

Imported module

modules/math.kokum

```
MODULE book.math VERSION "1.0.0"
EXPORT FUNCTION Twice

FUNCTION Twice(value AS INTEGER) AS INTEGER
  RETURN value * 2
ENDFUNC
```

src/main.kokum

```

MODULE book.app VERSION "1.0.0"
IMPORT FUNCTION Twice FROM book.math VERSION "^1.0.0"
EXPORT FUNCTION Main

FUNCTION Main AS INTEGER
  ? Twice(21)
  RETURN 0
ENDFUNC

```

Add this module to the wizard-created project manifest

```

[[module]]
name = "book.math"
version = "1.0.0"
source = "modules/math.kokum"

```

Check and bundle kokum.toml rather than running an unlinked importer as an independent source file. Expected output is 42. Updating an imported native/common implementation is a separate product compatibility step, not a source string replacement.

Streaming file search and natural patterns

Kokum example

```

FUNCTION Main AS INTEGER
  LOCAL file AS FILE
  LOCAL count AS INTEGER
  file = OPEN("scan-lab.txt", "w")
  WRITE(file, "INFO ready\nERROR first\nERROR second\n")
  CLOSEFILE(file)
  file = OPEN("scan-lab.txt", "r")
  FIND FROM FILE file ALL LINES ;
    WHERE STARTS WITH "ERROR" COUNT TO count
  ? count
  CLOSEFILE(file)
  RETURN 0
ENDFUNC

```

Expected count is 2. This writes only scan-lab.txt in its working directory. LINES/WORDS streaming avoids whole-subject materialization for suitable work. COUNT TO avoids retaining values; INTO collects them. Direct regex FILE subjects read from the current position to EOF and leave the handle open. General cross-line regex is not automatically parallelized. [S05]

Natural operations include FIND, REPLACE, SPLIT, VALIDATE and ITERATE, with source units, selectors and optional details/captures. Their complete grammar is separate from the global function signatures. FIRST/LAST/ALTERNATE selectors and LIMIT have source-order semantics; do not reinterpret them independently per worker chunk.

A small Kokum HTTP service

service.kokum

```

MODULE book.service VERSION "1.0.0"
EXPORT FUNCTION Main
EXPORT FUNCTION Hello

FUNCTION Main AS INTEGER
  RETURN 0
ENDFUNC

FUNCTION Hello(request AS MAP) AS MAP
  RETURN {"Json": {"message": "Hello from Kokum"}}
ENDFUNC

```

service.conf

```
[server]
listen=127.0.0.1
port=18081
workers=2
queue_capacity=8
max_request_bytes=1048576
max_body_bytes=65536
read_timeout_ms=3000
write_timeout_ms=3000
graceful_shutdown_ms=5000

[route hello]
method=GET
path=/hello
handler=Hello
parameters=1
```

Linux terminal

```
kokumc compile service.kokum -o service.kbc
kokumvm serve service.kbc --config service.conf
```

With the service running, `URLGET("http://127.0.0.1:18081/hello")` returns a JSON response containing message. This is an independent local lab, not an authenticated public service. Handlers receive request MAPs; response MAPs can specify Json, Body, Status and headers according to the backend contract. Keep error responses bounded and do not forward credentials indiscriminately. [S16]

Concurrency and shutdown

Workers do not share a live GUI form or arbitrary mutable globals. Use snapshots, explicit channels/atomics/mutexes where supported, and owner-side publication. A database worker opens a connection from copied configuration; it does not transport a transaction or the contents of an independent in-memory database. Graceful service shutdown cannot force arbitrary user code or external I/O to complete instantly. No automatic retry/exactly-once guarantee is implied.

Sources, scope and verification

This edition describes the completed Kokum Linux source package below. It is application-developer documentation, not a C embedding API reference or a promise that every historical platform and renderer feature is complete.

Source identity

```
Kokum_Linux_1_1_0_MakeDist_Existing_Binaries.zip
SHA-256
9123f697fc536700a3d86926c8dc9548c5ddfa47b94e4bd01fa4d2d578f9561f
```

Product 1.1.0; VM ABI 1.7; browser GUI contract 1.9; bytecode/bundle format 1.3. Windows and macOS remain separate projects and have not adopted this Linux DatePicker/GetDate revision. No source code was edited for these books.

S01 — Built-in catalogue and language behavior

src/tlang_builtin_catalog.c; src/tlang_language_service.c; src/tlang_interpreter.c; src/tlang_ir_executor.c; src/tlang_vm_engine.c

S02 — Widget schema, control methods and form bridge

src/tlang_gui_schema.c; src/kokum_form_control_internal.h; src/tlang_web_form.c; src/tlang_form_variables.c; include/tlang_gui_schema.h

S03 — Designer workflow, events and automatic variables

MD/docs/BROWSER_VISUAL_FORM_DESIGNER.md; MD/docs/SLOT_BINDINGS.md; MD/docs/AUTOMATIC_WIDGET_VARIABLES.md; web/ide/kokum-form-designer.js

S04 — Scalar mathematics and array statistics

MD/docs/SCALAR_MATHEMATICS.md; MD/docs/ARRAY_STATISTICS.md; src/kokum_math.c; src/kokum_statistics.c

S05 — Files, regex and natural patterns

MD/docs/FILE_IO.md; MD/docs/PHASE4_16_7_NATURAL_PATTERN_OPERATIONS.md; src/kokum_fileio.c; src/kokum_regex.c; src/kokum_find.c

S06 — Tasks, named threads and synchronization

MD/docs/ASYNC_TASKS.md; MD/docs/PHASE4_14_2_NAMED_THREADS.md; MD/docs/PHASE4_14_4_SAFE_SHARED_STATE.md; src/kokum_task.c; src/kokum_sync.c

S07 — Database components and Remote FlatDB

MD/docs/DATABASE_COMPONENTS.md; src/kokum_database.c; src/kokum_flatdb.c; tests/run_kokum_029_flatdb_tests.py

S08 — Graph resource

MD/docs/GRAPH_WIDGET.md; src/kokum_graph.c

S09 — Image and WebSocket resources

MD/docs/IMAGE_WIDGET.md; MD/docs/WEBSOCKET_CLIENT.md; src/kokum_image.c; src/kokum_websocket.c

S10 — REST and HTTP

MD/docs/REST_REQUEST_EXECUTION.md; MD/docs/REST_SECURITY_FAILURES_CONCURRENCY.md; MD/docs/URL_ASYNC_MICROSERVICES_IDE.md; src/kokum_url.c

S11 — TOTP

MD/docs/TOTP_BUILTIN.md; src/kokum_totp.c; src/kokum_totp_crypto.c

S12 — String trimming

src/kokum_string_trim_internal.h; tests/features/string_trim/; Kokum_String_Trim_Guide.md supplied with the trimming release

S13 — DatePicker and GetDate

Current DatePicker/GetDate guides, feature tests and src/kokum_form_control_internal.h; src/tlang_gui_schema.c

S14 — Local and network IDE administration

MD/docs/LAN_CSV_AUTHENTICATION.md; MD/docs/LAN_PROJECT_OPERATIONS.md; MD/docs/LAN_REMOTE_WORKSPACES.md; config/kokum-ide-lan.conf.example

S15 — Current binary distribution

Root make_dist; Kokum_MakeDist_Existing_Binaries_Guide.md, distributed with the current source edition

S16 — Backend services and project operations

src/kokum_backend.c; examples/kokum_029_backend_services/; current kokumc and kokumvm command help

S17 — Reference example foundations

Kokum_Complete_Command_Reference_0_29_0.docx; earlier GUI Designer manual screenshots. Entries were checked against the current source and newly tested where stated.

What was checked for this edition

The unchanged current Linux compiler, VM, IDE and FlatDB server were built out of tree. All 244 callable examples in the companion function reference passed compiler checking. Of these, 241 were executed: 133 through the source, IR, bytecode and external-VM paths; 103 through compiled GUI slots; and five against a real local Kokum FlatDB server. Three server-specific database examples were compiler-checked only and require an operator-supplied remote test database.

GUI-slot execution verifies native calls and protocol results; it is not a claim that every widget was visually tested in this documentation pass. Earlier acceptance evidence, static source contracts and current executable examples are distinguished in the text. Synthetic/local networking is not production broker or database-server acceptance. A check can pass while a separately documented renderer limitation remains open.

The documents were generated separately from the source checkout. DOCX pagination and navigation were rendered and inspected. No native Windows/macOS run, full product regression, clean-machine distribution test, comprehensive browser/accessibility test or new security certification is asserted.

A-Z callable index

Page numbers refer to this edition. Entries are linked; use Ctrl+F for an exact function, control or method name.

Entry	Section	Page
AADD	Built-in	12
ABS	Built-in	12
ADEL	Built-in	13
AINS	Built-in	13
APPEND	Built-in	13
ASET	Built-in	14
ATOMICADD	Built-in	14
ATOMICCOMPAREEXCHANGE	Built-in	15
ATOMICDEC	Built-in	15
ATOMICGET	Built-in	16
ATOMICINC	Built-in	16
ATOMICINTEGER	Built-in	16
ATOMICINTEGER.Add	Resource method	53
ATOMICINTEGER.CompareExchange	Resource method	53
ATOMICINTEGER.Decrement	Resource method	53
ATOMICINTEGER.Exchange	Resource method	54
ATOMICINTEGER.Get	Resource method	54
ATOMICINTEGER.Increment	Resource method	54
ATOMICINTEGER.Set	Resource method	55
ATOMICINTEGER.Status	Resource method	55
ATOMICSET	Built-in	17
AVERAGE	Built-in	17
CEIL	Built-in	17
CEILING	Built-in	18
CHANNEL	Built-in	18
CHANNEL.Close	Resource method	55
CHANNEL.Receive	Resource method	56
CHANNEL.Send	Resource method	56
CHANNEL.Status	Resource method	56
CHANNEL.TryReceive	Resource method	57
CHANNEL.TrySend	Resource method	57
CHANNELCAPACITY	Built-in	18
CHANNELCLOSE	Built-in	19
CHANNELCLOSED	Built-in	19
CHANNELCOUNT	Built-in	19
CHANNELRECEIVE	Built-in	20
CHANNELSEND	Built-in	20
CHANNELSTATUS	Built-in	20
CLAMP	Built-in	21
CLASSNAME	Built-in	21
CLOSEFILE	Built-in	21
CLOSEFORM	Built-in	22

Entry	Section	Page
ComboBox.AddItem	Control method	91
ComboBox.ClearItems	Control method	91
ComboBox.ContainsItem	Control method	91
ComboBox.InsertItem	Control method	92
ComboBox.RemoveItem	Control method	92
ComboBox.RemoveItemAt	Control method	92
ComboBox.SetItems	Control method	93
COUNTNUM	Built-in	22
DATABASE.Begin	Resource method	57
DATABASE.Close	Resource method	58
DATABASE.Commit	Resource method	58
DATABASE.CreateTableFromJson	Resource method	58
DATABASE.Execute	Resource method	59
DATABASE.Open	Resource method	59
DATABASE.Ping	Resource method	59
DATABASE.Query	Resource method	60
DATABASE.QueryOne	Resource method	60
DATABASE.QueryTable	Resource method	60
DATABASE.QueryValue	Resource method	61
DATABASE.Raw	Resource method	61
DATABASE.Rollback	Resource method	61
DATABASE.UseDatabase	Resource method	62
DATE	Built-in	22
DatePicker.GetDate	Control method	93
DATETIME	Built-in	23
DECIMAL	Built-in	23
EVAL	Built-in	23
EXP	Built-in	24
FLOOR	Built-in	24
GCCOLLECT	Built-in	24
GCSTATS	Built-in	25
GETPROPERTY	Built-in	25
GRAPH.AddCandle	Resource method	62
GRAPH.AddPoint	Resource method	62
GRAPH.AddSeries	Resource method	63
GRAPH.AddSlice	Resource method	63
GRAPH.Clear	Resource method	63
GRAPH.ClearData	Resource method	64
GRAPH.Refresh	Resource method	64
GRAPH.RemoveSeries	Resource method	64
GRAPH.SetAnimation	Resource method	65
GRAPH.SetCandles	Resource method	65
GRAPH.SetData	Resource method	65
GRAPH.SetEmptyText	Resource method	66
GRAPH.SetGraphType	Resource method	66

Entry	Section	Page
GRAPH.SetGrid	Resource method	66
GRAPH.SetLabels	Resource method	67
GRAPH.SetLegend	Resource method	67
GRAPH.SetLineWidth	Resource method	67
GRAPH.SetMaximumPoints	Resource method	68
GRAPH.SetMaxPoints	Resource method	68
GRAPH.SetOption	Resource method	68
GRAPH.SetPalette	Resource method	69
GRAPH.SetPie	Resource method	69
GRAPH.SetSeries	Resource method	69
GRAPH.SetSmooth	Resource method	70
GRAPH.SetStacked	Resource method	70
GRAPH.SetTitle	Resource method	70
GRAPH.SetType	Resource method	71
GRAPH.SetXAxisTitle	Resource method	71
GRAPH.SetYAxisTitle	Resource method	71
GRAPH.ShowGrid	Resource method	72
GRAPH.ShowLegend	Resource method	72
GRAPH.Snapshot	Resource method	72
GRAPH.SupplyData	Resource method	73
HASKEY	Built-in	25
HASMETHOD	Built-in	26
HASPROPERTY	Built-in	26
HIDEFORM	Built-in	26
IMAGE.Clear	Resource method	73
IMAGE.SetImage	Resource method	73
IMAGE.Snapshot	Resource method	74
INT	Built-in	27
JSONDECODE	Built-in	27
JSONENCODE	Built-in	27
KCONNECT	Built-in	28
KCONNECTED	Built-in	28
KDISCONNECT	Built-in	29
KEXECUTE	Built-in	29
KEYS	Built-in	30
KQUERY	Built-in	30
LEN	Built-in	30
LOCK	Built-in	31
LOG	Built-in	31
LOG10	Built-in	31
LOWER	Built-in	32
LTRIM	Built-in	32
MAPREMOVE	Built-in	32
MAPSET	Built-in	33
MAX	Built-in	33

Entry	Section	Page
MEAN	Built-in	33
MEDIAN	Built-in	34
METHODS	Built-in	34
MIN	Built-in	34
MOD	Built-in	35
MUTEX	Built-in	35
MUTEX.Lock	Resource method	74
MUTEX.Status	Resource method	74
MUTEX.TryLock	Resource method	75
MUTEX.Unlock	Resource method	75
MUTEXSTATUS	Built-in	35
OPEN	Built-in	36
PERCENTILE	Built-in	36
POW	Built-in	36
PROPERTIES	Built-in	37
PropertyGrid.AddProperty	Control method	93
PropertyGrid.Clear	Control method	94
PropertyGrid.GetValue	Control method	94
PropertyGrid.RemoveProperty	Control method	94
PropertyGrid.SetValue	Control method	95
RANGE	Built-in	37
READ	Built-in	38
READLINE	Built-in	38
REGEX.Close	Resource method	75
REGEX.Find	Resource method	76
REGEX.FindAll	Resource method	76
REGEX.Flags	Resource method	76
REGEX.IsClosed	Resource method	77
REGEX.IsMatch	Resource method	77
REGEX.Pattern	Resource method	77
REGEX.Replace	Resource method	78
REGEX.Split	Resource method	78
REGEX.Test	Resource method	78
REGEXCOMPILE	Built-in	39
REGEXESCAPE	Built-in	39
REGEXFIND	Built-in	40
REGEXFINDALL	Built-in	40
REGEXMATCH	Built-in	41
REGEXREPLACE	Built-in	41
REGXSPLIT	Built-in	41
ROUND	Built-in	42
RTRIM	Built-in	42
SETPROPERTY	Built-in	42
SHOWFORM	Built-in	43
SIGN	Built-in	43

Entry	Section	Page
SLEEP	Built-in	43
SQRT	Built-in	44
STDDEV	Built-in	44
STR	Built-in	44
SUM	Built-in	45
SYNCSTATS	Built-in	45
TableView.AddColumn	Control method	95
TableView.AddRow	Control method	95
TableView.ClearColumns	Control method	96
TableView.ClearRows	Control method	96
TableView.GetCell	Control method	96
TableView.RemoveColumn	Control method	97
TableView.RemoveRow	Control method	97
TableView.SetCell	Control method	97
TableView.UpdateRow	Control method	98
TASK.Await	Resource method	79
TASK.Result	Resource method	79
TASK.Wait	Resource method	79
THREADDONE	Built-in	45
THREADERROR	Built-in	46
THREADID	Built-in	46
THREADPOOLSIZE	Built-in	46
THREADSTATUS	Built-in	47
TOTP	Built-in	47
TRIM	Built-in	48
TRUNC	Built-in	48
TYPEOF	Built-in	48
UNLOCK	Built-in	49
UPPER	Built-in	49
URLGET	Built-in	49
URLPOST	Built-in	50
URLREQUEST	Built-in	50
VAL	Built-in	51
VALUES	Built-in	51
VARIANCE	Built-in	51
WEBSOCKET.AddQueryParameter	Resource method	80
WEBSOCKET.ClearHeaders	Resource method	80
WEBSOCKET.ClearMessages	Resource method	80
WEBSOCKET.ClearQueryParameters	Resource method	81
WEBSOCKET.Close	Resource method	81
WEBSOCKET.Connect	Resource method	81
WEBSOCKET.Disconnect	Resource method	82
WEBSOCKET.Drain	Resource method	82
WEBSOCKET.Ping	Resource method	82
WEBSOCKET.Poll	Resource method	83

Entry	Section	Page
WEBSOCKET.Receive	Resource method	83
WEBSOCKET.ReceiveJSON	Resource method	83
WEBSOCKET.ReceiveText	Resource method	84
WEBSOCKET.RemoveHeader	Resource method	84
WEBSOCKET.RemoveQueryParameter	Resource method	84
WEBSOCKET.SendBinary	Resource method	85
WEBSOCKET.SendJSON	Resource method	85
WEBSOCKET.SendText	Resource method	85
WEBSOCKET.SetApiKey	Resource method	86
WEBSOCKET.SetBasicAuth	Resource method	86
WEBSOCKET.SetBearerToken	Resource method	86
WEBSOCKET.SetCookie	Resource method	87
WEBSOCKET.SetFeedToken	Resource method	87
WEBSOCKET.SetHeader	Resource method	87
WEBSOCKET.SetProtocols	Resource method	88
WEBSOCKET.SetQueryParameter	Resource method	88
WEBSOCKET.SetReconnect	Resource method	88
WEBSOCKET.SetSubprotocols	Resource method	89
WEBSOCKET.SetUrl	Resource method	89
WEBSOCKET.Snapshot	Resource method	89
WEBSOCKET.Wait	Resource method	90
WRITE	Built-in	52