Understanding CREXX Plugins

Power, Possibilities, and Pitfalls

Peter Jacob
unplugged, but not logged-off

What are CREXX Plugins?

- Extensions enhancing REXX scripts
- Dynamically integrate powerful C libraries
- Easily enable complex functionalities

Why Should You Care?

- Quickly add advanced capabilities
- Avoid rewriting core logic
- Boost efficiency and flexibility

Agenda

CREXX Plugin Architecture

CREXX - Plugin Architecture

CREXX/PA: Modular Plugin Architecture

Extend core system without modifying it.

Strict Decoupling

- Plugins interact via APIs, *never* internal code.

Function-Pointer Binding

- Dynamic runtime linking ensures zero hard dependencies.
  Flexible Linking, choose static (performance) or dynamic (flexibility).

Developer Benefits

- Macros/Helpers
  Hide boilerplate; focus on logic, not glue code.
- Auto-Discovery
  Plugins detected & invoked by compiler/runtime.
- Version Agnostic
  Maintain compatibility across core updates.

Anatomy of Plugin Functions

- **REXX** Function Call, Level B
- **Return Type**:
  .void, .int, .string,.float
- **Parameters:**
  int, string, float, .string[]

- **C Function**

PROCEDURE(c-function)
ADDPROC(c-function,rexx-function,“b“, return-type, „in-type-1, in-type-2“)

CREXX PLUGIN ARCHITECTURE

AGENDA

CREXX PLUGIN ARCHITECTURE

Writing a plugin feels like you've got it all figured out—until it doesn't.

Core Features of Plugin Architecture

- REXX SAA Standard?
- Interfacing with REXX and C Functions
- Step-by-Step Plugin Development

  show a simple plugin and how its parts fit together
- Defining C Functions
- Handling Parameters
- Working with Arrays
- Plugin Hacks & Tricks
- Various Examples
- Conclusion

CREXX/PA - Plugin Architecture

REXX Systems Application Architecture (SAA)

Reading/Writing REXX Variables
External programs can **read and manipulate REXX variables** via API calls.

File Operations: External programs can **create, read, and write files,** then either insert the contents into passed CREXX variables or output them from REXX

Database Access: External SAA-compliant programs provide interfaces to **databases** (e.g., DB2, SQL), enabling data processing and storing them into the passed CREXX variables.

Interactive User Interfaces (GUIs): External programs such as ISPF or custom GUIs allow REXX to interact with users and use the passed CREXX variables during this interaction.

only CREXX variables passed as parameters can be read and updated

Direct reading, updating and inserting of the CREXX variable pool not possible

↳ because there is no pool

CREXX plugins are fully SAA compatible—just like Schrödinger's cat is alive: technically, yes… until you check

CREXX/PA - Plugin Architecture

Why It is Incompatible

REXX Systems Application Architecture (SAA)

```
/*
 * rexx LEFT bif in rxas. Uses substr.rxas
 * no error checking
 * left: procedure = .string
 *       arg string = .string, length1 = .int, pad = ' '
 */
          .globals=0
substr()  .expose=rxfnsb.substr

left()    .locals=10 .expose=rxfnsb.left
   .meta "rxfnsb.left"="b" ".string" left() "string = .string, length1 = .int, pad = \' \'" ""
    ieq  r7,a2,0   /* return empty when length = 0 */
    brt  return_empty,r7
    brtpandt havepad,a3,1
    load a3," "
havepad:
    dec  r2
    load r3,4          /* there are 4 arguments            */
    swap r4,a1         /* the string                       */
    load r5,1          /* second argument, startpos == 1   */
    settp r5,1
    swap r6,a2         /* third argument, length           */
    settp r6,1
    swap r7,a3         /* fourth argument, pad             */
    settp r7,1
    call r1,substr(),r3 /* call substr                       */
    swap r4,a1         /* Swap back - pedantic             */
    swap r6,a2         /* Swap back - pedantic             */
    swap r7,a3         /* Swap back - pedantic             */
    ret r1
return_empty:
    ret "„
```

- CREXX does not have a variable pool; it operates solely with registers.

CREXX/PA - Plugin Architecture

Dynamic or Static

Dynamic

Built as a shared library (.rxplugin)
Plugins are dynamically loaded at runtime
loaded at runtime by dlopen(dlsym)
- LoadLibrary(GetProcAddress), or
- equivalent, hooking into the interpreter.

Static

Bundled directly into the final executable.

Special linker flags

to ensures plugin code is included
Inits at startup.
Better Performance

Plugin's Interaction with REXX Functions

Compiler Integration

- Compiler recognises plugin-registered functions
- Reads its meta-data and uses them

Procedure Registration

- Registered with CREXX core system when plugin is loaded
- Maps procedure name to corresponding C function

Calling from REXX

- REXX script calls procedure by its name
- CREXX interpreter looks up registered procedures
- Invokes corresponding C function

Parameter Passing

- Parameters passed from REXX script are accessible within C function
- Parameters automatically converted to appropriate types
- Based on definitions in ADDPROC statement

Return Values

- Delivered to calling CREXX

Plugin

Example

```
/* Simple Math Plugin */
options levelb
import simpleMath
import rxfnsb
say '-----------------------------------'
say ' Setting up a Plugin Sample'
say '-----------------------------------'
say "Simple Math Plugin test"
say '5 + 3    = 'add(5,3)
say '99 - 33  = 'subtract(99,33)
say '7 * 14   = 'multiply(7,14)
say '1024 : 8 = 'divide(1024,8)
```

The proof of the pudding is in the eating.

Plugin

Ex C

```c
/#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>         // For POSIX systems (Linux/macOS)
#include "crexxpa.h"        // crexx/pa - Plugin Architecture header file

PROCEDURE(add) {    // Function to add two integers
        int a = GETINT(ARG0);    // Get first integer from REXX
        int b = GETINT(ARG1);   // Get second integer from REXX
        RETURNINT(a + b);   // Return the sum
}
PROCEDURE(subtract) {       // Function to subtract two integers
        int a = GETINT(ARG0);   // Get first integer from REXX
        int b = GETINT(ARG1);   // Get second integer from REXX
        RETURNINT(a - b);       // Return the difference
}
PROCEDURE(multiply) {      // Function to multiply two integers
        int a = GETINT(ARG0);    // Get first integer from REXX
        int b = GETINT(ARG1);   // Get second integer from REXX
        RETURNINT(a * b);   // Return the product
}
PROCEDURE(divide) { // Function to divide two integers
        int a = GETINT(ARG0);   // Get first integer from REXX
        int b = GETINT(ARG1);   // Get second integer from REXX
        if (b == 0) RETURNINTX(-1); // Return -1 for division by zero error
        RETURNINT(a / b);        // Return the quotient
}
LOADFUNCS
   ADDPROC(add,     "simplemath.add",     "b", ".int","op1=.int,op2=.int");
   ADDPROC(subtract,"simplemath.subtract","b", ".int","op1=.int,op2=.int");
   ADDPROC(multiply,"simplemath.multiply","b", ".int","op1=.int,op2=.int");
   ADDPROC(divide,  "simplemath.divide",  "b", ".int","op1=.int,op2=.int");
ENDLOADFUNCS

```

Overview

- **Purpose of ADDPROC Statement**
  - Registers a C function as a callable procedure
  - Allows invocation from REXX scripts
- **Functionality**
  - Maps C function to a specific name
  - Enables core system to recognize and call the function
- **Interaction C Program and REXX**
  - **C-Program Definition**

```
ADDPROC(add,"simplemath.add","b", ".int","op1=.int,op2=.int");
```

C-Function

REXX-Function

Level-B

Return Type

Input Type per Parameter

  - **REXX**

```
say '5 + 3    = 'add(5,3)
```

  - **Compiler imported Definition**

```
add() .expose=simplemath.add
      .meta "simplemath.add"="b" ".int" add() "op1=.int,op2=.int"
```

Defining C-Function

- PROC / ENDPROC Keywords Used to define a procedure in the plugin

  Seamlessly embed and leverage the underlying RxPA data structures, providing macros and helper functions to simplify user interaction."

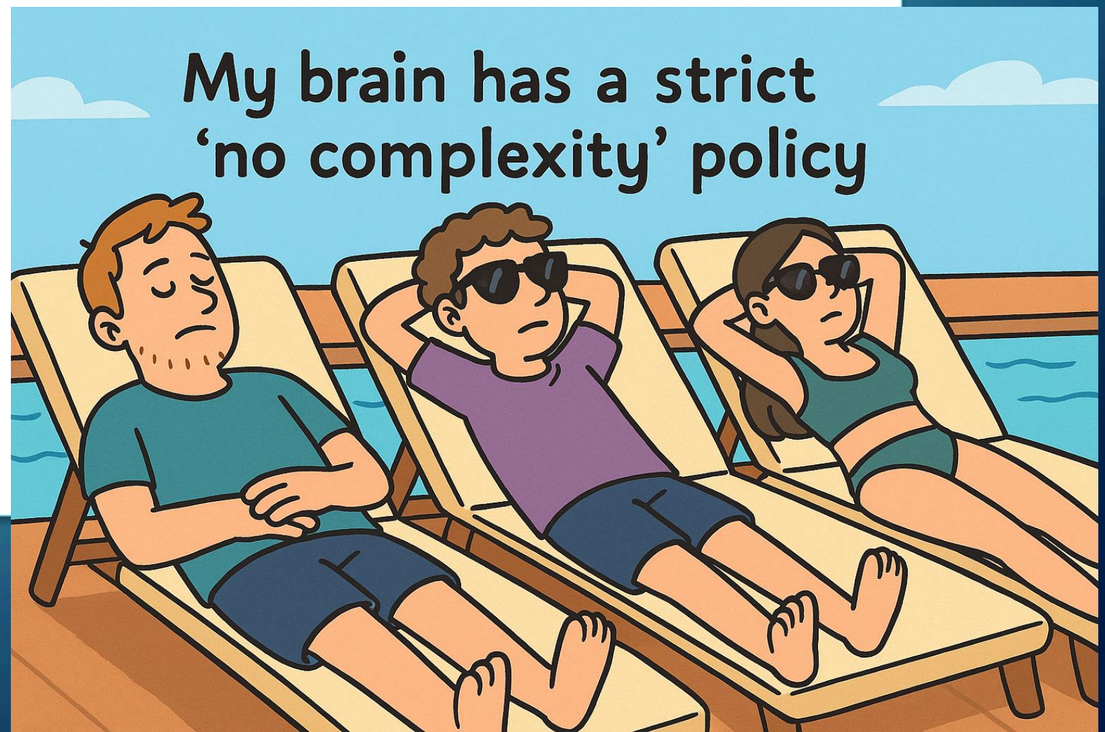
  - PROC begins the function definition
  - ENDPROC ends the function definition

- Example of Function Definition
  - PROC(MyFunction)
  - // Function implementation
  - ENDPROC

Addressing REXX Parameters

Helper Macros

- ARG0 first parameter
- ARG1 Second parameter
- ARG2 third parameter
- …
- **RETURN** **return parameter**
- **SIGNAL** **signal condition**



Reading input parameters

Read Input Parameter

- GETINT Function
  - Retrieves an integer value from the specified argument
  - Syntax: int value = GETINT(ARG0);
- GETSTRING Function
  - Retrieves a string value from the specified argument
  - Syntax: char* str = GETSTRING(ARG1);
- GETFLOAT Function
  - Retrieves a floating-point value from the specified argument
  - Syntax: float fvalue = GETFLOAT(ARG2);

Updating input variables

SETINT

SETSTRING

SETFLOAT

Write to Passed Parameters

- SETINT Function
  - Retrieves an integer value from the specified argument
  - Syntax: SETINT(ARG0,integer);
- SETSTRING Function
  - Retrieves a string value from the specified argument
  - Syntax: GETSTRING(ARG1,string);
- SETFLOAT Function
  - Retrieves a floating-point value from the specified argument
  - Syntax: SETFLOAT(ARG2,float);

ARRAYS I

GETARRAYHI

SETARRAYHI

GETSARRAY

SETSARRAY

SWAPARRAY

```
/* ---------------------------------------------------------------------------
* Shell Sort
* ---------------------------------------------------------------------------
*/
PROCEDURE (shell_sort) {
int from, to;
int i, j, gap;
char *val1, *order;
to = GETARRAYHI(ARG0);      // number of contained array items
order  = GETSTRING(ARG1);   // sort order
from = 1;
for (gap = to / 2; gap > 0; gap /= 2) {
    for (i = gap; i < to; i++) {
        val1 = GETSARRAY(ARG0, i);
        for (j=i;j>=gap && strcmp(GETSARRAY(ARG0,j-gap),val1) > 0; j -= gap) {
            SWAPARRAY(ARG0, j, j - gap);
        }
        SETSARRAY(ARG0, j, val1);
    }
}
ENDPROC
}

ADDPROC(shell_sort,"arrays.shell_sort","b",".void","expose a=.string[],offset=.int,order=.string");
```

ARRAYS II

GETIARRAY

SETIARRAY

GETFARRAY

SETFARRAY

INSERTARRAY

REMOVEARRAY

```
/* ---------------------------------------------------------------------------
 * Insert empty item(s) in an array and shift elements accordingly
 * ---------------------------------------------------------------------------
 */
PROCEDURE (insert_array) {
  int i, hi, from, new, del = 0, todel;
  hi = GETARRAYHI(ARG0) - 1; // make max index to offset
  from = GETINT(ARG1) - 1;     // get from offset
  new = GETINT(ARG2);        // lines to insert
  if (from < 0 || new < 1) {
    RETURNINT(0);
    PROCRETURN
    }
  if (from > hi) from = hi+1;
  for (i = from; i < from + new; ++i) {
    INSERTATTR(ARG0, i);

  RETURNINT(new);
  ENDPROC
}
```

Plugin Hacks

1. Plugin Loading and Unloading

- A plugin is loaded into a process when it is imported.
- It remains in memory within that process until the process terminates.
- Any global or static variables defined inside the plugin will retain their values between function calls, as long as the plugin remains loaded within the same process.

2. Multiple Calls from the Same Process

- When a program calls a function from a plugin multiple times, the plugin stays loaded in memory
- Global or static variables persist across calls within that same process.
- Any storage allocated persists also across calls

3. Multiple Processes Using the Same Plugin

- While the executable code (code sections) of a Plugin can be shared between processes, its data sections are not.
- When another process loads the same Plugin, the operating system creates a new instance of the Plugin that process's address space.
- Each process gets its copy of the Plugin's global and static variables, meaning changes in one process do not affect others.

Plugin Hacks Sample

Keep Matrix as global in DLL

```
#define mat(mptr,row,col) mptr.vector[row * mptr.cols + col]
#define matp(mptr,row,col) mptr->vector[row * mptr->cols + col]
// Helper macro for minimum value
#define MIN(a,b) ((a) < (b) ? (a) : (b))
struct Matrix {
    double* CBselfref;     // CB self reference
    char id[64];
    int rows;
    int cols;
    double * vector;  // Pointer to the matrix data
};

int matrixmax = MATRIX_MAX_COUNT;
void * allVectors[MATRIX_MAX_COUNT];
…
PROCEDURE(mtranspose) {
    int i, j, ii, jj, rows, cols, matnum, status;

    status = validateMatrix(GETINT(ARG0));
    if (status != MATRIX_VALID) RETURNINT(status);

    struct Matrix matrix = *(struct Matrix *) allVectors[GETINT(ARG0)];

    matnum = matcreate(matrix.cols, matrix.rows, 1,GETSTRING(ARG1));
    if (matnum < 0) RETURNINT(matnum);

    struct Matrix mtrans = *(struct Matrix *) allVectors[matnum];
    rows = matrix.rows;
    cols = matrix.cols;

    for (ii = 0; ii < rows; ii += MATRIX_BLOCK_SIZE) {// Block transpose for better cache usage
        for (jj = 0; jj < cols; jj += MATRIX_BLOCK_SIZE) {
            for (i = ii; i < MIN(ii + MATRIX_BLOCK_SIZE, rows); i++) {
                for (j = jj; j < MIN(jj + MATRIX_BLOCK_SIZE, cols); j++) {
                    mat(mtrans,j,i) = mat(matrix,i,j);
                }
            }
        }
    }

    RETURNINT(matnum);
```

Plugin

Fun

```
//
// System Information Plugin for crexx/pa - Plugin Architecture
//
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>   // For POSIX systems (Linux/macOS)
#include "crexxpa.h"    // crexx/pa - Plugin Architecture header file
#include <math.h>
#include <stdint.h>
#include <time.h>

/* ------------------------------------------------------------------
 * some Mathematical functions
 * * ------------------------------------------------------------------
 */
    MATHPROC(acos)
    MATHPROC(asin)
    MATHPROC(atan)
    MATHPROC(cos )
    MATHPROC(cosh)
    MATHPROC(exp )
    MATHPROC(exp2)
    MATHPROC(log )
    MATHPROC(log2)
    MATHPROC(log10)
    MATHPROC(ceil)
    MATHPROC(floor)
    MATHPROC(fabs)
    MATHPROC(round)
    MATHPROC(trunc)
    MATHPROC(sin )
    MATHPROC(sinh)
    MATHPROC(sqrt)
    MATHPROC(cbrt)
    MATHPROC(tan )
    MATHPROC(tanh)
    MATHPROC(erf)
    MATHPROC(erfc)
    MATHPROC(tgamma)
    MATHPROC(lgamma)
    MATHPROC(asinh)
    MATHPROC(acosh)
    MATHPROC(atanh)
```

```
// RXMATH function definitions
LOADFUNCS
    ADDMATH(acos)
    ADDMATH(asin)
    ADDMATH(atan)
    ADDMATH(cos)
    ADDMATH(cosh)
    ADDMATH(exp)
    ADDMATH(exp2)
    ADDMATH(log)
    ADDMATH(log2)
    ADDMATH(log10)
    ADDMATH(pow10)
    ADDMATH(ceil)
    ADDMATH(floor)
    ADDMATH(fabs)
    ADDMATH(round)
    ADDMATH(trunc)
     ADDMATH(sin)
    ADDMATH(sinh)
    ADDMATH(sqrt)
    ADDMATH(cbrt)
    ADDMATH(tan)
    ADDMATH(tanh)
    ADDMATH(erf)
    ADDMATH(erfc)
    ADDMATH(tgamma)
    ADDMATH(lgamma)
    ADDMATH(asinh)
    ADDMATH(acosh)
    ADDMATH(atanh)
ENDLOADFUNCS
```

Plugin Future Enhancements

But: The future hasn't been written yet. No one's has

Future Changes

- Support Address environments and variable "pool" access
- JSON Remote Plugin Support implementation (aka CREXXSAA)
- Additional Core Plugins (e.g.NCurses, SQLLite, Curl, etc)
- Object and Decimal Support

CREXX Plugin Factory

Enough is never enough for those who have enough

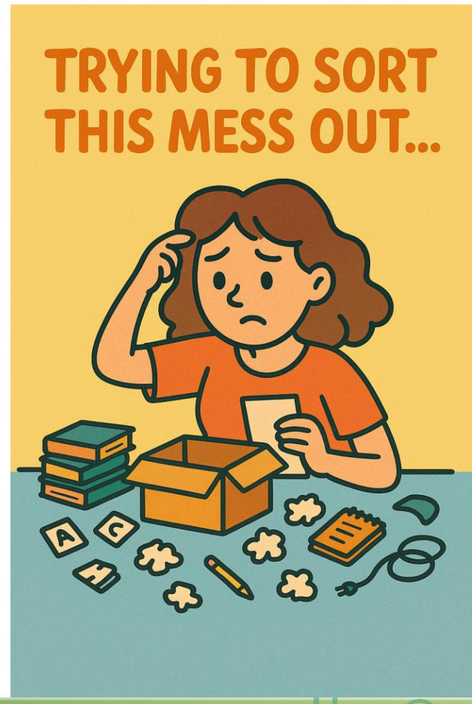
– Epicurus

Where plugins are born. And where sanity goes to retire.

CREXX Plugin Examples

These plugin samples are experiments—proof-of-concept demos. They're not finished products, but they'll give you a taste of what's possible.

Brought to you by coffee and regret, and hundreds of cigarettes – if I smoked.



1.Connection Management

odbc_connect(dsn, username, password)

Establishes connection to database or CSV files

Returns: 0 (success) or negative values (error codes)

Example: rc = odbc_connect("myDB", "user", "pass")

odbc_database(newdb)

Gets or sets the current database (MySQL-specific)

Important: Must be called after connecting

Example: newdb = odbc_database("CompanyDB")

odbc_disconnect()

Closes connection and frees resources

Example: call odbc_disconnect

2.Query Execution

odbc_execute(sql)

Executes SQL statements

Returns: 0 (success) or negative values (error codes)

Example: rc = odbc_execute("SELECT * FROM employees")

3.Result Set Navigation

odbc_fetch(row)

Fetches next row or specific row number

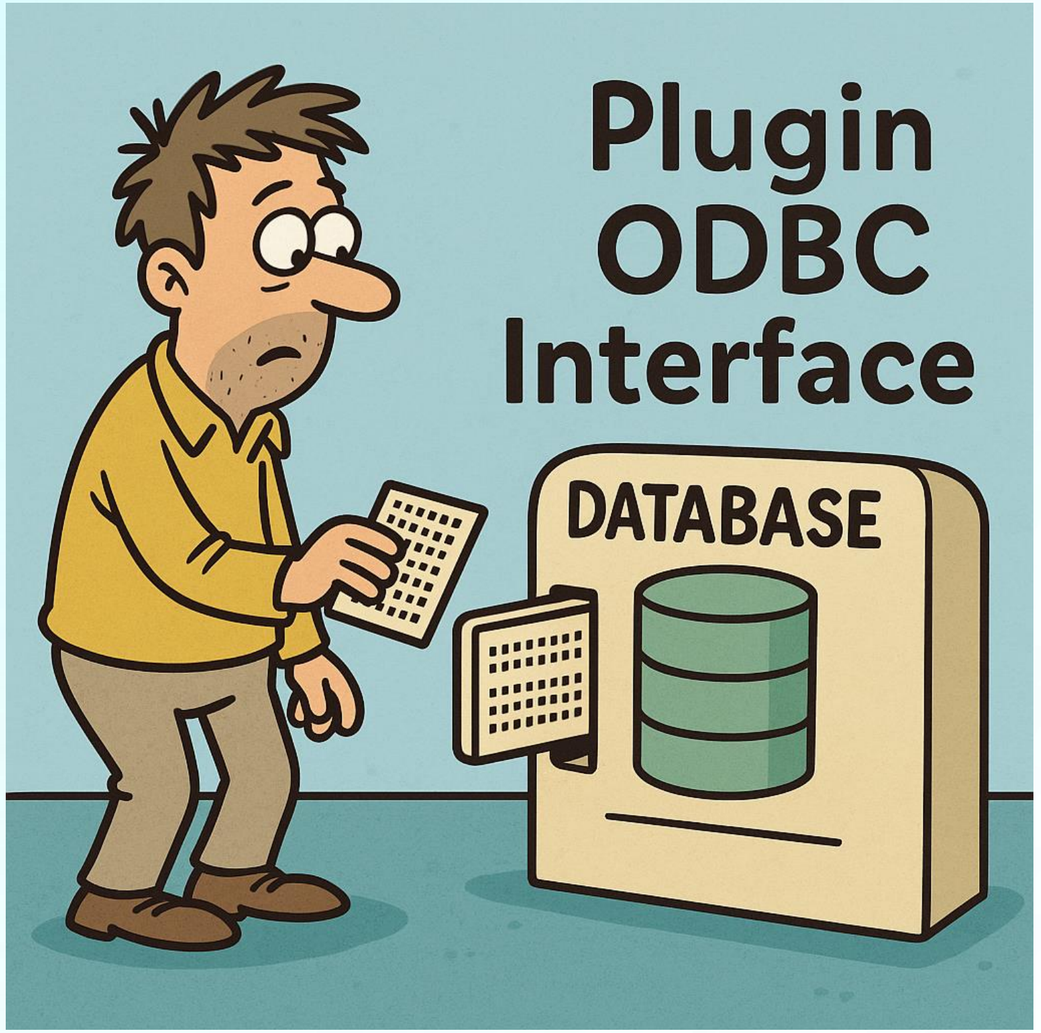
Returns: 0 (success), -1 (no more rows), -2 (error)

Example: do while odbc_fetch(0) >= 0

odbc_getcolumn(column)

Retrieves value from specific column in current row

Example: value = odbc_getcolumn(1)



4. Metadata Functions

odbc_tables()

Returns space-separated list of tables in current database

Example: tables = odbc_tables()

odbc_primary_keys(table)

Returns space-separated list of primary key columns

Example: keys = odbc_primary_keys("employees")

odbc_columns()

Returns number of columns in result set

Example: cols = odbc_columns()

5. Transaction Management

odbc_begin_transaction()

Starts a new transaction by turning off autocommit.

odbc_commit()

Commits the current transaction.

odbc_rollback()

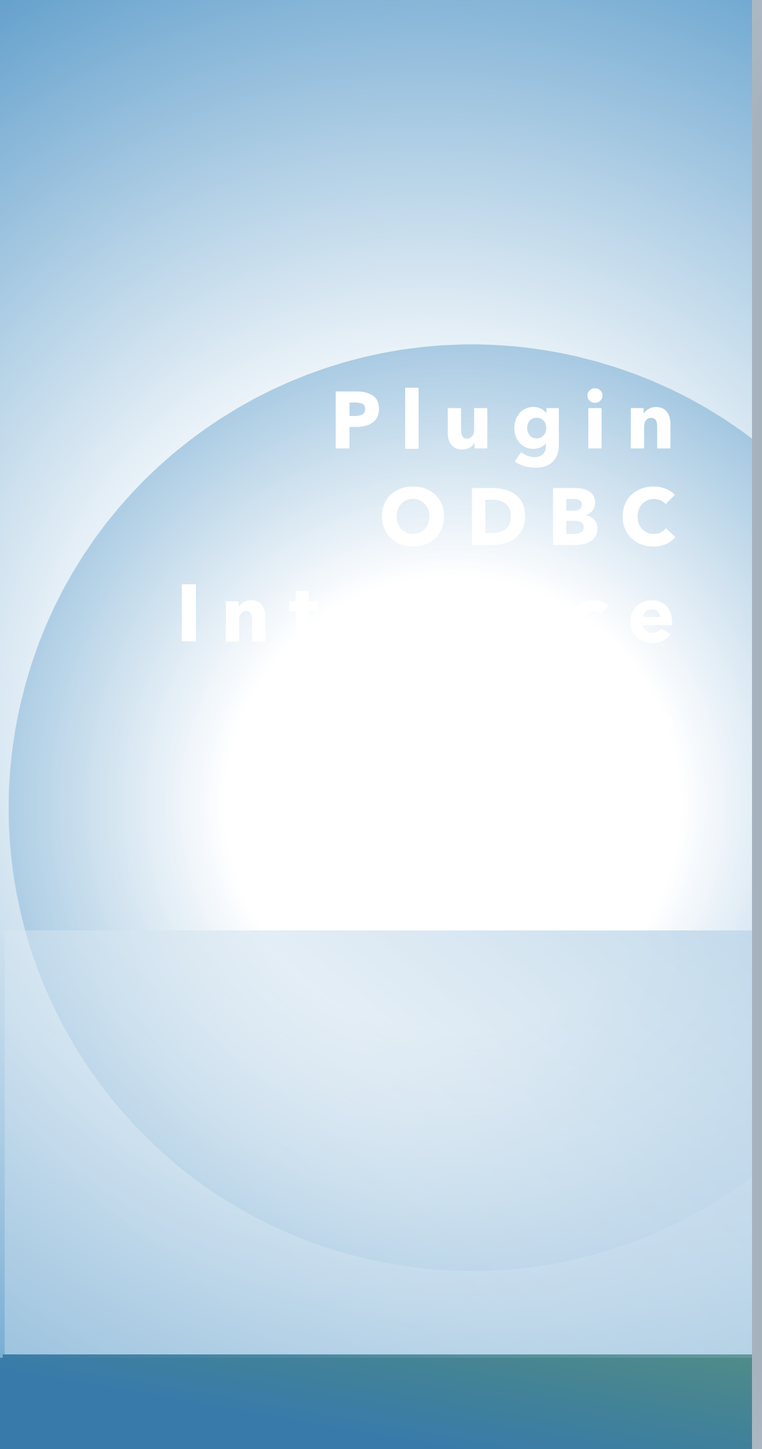
Rolls back the current transaction.

6. Error Handling

odbc_error_message()

Gets the last error message from ODBC.

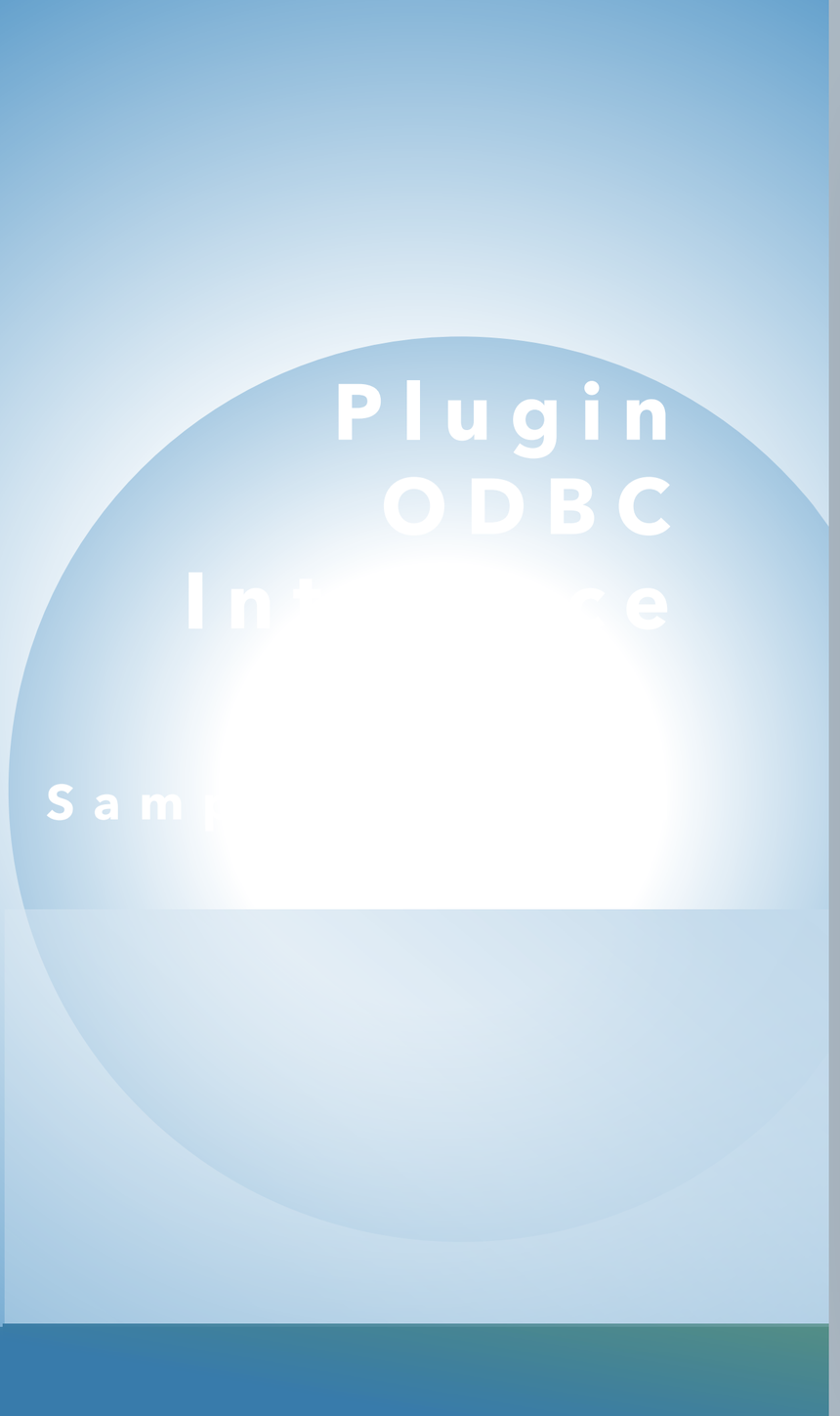
Plugin ODBC Interface




```
/* ODBC Sample */
options levelb
import odbc
import rxfnsb
database='CompanyDB'
table    =database'.Employees'
/* Connect to Database */
/* ------------------------------------------------------------------
 * Connect to Database
 * ------------------------------------------------------------------
 */
say "*** Connect to Database..."
  rc = odbc_connect("myCustomDB", "root", "pjjp")
  if rc < 0 then call Error_exit "Connection failed: "odbc_error_message(),
"Full diagnostics: "odbc_get_diagnostics()
  dbinfo = odbc_get_info()
  say "Connected to database system:" dbinfo
/* ------------------------------------------------------------------
 * Explicitly set the database
 * ------------------------------------------------------------------
 */
say "*** set/change Database..."
  newdb = odbc_database('customer')
  if newdb = "Error changing database" then call error_exit "Failed to set
database: "odbc_error_message(),""
  say "Database set to: '"newdb"'"
  /* Explicitly set the database */
  newdb = odbc_database(database)
  if newdb = "Error changing database" then call error_exit "Failed to set
database: "odbc_error_message(),""
  say "Database set to: '"newdb"'"
/* ------------------------------------------------------------------
 * Execute SELECT * FROM table and retrieve column information
 * ------------------------------------------------------------------
 */
say "*** Column Information Example..."
  rc = odbc_execute("SELECT * FROM "table)
  if rc < 0 then call error_exit "Query failed: "odbc_error_message(),""

  cols = odbc_columns()
  do col = 1 to cols
     colname = odbc_colname(col)
     coltype = odbc_coltype(col)
     say "Column" col": Name =" colname "Type =" coltype
  end
```

Plugin
ODBC
Int[illegible]ce
Sam[illegible]

```
*** Connect to Database...
Available Driver: SQL Server
Available Driver: MySQL ODBC 9.2 ANSI Driver
Available Driver: MySQL ODBC 9.2 Unicode Driver
Available Driver: Microsoft Access Driver (*.mdb, *.accdb)
Available Driver: Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
Available Driver: Microsoft Access Text Driver (*.txt, *.csv)
Available Driver: Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
Connected to database system: MySQL
*** set/change Database...
Database set to: 'customer'
Database set to: 'companydb'
*** Column Information Example...
Table has 5 Columns
--------------------
Column 1: Name=Employee_ID, Type=4, Size=10 foundCols=5
Column 2: Name=Name, Type=-9, Size=100 foundCols=5
Column 3: Name=Age, Type=4, Size=10 foundCols=5
Column 4: Name=Department, Type=-9, Size=50 foundCols=5
Column 5: Name=Salary, Type=4, Size=10 foundCols=5
--------------------
Column 1: Name = Employee_ID Type = 4
Column 2: Name = Name Type = -9
Column 3: Name = Age Type = 4
Column 4: Name = Department Type = -9
Column 5: Name = Salary Type = 4
```




```
    SELECT Name, Salary FROM CompanyDB.Employees WHERE Salary > 60000)
Table has 2 Columns
--------------------
Column 1: Name=Name, Type=-9, Size=100 foundCols=2
Column 2: Name=Salary, Type=4, Size=10 foundCols=2
--------------------
            Name          Salary
---------------------------------
      Daniel Lee           85000
     Olivia White          64000
      Liam Martin          61000
    Emma Thompson          78000
     James Harris          77000
      Lucas Clark          71000
     Amelia Young          82000
     Henry Walker          65000
   Isabella Allen          72000
    Benjamin King          68000
      Oliver Hill          88000
        Mia Adams          67000
     Elijah Baker          86000
      Aria Carter          73000
William Mitchell          75000
     Harper Perez          62000
    Jackson Evans          61000
```

Plugin

Windows

ets

3

**Command line in the front, GUI in the back–
hardcore on the outside, comfy on the inside**

Overview

The CREXX GUI Framework provides a lightweight graphical user interface for CREXX applications using GTK. It offers essential widgets and functions for building interactive applications with minimal code.

Key Components

Window Management

- init_window: Creates the main application window
- show_window: Displays the window to users
- process_events: Handles user interactions and updates

Basic Widgets

- **Buttons:** add_button creates clickable buttons
- **Text Labels:** add_text displays static text
- **Input Fields:** add_edit allows text entry
- **Checkboxes:** add_checkbox for boolean options
- **Lists:** add_list for displaying multiple items
- **Combo Boxes:** add_combo for dropdown selections

Plugin

Windows

W i t s

U t

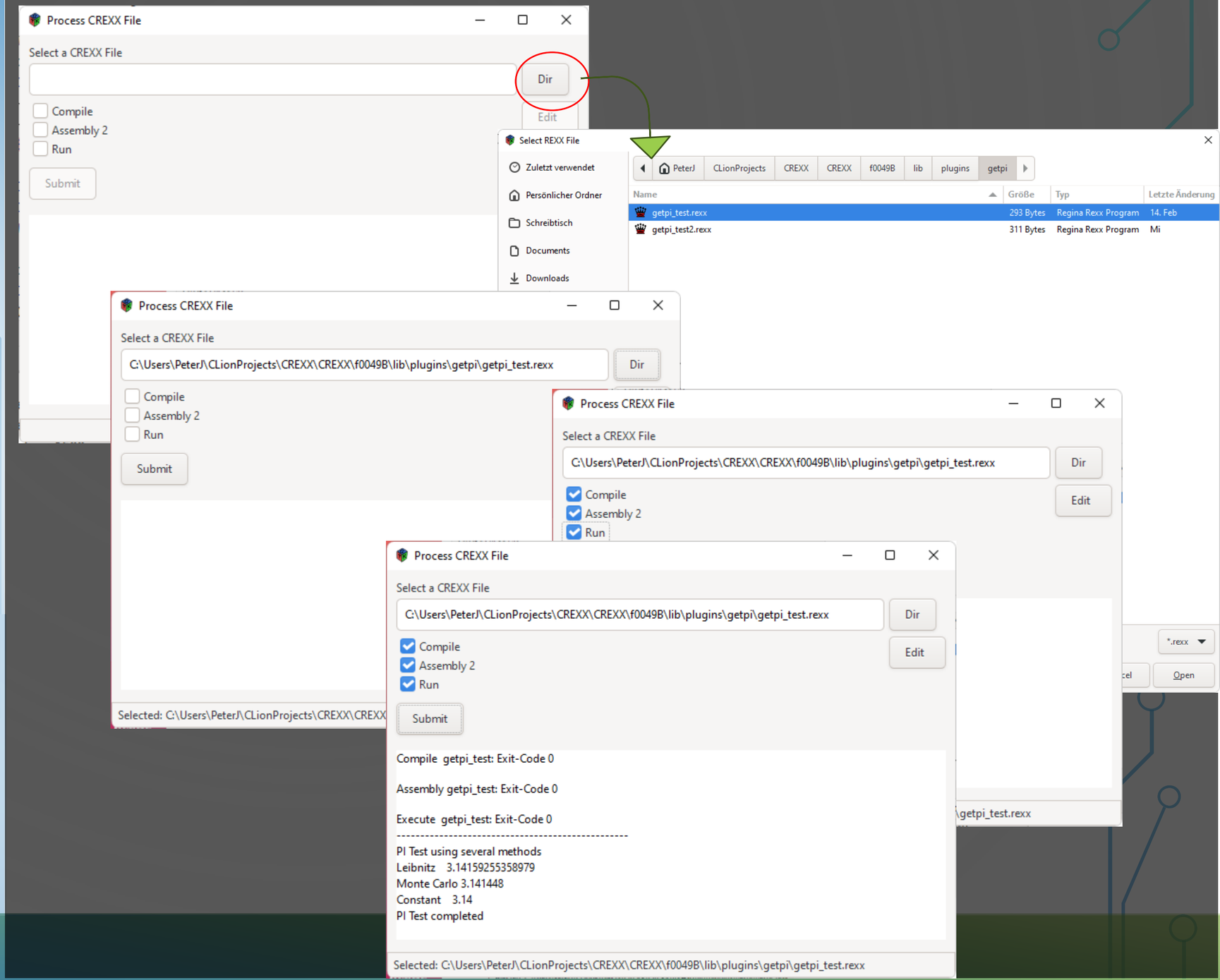
```
options levelb
import gui
import rxfnsb


home="C:\Users\PeterJ\CLionProjects\CREXX\CREXX\f0049B\"
say Substr("GETPI Test",3,4)
pluginhome=home"\lib\plugins\"
/* ---------------------------------------------------------------------------------------------
 * Initialize splash screen and store return value
 * ---------------------------------------------------------------------------------------------
 */
  splash_rc = splash_pick("Let's do the Time Warp again!", ,
                          "it's a jump to the left and then a step to the right", ,
                          3, 500,500, ,
                          "CREXX.png")
/* Initialize the GUI */
  call init_window "Process CREXX File", 600, 430
/* Add a label for the input field */
  label_index = add_text("Select a CREXX File ", 10, 10)
/* Add an entry field for input */
  input_field_index = add_edit(10, 30,515)
/* Add a small button next to the input field */
  select = add_button("Dir", 530, 30)
  update = add_button("Edit  ", 530,70)
/* Add checkboxes */
  checkbox1_index = add_checkbox("Compile", 10, 70)
  checkbox2_index = add_checkbox("Assembly 2", 10, 90)
  checkbox3_index = add_checkbox("Run", 10, 110)
  submit = add_button("Submit", 10, 140)
  message_area = add_message_area(10, 190, 580, 200)  /* x, y, width, height */
  status_bar = add_status_bar()
/* Show the window */
  call show_window -1,-1
## hidemsg=hide_widget(submit)
  call set_sensitive submit,0
  call set_sensitive update,0
/* ---------------------------------------------------------------------------------------------
 * Event Handler
 * ---------------------------------------------------------------------------------------------
 */
do forever
    event_token = process_events(500)  /* Wait for events with a timeout of 500ms */
    if event_token < 0 then leave      /* Exit loop on timeout */
    else if event_token = select then do
       input_rexx = file_pick('Select REXX File', pluginhome,0,"*.rexx")
       if input_rexx <> "" then do     /* Check if the input text is not empty */
          call set_edit input_field_index, input_rexx
          call set_status status_bar, "Selected: "input_rexx
          ## call show_widget(submit)
          call set_sensitive submit,1
          call set_sensitive update,1
       end
    end
    else if event_token = submit then do
       call submitit input_rexx,home,message_area
    end
    else if event_token = update then do
       call run_sync "C:\Program Files\Notepad++\Notepad++.exe",input_rexx,""
    end
end
exit 0
```

Plugin

Windows

Wi ts

Ut



Plugin

Picker

Ut

For those who love command line parameters—until the 10th one hits. After that, it's like a treasure hunt with a broken map. Time to call in the picker service!

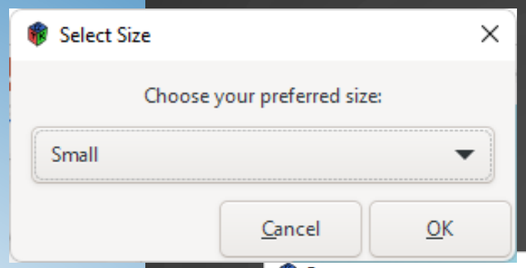
CREXX Picker Dialog Services

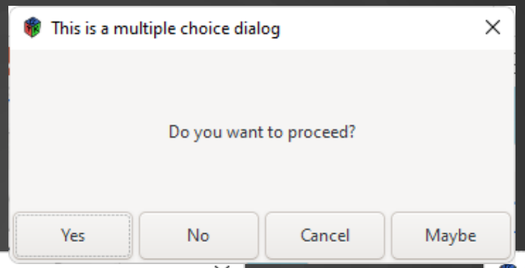
Table of Contents

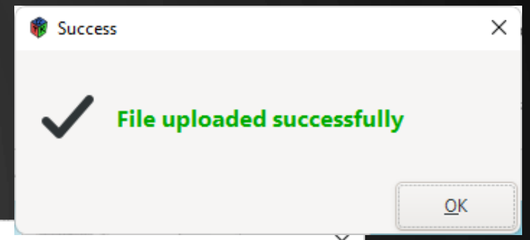
Plugin

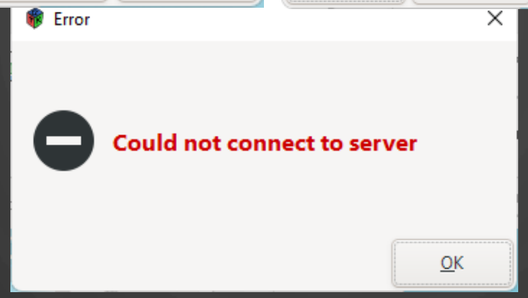
Picker

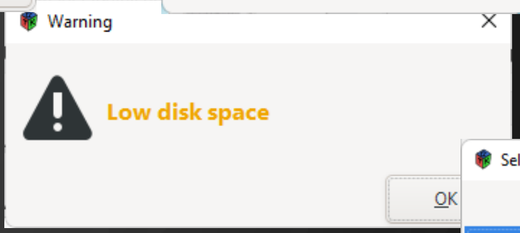
Ut

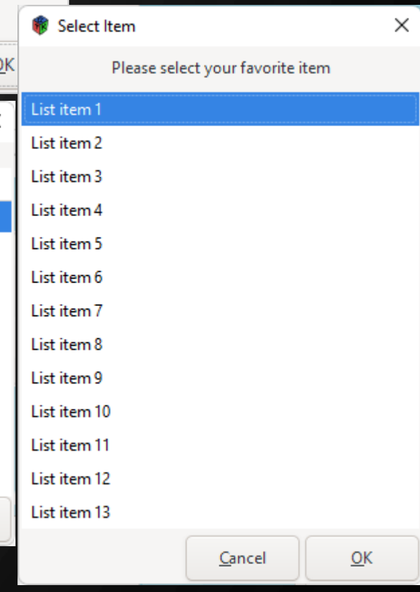


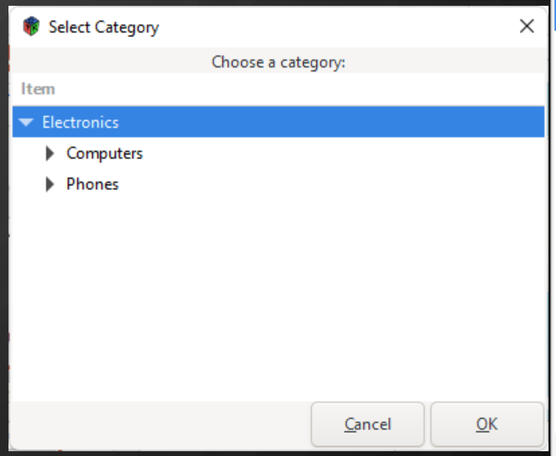


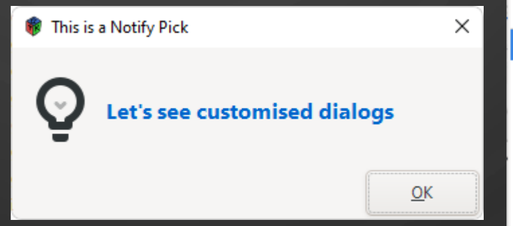


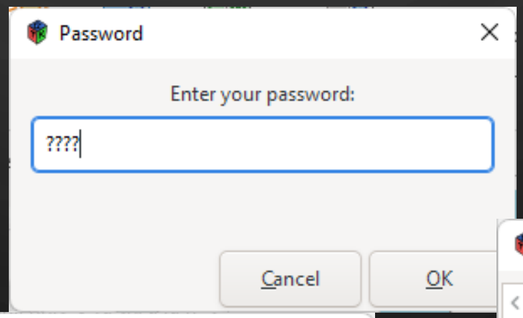


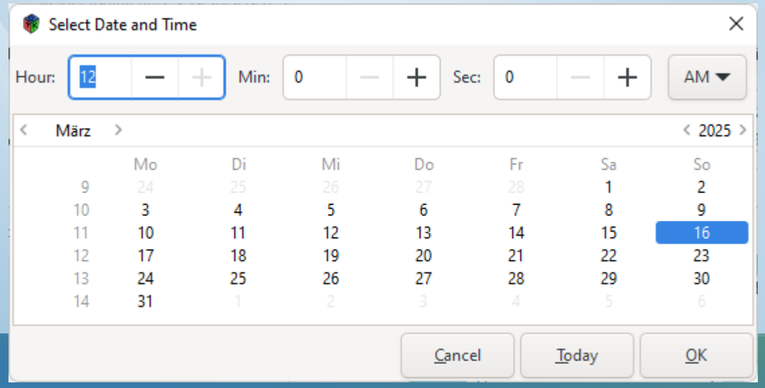


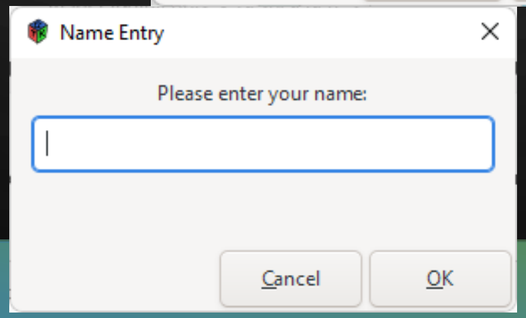


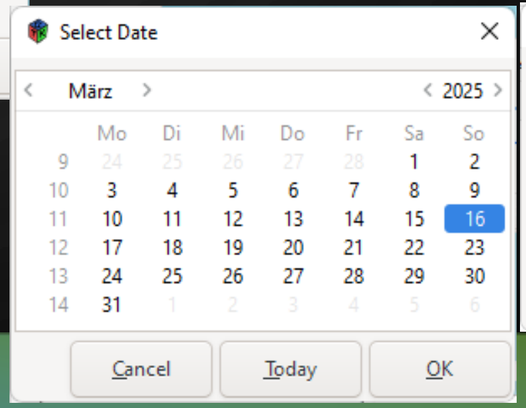


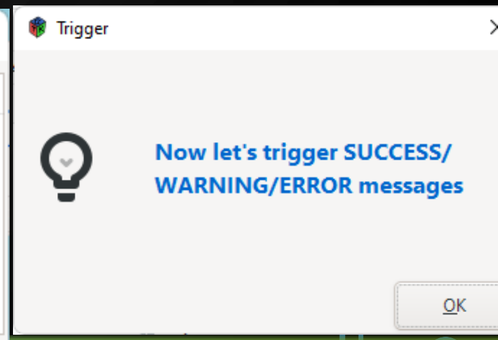








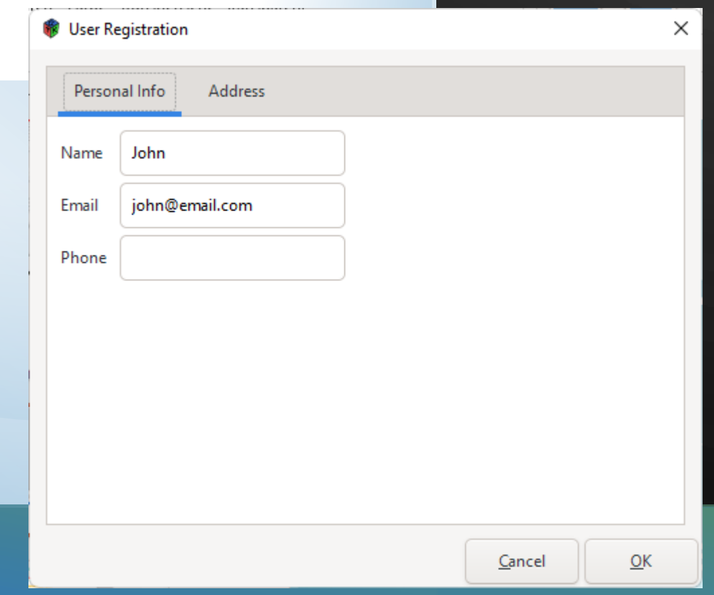


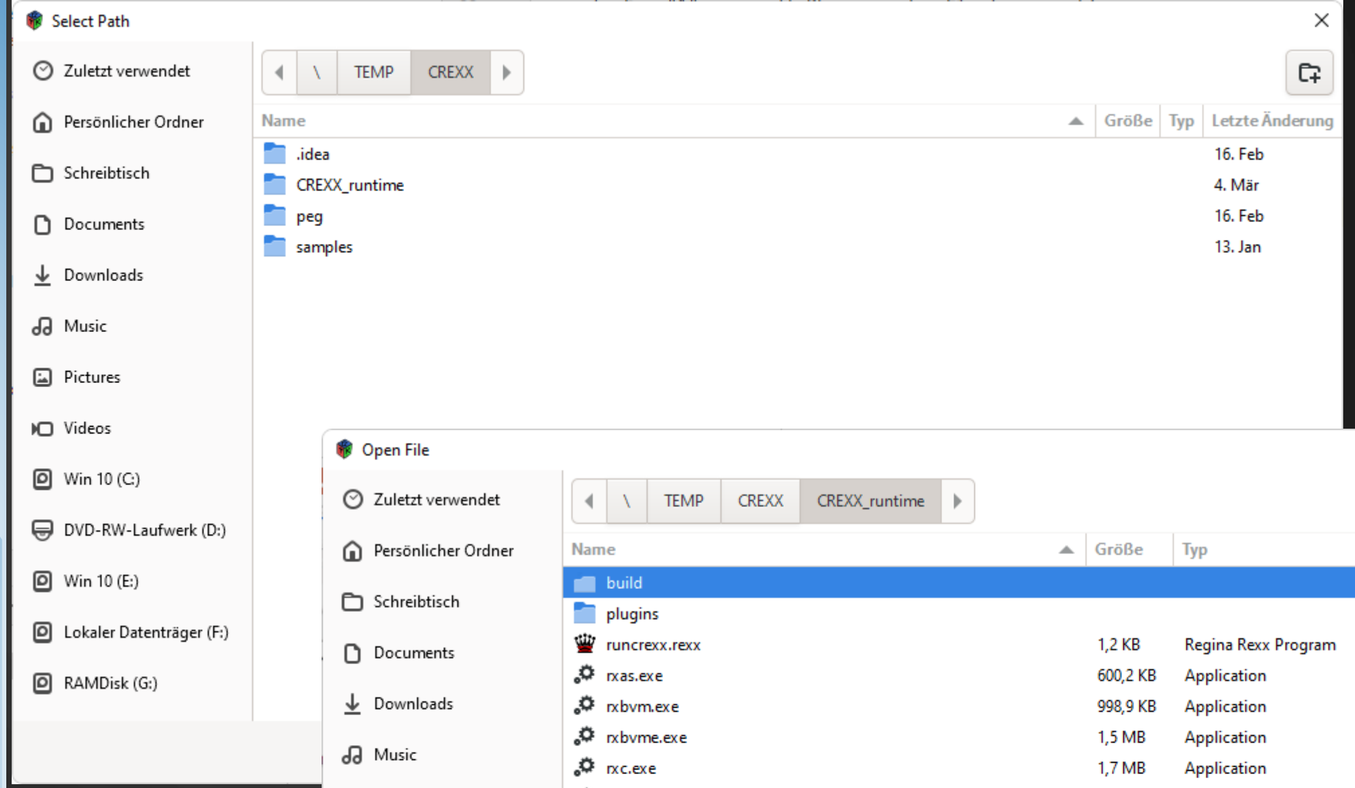


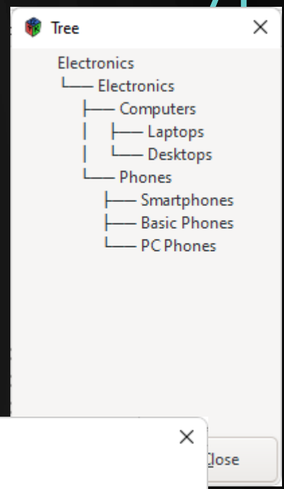
Plugin

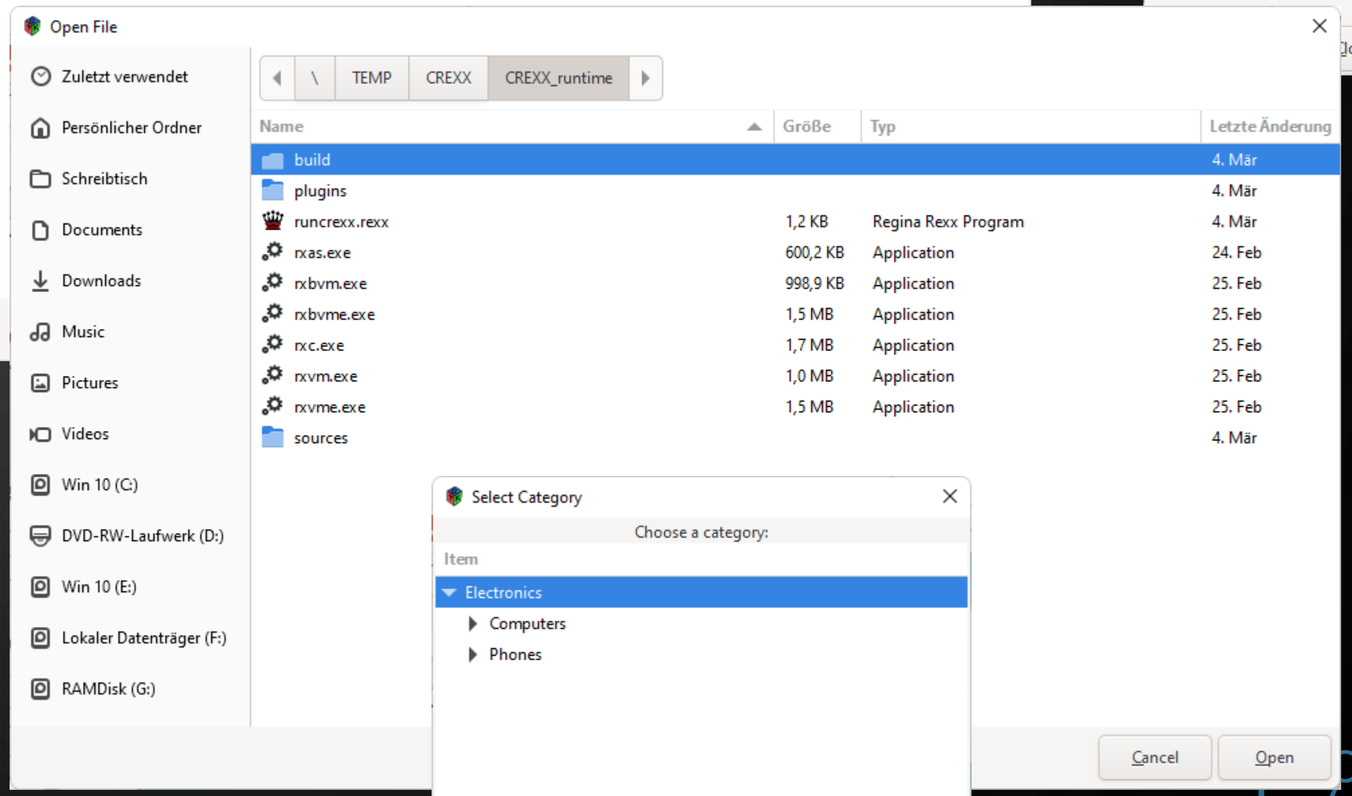
Picker

Ut









Plugin

Pi ces

Buckle up. W ed roads.

Plugin

TCP/IP Client

```
/* TCP Test */
options levelb
import rxtcp
import rxfnsb
/* ---------------------------------------------------------------------------
 * Sample CREXX TCP Client
 * ---------------------------------------------------------------------------
 */
say tcpflags("Debug")

socket=tcpopen("127.0.0.1",3033)
say "Socket: "socket
say "TCPSEND    : "tcpsend(socket,"Hello Server, I am a CREXX script")
record=TCPRECEIVE(socket,10)                    ## timeout in milliseconds
say "TCPRECEIVE : Length: "length(record) record

say "TCPSEND    : "tcpsend(socket,"How is it going")
record=TCPRECEIVE(socket,1000)                    ## timeout in milliseconds
say "TCPRECEIVE : Length: "length(record) record

say "TCPSEND    : "tcpsend(socket,"I am happy to meet you")
record=TCPRECEIVE(socket,10)                    ## timeout in milliseconds
say "TCPRECEIVE : Length: "length(record) record

say "TCPSEND    : "tcpsend(socket,"Sorry, I must leave now")
record=TCPRECEIVE(socket,10)                    ## timeout in milliseconds
say "TCPRECEIVE : Length: "length(record) record

say "TCPCLOSE   : "TCPCLOSE(socket)
```

Plugin

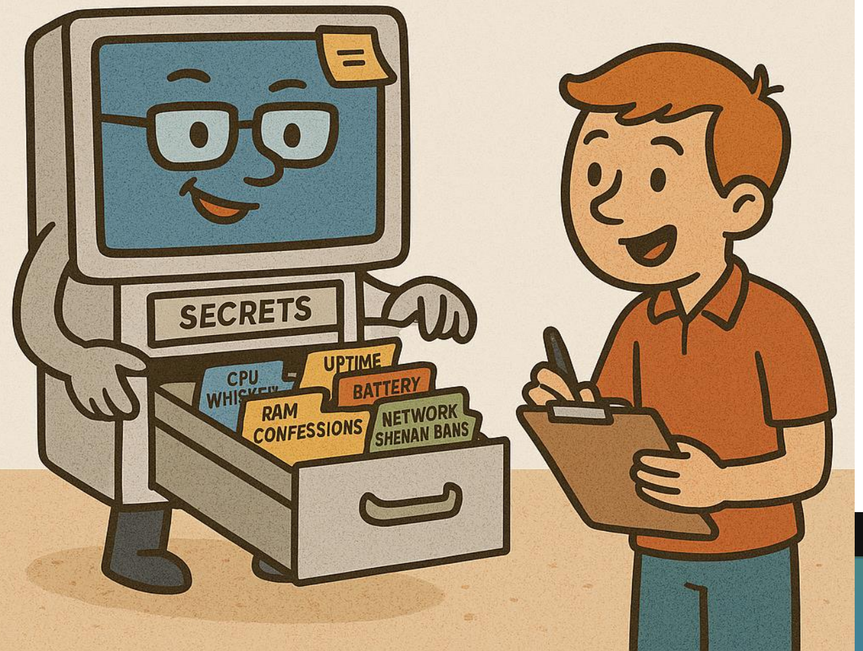
TCP


```
/* TCP Test */
options levelb
import rxtcp
import rxfnsb

say tcpflags("Debug")

sockets.1=""
server=TCPSERVER(3090,sockets)
if server<0 then do
   say "Server Create faile with rc="server
   exit
end
say "Server Socket "server
i=0
do forever
   i=i+1
   if i>100 then leave
   client=TCPWAIT(server,100,sockets)
   if client>0 then do           ## -4 is timeout occurred, nothing new
      say "Client Socket "client
      record=TCPRECEIVE(client,100)                    ## timeout in milliseconds
      say "TCPRECEIVE : Length: "length(record) record  ## receive record of client
      say "TCPSEND    : "tcpsend(client,"Hello Client, I am a CREXX server")  ## reply to client
      say "** Active clients **"
      do j=1 to sockets.0
         say "current open sockets "j" "sockets.j
      end
   end
   call wait 100
end
say "TCPCLOSE   : "TCPCLOSE(server)
exit
```

Plugin

System Functions

Accessing System Info Like a Pro



System Functions Documentation

This documentation includes both system-level functions and file system operations. Although most of them are designed to support multiple operating systems, they are currently activated only for Windows.

GETENV

Retrieves the value of a system environment variable.

Syntax: `value = getenv(env_name)`

Parameters:

- `env_name` - String containing the environment variable name

Returns:

- `string` - Value of the environment variable if found
- `SIGNAL_FAILURE` - If variable not found or invalid argument

Example: `rexx home = getenv("HOME") path = getenv("PATH")`

GETDIR

Returns the current working directory path. Works cross-platform on Windows, Linux, and macOS.

Syntax: `path = getdir()`

Parameters:
None

Returns:

- `string` - Current working directory path
- `SIGNAL_FAILURE` - If unable to get directory

Example: `rexx`

SETDIR

Changes the current working directory path. Works cross-platform on Windows, Linux, and macOS.

Syntax: `rc = setdir(new-current-working-directory)`

Parameters:
None

Returns:

- 0 - Current working directory path has been changed
- -8 - unable to change the directory

TESTDIR

Tests if a directory exists and is accessible. Path separators are automatically normalized.

Syntax: `rc = testdir(directory-to-check)`

Parameters:

- `directory` - String containing the directory path to test

Returns:

- `0` - Success: Directory exists
- `-8` - Error: Directory does not exist or is not accessible

Example:

```
if testdir("/path/to/check") == 0 then say "Directory exists"
```

CREATEDIR

Creates a new directory. On Unix systems, creates with permissions 0755.

Syntax: `rc = createdir(directory)`

Parameters:

- `directory` - String containing the directory path to create

Returns:

- `0` - Success: Directory created
- `-4` - Error: Directory already exists
- `-8` - Error: Unable to create directory

Example:

```
rc = createdir("new_folder")
```

RENAMEFILE

Renames or moves a file from the source path to the target path. Works across directories on the same filesystem.

Syntax: `rc = renamefile(oldname, newname)`

Parameters:

- `oldname` - String containing the source file path/name
- `newname` - String containing the target file path/name

Returns:

- `0` - Success: File was renamed
- `-4` - Error: Source file does not exist
- `-8` - Error: Rename operation failed (target exists, permission denied, etc.)

Example:

```
rc = renamefile("old.txt", "new.txt")
```

Plugin

DELETEFILE

Deletes the specified file. Path separators are automatically normalized.

Syntax: `rc = deletefile(filename)`

Parameters:

- `filename` - String containing the file path to delete

Returns:

- `0` - Success: File deleted
- `-3` - Error: Permission denied
- `-4` - Error: File does not exist
- `-5` - Error: File is in use
- `-8` - Error: Other error

Example:

```
rc = deletefile("unwanted.txt")
```

SLEEP

Pauses execution for the specified duration in milliseconds.

Syntax: `rc = sleep(milliseconds)`

Parameters:

- `milliseconds` - Integer containing the time to sleep in milliseconds

Returns:

- `0` - Success: Sleep completed
- `-1` - Error: Invalid argument

OPSYS

Returns the current operating system platform identifier.

Syntax: `platform = opsys()`

Parameters:
None

Returns:

- `string` - Platform identifier ("WINDOWS", "LINUX", "macOS", etc.)
- `unknown` - If unable to determine platform

Example:

```
if opsys() == "WINDOWS" then say "Running on Windows"
```

HOST

Returns the computer name of the current system.

Syntax: `name = host()`

Parameters:
None

Returns:

- `string` - System hostname
- `unknown` - If unable to get hostname

Example:

```
say "Computer name:" host()
```

USERID

Returns the username of the current user.

Syntax: `name = userid()`

Parameters:
None

Returns:

- `string` - Current username
- `unknown` - If unable to get username

Example:

```rexx say "Current user:" userid()
```

Plugin

Regular Expressions

REGEX

A Pinch of Quantum Physics: Where the Match Exists

and Doesn't Exist at the Same Time

Regex Plugin Documentation

Overview

The regex plugin provides regular expression functionality and string distance calculations for CREXX/PA. It includes pattern matching, compilation, and string distance metrics.

Functions

RXCOMPILE(pattern, flags)

Compiles a regular expression pattern.

Parameters:

- pattern: String containing the regular expression
- flags: Integer containing compilation flags
  - RX_BASIC (0): Basic Regular Expressions (BRE)
  - RX_EXTENDED (1): Extended Regular Expressions (ERE)
  - RX_ICASE (2): Case insensitive matching
  - RX_NEWLINE (4): Honor newline as special character
  - RX_NOSUB (8): Only report success/failure

Returns:

- handle: Positive integer handle on success
- Negative error code on failure:
  - RX_ERROR_PARAM (-1): Invalid parameters
  - RX_ERROR_MEMORY (-2): Memory allocatio
  - RX_ERROR_COMPILE (-3): Pattern compilati

RXMATCH(handle, string, flags)

Matches a compiled pattern against a string.

Parameters:

- handle: Handle returned by rxcompile
- string: String to match against
- flags: Integer containing match flags
  - RX_NOTBOL (16): ^ doesn't match beginning of string
  - RX_NOTEOL (32): $ doesn't match end of string

Returns:

- 1: Match found
- 0: No match
- Negative error code on failure

RXFREE(handle)

Frees resources associated with a comp

Parameters:

- handle: Handle returned by rxcomp

Returns:

LEVENSHTEIN(string1, string2)

Calculates Levenshtein (edit) distance between two strings.

Parameters:

- string1: First string
- string2: Second string

Returns:

- Positive integer representing edit distance
- -1: Invalid parameters
- -2: Memory allocation error

HAMMING(string1, string2, uppercase)

Calculates Hamming distance between two equal-length strings.

Parameters:

- string1: First string
- string2: Second string
- uppercase: Convert to uppercase before comparison (1=yes, 0=no)

Returns:

- Positive integer representing Hamming distance
- -1: Invalid parameters
- -2: Strings must be equal length

Plugins: This is the End... Or Just the Beginning?

Interaction with C Functions:

REXX scripts can seamlessly call plugin functions using names defined in ADDPROC statements, extending functionality.

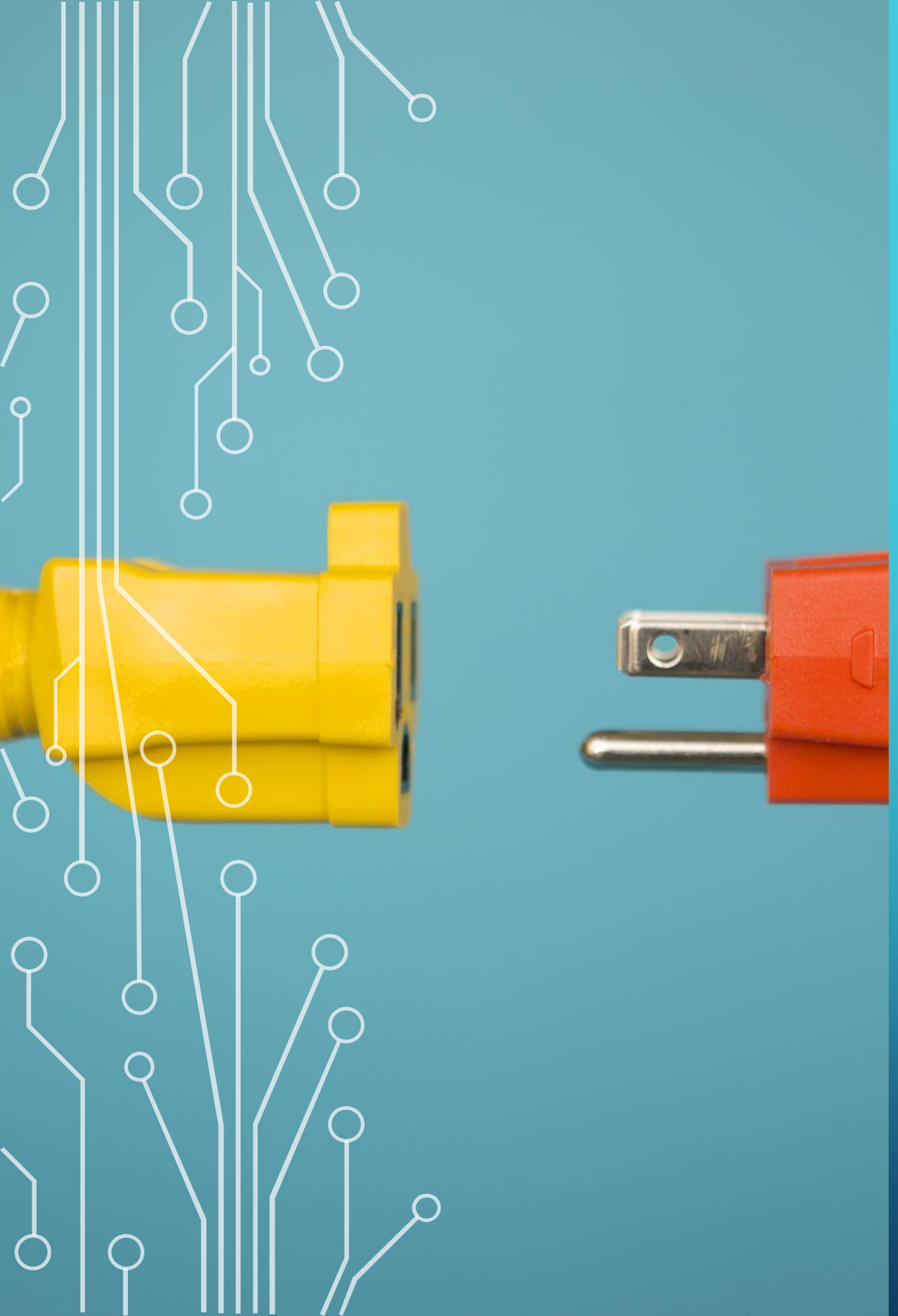
Expansive Capabilities:

Beyond just database operations, the plugin unlocks a wide range of powerful features, enabling complex tasks and offering unparalleled flexibility for developers.

Versatile Tool for Developers:

The plugin provides a broad spectrum of functionalities, from data management to advanced integrations, empowering developers to tackle a variety of challenges with ease.

Warning: In the end, you might just go insane trying to grasp all the possibilities



FINAL THOUGHTS

CREXX Plugins give REXX scripts superpowers—fast, flexible

- What we covered:
  - How plugins extend REXX
  - The anatomy of a plugin
  - Examples and experiments

Any Questions Before We're All Replaced by Deepfake Consultants?

Ask now, while your face still belongs to you.

```
>
```

"This is the threat AI hurled at me when I asked one too many questions."