THE and cREXX

A live editor integration story, rendered now as a paper-record deck.

- Syntax highlighting through DSLSH
- cREXX macros through crexxsaa
- Source left, output or RXAS right

Paper record

The live presentation was itself a cREXX macro running inside THE.

The Demo Was The Product

The editor drove the talk: navigation, source, output, refresh, and RXAS view.

05-highlight-sample.rexx

```
options levelb
import rxfnsb

say "!CParser-backed highlighting!N"
items = .string[]
items[1] = "profile"
items[2] = "macro"

select
  when total > 10 then say "!Ginteresting!N"
```

output

```
Parser-backed highlighting

 1 : profile
 2 : macro
 3 : source buffer
 4 : output buffer

select/when says total is interesting: 15
```

- R F6 saves and reruns
- R F5 toggles output/RXAS
- R F9/F10 move slides
- R F4 changes layout

SDSLH Made Highlighting External

THE delegates parsing, then paints semantic token ranges in the file area.

THE buffer
file identity + edits

SDSLH process
registered by SET SDSLH

parser
rxc, rxas, mdp, dslsh-c, pyp, jsp

tokens
semantic ranges

THE paint
ECOLOR areas

Per-file lifecycle

Parser state follows the editor file, not a global previous parse.

Configurable colours

`SET ECOLOR keyword`, `string`, `comment`, `type`, `function`.

Presentation stress test

Markdown, cREXX, RXAS, generated buffers, and panes changed rapidly.

Profile Mappings Became Ordinary Configuration

The presentation profile wires the editor to each parser.

File type	Parser registration	Autocolor mapping
.rexx / .rex / .the	SET SDSLH rxc ... --syntaxhighlight	SET AUTOCOLOR *.rexx rxc
.rxas	SET SDSLH rxas ... --syntaxhighlight	SET AUTOCOLOR *.rxas rxas
.md / .markdown	SET SDSLH mdp ... --syntaxhighlight	SET AUTOCOLOR *.md mdp
.c / .h	SET SDSLH dslsh-c ...	SET AUTOCOLOR *.c dslsh-c
.py / .js	SET SDSLH pyp / jsp ...	SET AUTOCOLOR *.py pyp, *.js jsp

The New Parser Adapters Share A Shape

C, Python, JavaScript and Markdown use the same editor-facing path.

Tree-sitter adapter shape

```
static const TSAdapterProfile c_profile = {
    "c",
    TS_ADAPTER_C
};

void c_parser(CodeBuffer *codeBuffer) {
    dslsh_tree_sitter_parse(
        codeBuffer,
        tree_sitter_c(),
        &c_profile);
}
```

output

```
DSL-Syntax-Highlighter/parsers/
  c/           -> dslsh-c
  python/      -> pyp
  javascript/  -> jsp
  markdown/    -> mdp

THE sees parser tokens,
not language-specific parser internals.
```

cREXX Macros Drive THE Through ADDRESS

The macro is cREXX source; THE is the command environment.

ADDRESS THE

```
options levelb
import rxfnsb
address the

filename = .string[]
address the "extract /filename/" expose filename[]

note = "state stored by cREXX"
address the "editv put note" expose note

'emsg current file: ' || filename[1]
'file'
```

output

```
ADDRESS THE dispatched normal editor commands
current file: 11-editor-state-demo.the
editv value : state stored by cREXX

The existing THE command parser did the work.
```

Design point

THE does not inject a hidden prelude; hosted cREXX declares its own level and ADDRESS environment.

INPUTSTEM Turns Arrays Into Buffer Lines

This was the clean bridge for generated output panes.

12-inputstem-demo.the

```
output_lines = .string[]
output_lines[1] = "!CGenerated by cREXX!N"
output_lines[2] = "The macro built an array."
output_lines[3] = "THE inserted it with INPUTSTEM."

address the "all"
address the "top"
address the "delete *"
address the "inputstem output_lines." expose output_lines[]
```

output

```
Generated by cREXX
The macro built an array.
THE inserted it with INPUTSTEM.

Same mechanism:
OSREDIR -> capture file
capture file -> .string[] stem
INPUTSTEM -> right pane buffer
```

FILECTLCHAR Brought XEDIT-Style Colour To Output

Useful display markup without taking on protected panel editing yet.

THE display setup

```
SET CTLCHAR ! ESCAPE
SET CTLCHAR R PROTECT #F44747 ON #1E1E1E
SET CTLCHAR G PROTECT #6A9955 ON #1E1E1E
SET CTLCHAR Y PROTECT #DCDCAA ON #1E1E1E
SET CTLCHAR N OFF
SET FILECTLCHAR ON
```

output

```
!Ccyan: presenter heading!N
!Ggreen: command completed!N
!Yyellow: something worth pointing at!N
!Rred: error or risk!N
```

Displayed in the right pane with marker bytes hidden and the file-area background preserved.

REXXSAA Versus CREXXSAA

The bridge is intentionally narrower than the historical SAA variable pool.

Classic REXXSAA	Shape
Run program	RexxStart()
Variables	RexxVariablePool + SHVBLOCK
Command env	Exit handlers / host callbacks
Goal	Historical SAA ABI compatibility

CREXXSAA	Shape
Host lifetime	crexxsaa_context
Run program	run_source() or run_rxbin()
Variables	explicit exposed/sandbox helpers
Goal	small stable facade over cREXX runtime

Key difference

CREXXSAA works with variables the hosted program explicitly exposes during active ADDRESS calls.

Bridge Code: ADDRESS Callback

A cREXX command string lands in THE's existing command parser.

THE native callback

```
static int the_crexx_address_callback(
    const crexxsaa_address_request *request,
    crexxsaa_address_response *response,
    void *userdata)
{
    CHARTYPE *command = strrmdup(request->command);
    short rc = command_line(command, COMMAND_ONLY_FALSE);

    response->rc = rc;
    return 0;
}
```

output

Hosted cREXX writes:

```
address the "extract /filename/" expose filename[]
'msg hello from cREXX'
```

THE executes:

```
command_line("extract /filename/", ...)
command_line("msg hello from cREXX", ...)
```

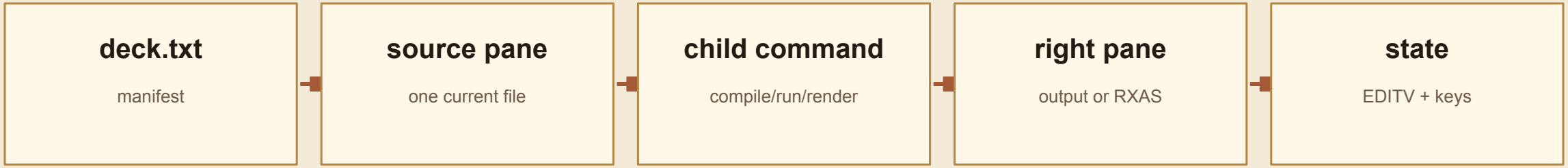
New THE Features Used By The Framework

The live deck exercised real editor features, not a presentation-only mode.

Feature	Purpose
SET SDSLH / SET AUTOCOLOR	Register parser commands and map file patterns
descriptive SET ECOLOR	Readable syntax colour profile
SET FILECTLCHAR	Apply CTLCHAR spans inside file-area output
INPUTSTEM stem.	Insert cREXX stem lines into the current buffer
VALIDTARGET	Expose target validation to macros
EXTRACT /FILECTLCHAR/	Query file-control display state
per-file parser lifecycle	Stable multi-file highlighting

The Presentation Macro Was The Integration Test

Repeated pane switching found bugs that unit-style tests would not naturally cover.



Found

parser state mismatches, colour reset, oversize row crash, RXAS-view lifetime edge.

Fixed

per-file parser handling, file-area control colour, top/delete/INPUTSTEM refresh flow.

Remaining

F5 return signal 11 is now a known follow-up after the symposium.

Where This Can Go Next

THE keeps its Rexx soul, but cREXX is now a modern hosted macro engine.

- Edit, compile, run, and inspect RXAS without leaving THE
- Use cREXX macros for normal editor automation
- Let external parsers handle syntax at editor speed
- Consider XEDIT-style protected panels as a larger design, not a quick flag