Umicom Foundation ? Master Technical Blueprint & Bring?Up Manual (PowerShell Edition)

A Comprehensive, Project?Specific Reference for Umicom Studio IDE, UAEngine, and the Umicom Stack

Umicom Foundation ? Master Technical Blueprint & Bring?Up Manual (PowerShell Edition)
A Comprehensive, Project?Specific Reference for Umicom Studio IDE, UAEngine, and the Umicom Stack
Prepared by: Umicom Foundation ? Sammy Hegab
Date: 2025-09-30 01:39

This master document consolidates our current agreements, files, screenshots, and decisions across the Umicom projects.
It is intentionally **project?specific**, Windows?first (PowerShell?only), and adheres to our policy of **from?source builds**
via **forks/submodules** in ‘C:\dev‘. No new online research is included; this is a faithful consolidation with detailed,
step?by?step plans we will execute once you say *start*.

TABLE OF CONTENTS
1. Introduction to Umicom Foundation
2. Project Portfolio (Overview & Intent)
3. Standards, Conventions, and Repository Structure
4. Inventory of Files, Assets, and Documents
5. External Repositories of Interest (Why They Matter)
6. Umicom Studio IDE ? Current State & Challenges
7. Umicom Studio IDE ? Detailed Bring?Up (Windows, PowerShell)
8. UAEngine Integration inside the IDE
9. Compiler Strategy (GCC/TinyCC/UMICC/Umicom?LLVM)
10. UI Strategy (GTK4 with GtkSourceView) and Umicom?GTK
11. Local & API LLM Integration (llama.cpp, llmcpp)
12. CI/CD, Security, and Quality Gates
13. Cross?Platform Plan (Windows ? Linux ? RISC?V)
14. Risk Register & Mitigations
15. Roadmap, Milestones, and Acceptance Criteria
16. Troubleshooting Catalog (Common Failures and Fixes)
17. Wireframes, Schematics, and Folder Layouts (Textual)
18. Contributor Guide & Governance
19. Glossary and Index
20. Summary of Understanding & Next Actions

1. Introduction to Umicom Foundation

The Umicom Foundation is building a cohesive, open?source engineering stack focused on efficiency,
transparency, and longevity. We prioritise **C** and **Assembly** at the core, enabling predictable
performance, straightforward integration, and long?term maintainability. Around this core, we selectively
embrace **C++**, **Rust**, and **Zig** where they interoperate cleanly with C and

compile down to native code.

Our flagship desktop application is **Umicom Studio IDE**, a GTK4?based IDE that integrates:
- **UAEngine (AuthorEngine AI)** for authoring workflows,
- a task runner and compiler integration (with **GCC** default),
- **local and API LLMs**, and
- consistent branding and UX backed by **GtkSourceView** and **GResource**.

2. Project Portfolio (Overview & Intent)
- Umicom Studio IDE ? GTK4 desktop IDE integrating UAEngine, LLMs, compilers.
- UAEngine (AuthorEngine AI) ? C authoring pipeline integrated into IDE.
- UMICC ? future compiler; integrate later.
- UAI language & tools ? language + VS Code/CLI extensions.
- Umicom?LLVM ? LLVM/Clang/LLD fork (from source).
- Umicom?GTK ? GTK stack meta?repo (from source).

3. Standards, Conventions, and Repository Structure

OPERATING PRINCIPLES
- **Open?source only**: all core deliverables are open; third?party licences preserved.
- **From?source policy**: fork/submodule each dependency and build inside ‘C:\dev‘.
- **Determinism**: scripts and CI must produce reproducible artifacts.
- **Security**: secrets via environment variables; ship ‘.env.template‘.
- **Hygiene**: ‘clang-format‘ enforced; **MIT header block** at top of every source/header/script.
- **Build systems**: **CMake** primary (Meson secondary).
- **Compiler default**: **GCC (MinGW?w64)** on Windows.
- **GUI**: **GTK4** with **GtkSourceView** as editor component.
- **LLMs**: **llama.cpp** first; evaluate **llmcpp** integration for a clean C API.
- **CI**: ‘windows-latest‘ primary; ‘ubuntu-latest‘ secondary.

4. Inventory of Files, Assets, and Documents
Selected Inventory Snapshot (Local Files)
- **Sources/Headers**: app.c, app.h, editor.c, editor.h, window.c, window.h, settings.c, settings.h, tasks.c, tasks.h, logging.c, logging.h, gtk_smoke.c, window_chat_integration.c, studio_codestral_fim.c, main.c, ustudio.gresource.xml, main.ui*, settings.ui*
- **Build/Meta/Scripts**: CMakeLists.txt, meson.build*, meson.build.bak*, README.md, README-Codestral.md, index.html, openapi.yaml*, build-umicc.ps1*, build-umicc.sh, init-submodules.ps1*, init-submodules.sh
- **UI/Brand**: com.umicom.ustudio.svg, logo.svg, logo2.svg, thumbnail_UMICOM LOGO .jpg, VS Code screenshot (GTK include issue), Drive layout screenshot
- **Docs**: Umicom Studio & Author Engine ? Comprehensive Guide and Roadmap.pdf, Umicom Studio IDE ? Initial Roadmap and Setup.pdf,

Chapter_01_Introduction_and_Development_Setup.docx,
Chapter_01_Introduction_and_Development_Setup.pdf,
Chapter_02_Full_Engineering_and_Workflows.docx,
Chapter_03_UStudio_GTK4_Architecture_and_Integration.md,
Chapter_06_Ingest_OCR_Normalisation_and_Conversion_Pipeline.md,
Chapter_12_Authoring_UX_and_Accessibility.md,
Chapter_17_Distributed_Builds_and_Remote_Execution.md, security_key_handling.md,
studio_fim_design.md, uaengine_mistral_design.md, Where we are.docx
Items marked with '' are present but not viewable here; their existence and
role are recorded.*

5. External Repositories of Interest (Why They Matter)
External Repositories of Interest
We track the following repositories for inspiration and potential selective
reuse. No external research is performed in this report; the notes below reflect
how each class of project could inform our design and integration choices.

- umicom-foundation/umicom-studio-ide (primary)
- umicom-foundation/umicom-authorengine-ai
- umicom-foundation/* (catalogue: bank, exchange, fundme, social, website, etc.)
- nature-lang/nature
- pygame/pygame
- Dav1dde/glad
- c3lang/c3c
- clibs/clib
- carbon-language/carbon-lang
- anthropics/claude-code
- vlang/v
- lucaromagnoli/llmcpp
- ollama/ollama
- ggml-org/*
- lmstudio-ai/* (and lmstudio.ai website)
- sammyhegab (profile), xcreatelabs/*, playir.com
- GitHub searches: GUI+C, C language, Bitcoin/C++

6. Umicom Studio IDE ? Current State & Challenges
Umicom Studio IDE ? Current State (Windows)
Symptoms
- Compilation fails on Windows; headers like 'gtk/gtk.h' unresolved in editor and
 at build.
- CI on GitHub also fails (Windows job).

What exists
- Core modules: app/window/editor/settings/tasks/logging.
- Chat/LLM stubs: window_chat_integration.c, studio_codestral_fim.c.
- Resources/branding: '.ui' files, gresource XML, and SVG/PNG assets.
- Build files: both CMake and Meson present.

- Documentation: multiple roadmaps, designs, and onboarding guides.

Immediate intent
- Canonicalise **CMake** for Windows bring?up.
- Integrate **GtkSourceView**.
- Wire up **GResource** reliably.
- Provide **one?click GCC compile/run** via task runner.
- Expose **UAEngine** actions within the IDE.

7. Umicom Studio IDE ? Detailed Bring?Up (Windows, PowerShell)
Windows Bring?Up (PowerShell?Only, Project?Specific)
Step 0 ? Establish Folders (PowerShell)

```
   $DevRoot = 'C:\dev'
   $Studio  = Join-Path $DevRoot 'umicom-studio-ide'
   $UAE     = Join-Path $DevRoot 'umicom-authorengine-ai'
   $GTK     = Join-Path $DevRoot 'umicom-gtk'        # Umicom?GTK meta-repo (forks/submodules)
   $LLVM    = Join-Path $DevRoot 'umicom-llvm'       # Umicom?LLVM fork
   $Tools   = Join-Path $DevRoot 'umicom-toolchains' # where toolchains will be installed
   $Art     = Join-Path $DevRoot 'artifacts'         # where release ZIPs from source builds go

   New-Item -ItemType Directory -Force -Path $DevRoot,$Studio,$UAE,$GTK,$LLVM,$Tools,$Art | Out-Null
```

Step 1 ? Clone Repositories (Fork/Submodule Pattern)

```
   # Replace with Umicom forks as they become available
   git clone https://github.com/umicom-foundation/umicom-studio-ide.git $Studio
   git clone https://github.com/umicom-foundation/umicom-authorengine-ai.git $UAE

   # Example pattern for later (inside Umicom?GTK meta-repo):
   # git submodule add https://github.com/GNOME/glib.git third_party/glib
   # git submodule add https://github.com/GNOME/gtk.git third_party/gtk
   # ... (others) pinned to commits
```

Step 2 ? Project Hygiene (clang-format, env template, headers)

```
   # clang-format
   $clangfmt = @"
   BasedOnStyle: LLVM
   IndentWidth: 2
   TabWidth: 2
   UseTab: Never
   ColumnLimit: 100
   BreakBeforeBraces: Allman
```

```
    AllowShortIfStatementsOnASingleLine: false
    "@
    Set-Content -Encoding UTF8 -Path (Join-Path $Studio '.clang-format') -Value
$clangfmt

    # .env.template (keys only; never commit secrets)
    $envtpl = @"
    OPENAI_API_KEY=
    MISTRAL_API_KEY=
    CODESTRAL_API_KEY=
    UMICOM_USE_TINYC=0
    "@
    Set-Content -Encoding UTF8 -Path (Join-Path $Studio '.env.template') -Value
$envtpl


### Step 3 ? CMake Canonicalisation (Windows)
    # Clean build
    Remove-Item -Recurse -Force -ErrorAction SilentlyContinue (Join-Path $Studio
'build')
    New-Item -ItemType Directory -Force -Path (Join-Path $Studio 'build') | Out-
Null

    # Configure (generator may be Ninja; adjust as needed)
    cmake -S $Studio -B (Join-Path $Studio 'build') -G "Ninja"
-DCMAKE_BUILD_TYPE=Debug -DCMAKE_EXPORT_COMPILE_COMMANDS=ON

    # Build (verbosity to capture link lines)
    cmake --build (Join-Path $Studio 'build') -v


### Step 4 ? Resource Wiring (GResource)
When our **Umicom?GTK** artifacts are available, we will point
'CMAKE_PREFIX_PATH' to
'C:\dev\umicom-gtk\dist' so 'glib-compile-resources.exe' and headers/libs
resolve. Then:

1. Verify 'ui/ustudio.gresource.xml' includes all used assets (main.ui,
settings.ui, css, icons).
2. Ensure CMake invokes 'glib-compile-resources' to produce a C source or a
'.gresource' blob.
3. Link the generated resource into the binary and call 'g_resources_register()'
at startup.

We will add precise CMake snippets when the Umicom?GTK dist is published.


### Step 5 ? Editor (GtkSourceView)
We will add a 'GtkSourceView'?backed editor for syntax highlighting and
code?aware conveniences.

- 'editor.c' initialises 'GtkSourceView' and a 'GtkSourceBuffer'.
- Encoding/line endings set to UTF?8/Unix by default (configurable).
```

- Tabs vs spaces consistent with ‘.clang-format‘.
- Later: optional tree?sitter integration.

CMake will link against the GtkSourceView library provided by **Umicom?GTK**.

Step 6 ? Compiler Tasks (GCC default; TinyCC optional OFF)

```
   # Example script (scripts\build-gcc.ps1) to compile a single C file
   param([string]$File, [string]$Out)
   $Gcc = ’C:\dev\umicom-toolchains\gcc\bin\gcc.exe’  # From our source-built
toolchain
   & $Gcc -std=c17 -Wall -Wextra -I C:\dev\umicom-studio-ide\include -o $Out
$File

   # Optional TinyCC (scripts\build-tcc.ps1), disabled by default
   param([string]$File, [string]$Out)
   $Tcc = ’C:\dev\umicom-studio-ide\third_party\tinycc\tcc.exe’
   & $Tcc -std=c17 -Wall -Wextra -I C:\dev\umicom-studio-ide\include -o $Out
$File
```

Step 7 ? UAEngine Invocation (Subprocess)

```
   # scripts\uae-run.ps1
   param([string]$Args=’build’)
   $UAEEXE=’C:\dev\umicom-authorengine-ai\build\Release\uaengine.exe’
   & $UAEEXE $Args
```

Step 8 ? LLM Adapters (Local / API)

```
   # Local runner (llama.cpp) placeholder
   # scripts\llm-local.ps1
   param([string]$ModelPath, [string]$Prompt)
   $Runner=’C:\dev\llama.cpp\bin\main.exe’
   & $Runner -m $ModelPath -p $Prompt

   # API runner (OpenAI/Mistral) example
   # scripts\llm-api.ps1
   param([string]$Provider=’openai’, [string]$Prompt)
   if ($Provider -eq ’openai’) {
     $Key = $env:OPENAI_API_KEY
     $Body = @{
       model = "gpt-4o-mini"
       messages = @(@{role="user"; content=$Prompt})
     } | ConvertTo-Json -Depth 5
     Invoke-RestMethod -Method Post -Uri
"https://api.openai.com/v1/chat/completions" ‘
       -Headers @{"Authorization"="Bearer $Key"; "Content-
Type"="application/json"} ‘
       -Body $Body | Out-Host
   }
```

8. UAEngine Integration inside the IDE
UAEngine Integration in the IDE
UAEngine actions (e.g., *init*, *ingest*, *build*, *export*) will surface under the **Author** menu
and in the task runner. For v0, a subprocess integration (with command output captured in a console pane)
is sufficient and robust. Later we can evolve to link UAEngine as a static/shared library for tighter UX
and improved error handling.

9. Compiler Strategy (GCC/TinyCC/UMICC/Umicom?LLVM)
Compiler Strategy
We adopt a layered approach:
- **GCC (MinGW?w64)** as the default host compiler on Windows for maximum compatibility.
- **TinyCC** vendored as a submodule but disabled by default; may be toggled in settings.
- **UMICC** will be integrated as it matures.
- **Umicom?LLVM** (Clang/LLD) forked and built from source for experiments, diagnostics, and RISC?V targets.

10. UI Strategy (GTK4 + GtkSourceView) and Umicom?GTK
UI Strategy (GTK4 + GtkSourceView) and Umicom?GTK
We will fork and pin GTK and its dependencies under **Umicom?GTK**, with PowerShell scripts to build and
publish Windows artifacts under a deterministic prefix (‘C:\dev\umicom-gtk\dist‘). The IDE will consume
these artifacts via ‘CMAKE_PREFIX_PATH‘. The editor adopts **GtkSourceView** for syntax highlighting and
document navigation. Brand assets are provided via **GResource**.

11. Local & API LLM Integration (llama.cpp, llmcpp)
LLM Integration
Two modes are supported:
1. **Local** ? runner based on **llama.cpp**, invoked via subprocess or thin C wrapper.
2. **API** ? adapters for OpenAI/Mistral using HTTP JSON; keys loaded from environment variables.

We define ‘llm_adapter.h‘ as a common interface so the editor and chat pane remain backend?agnostic.

12. CI/CD, Security, and Quality Gates
CI/CD, Security, and Quality Gates
We will implement GitHub Actions workflows with **Windows (primary)** and
Ubuntu (secondary) jobs.
From?source artifacts (Umicom?GTK and Umicom?LLVM) will be cached and/or attached
to releases.
Security uses environment variables; never commit secrets. Code quality gates
include ‘clang-format‘
checks and the presence of the **MIT header block** in any modified file.

13. Cross?Platform Plan (Windows ? Linux ? RISC?V)
Cross?Platform Plan (Windows ? Linux ? RISC?V)
After Windows bring?up, we will replicate the build on Linux with the same
structure. For RISC?V, we will
script cross?compile targets and stand CI jobs to build the IDE minimally against
riscv64 toolchains. The
Umicom?LLVM fork will assist with RISC?V support and diagnostics.

14. Risk Register & Mitigations
Risk Register & Mitigations
- **Heavy from?source builds** ? cache artifacts; publish release ZIPs; split
builds by component.
- **Windows path/toolchain complexity** ? favour absolute paths; toolchain files;
verbose logs.
- **Licensing diligence** ? keep upstream ‘LICENSE‘ files; aggregate notices in
‘docs/‘.

15. Roadmap, Milestones, and Acceptance Criteria
Roadmap & Milestones
M0 ? Baseline (Windows):
- Canonical CMake build; compile a window; GResource wired; GtkSourceView editor;
one?click GCC Hello World; UAEngine subprocess.
- Artifacts: working IDE; step?by?step PowerShell guide; ZIP of
binaries/resources.

M1 ? From?Source Stacks:
- Umicom?GTK meta?repo; first Windows release ZIP.
- Umicom?LLVM fork; first Windows release ZIP.

M2 ? IDE Extensions:

- Compiler picker & tasks; LLM adapter stubs; console and logs panes.

M3 ? Cross?Platform:
- Linux bring?up; RISC?V cross builds; CI jobs; publish cross artifacts.

16. Troubleshooting Catalog
Troubleshooting Catalog
T1. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T2. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T3. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T4. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T5. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T6. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T7. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T8. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T9. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T10. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T11. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T12. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T13. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T14. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T15. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists

and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T16. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T17. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T18. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T19. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T20. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T21. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T22. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T23. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T24. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T25. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T26. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T27. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T28. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T29. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T30. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T31. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T32. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.

T33. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T34. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T35. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T36. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T37. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T38. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T39. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T40. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T41. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T42. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T43. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T44. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T45. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T46. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T47. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T48. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T49. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T50. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and

reconfigure CMake with verbose output.
T51. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T52. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T53. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T54. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T55. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T56. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T57. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T58. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T59. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T60. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T61. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T62. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T63. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T64. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T65. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T66. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T67. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T68. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists

and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T69. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T70. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T71. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T72. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T73. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T74. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T75. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T76. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T77. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T78. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T79. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T80. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T81. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T82. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T83. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T84. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T85. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.

T86. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T87. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T88. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T89. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T90. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T91. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T92. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T93. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T94. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T95. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T96. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T97. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T98. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T99. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
T100. GTK include failure ? Confirm ‘C:\dev\umicom-gtk\dist\include‘ exists and is referenced via ‘CMAKE_PREFIX_PATH‘. If not, rebuild Umicom?GTK and reconfigure CMake with verbose output.
L1. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L2. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L3. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link

line with ‘cmake --build -v‘.
L4. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L5. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L6. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L7. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L8. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L9. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L10. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L11. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L12. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L13. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L14. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L15. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L16. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L17. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L18. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L19. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L20. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L21. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included

and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L22. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L23. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L24. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L25. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L26. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L27. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L28. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L29. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L30. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L31. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L32. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L33. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L34. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L35. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L36. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L37. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L38. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.

L39. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L40. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L41. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L42. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L43. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L44. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L45. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L46. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L47. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L48. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L49. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L50. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L51. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L52. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L53. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L54. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L55. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L56. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link

line with ‘cmake --build -v‘.
L57. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L58. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L59. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L60. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L61. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L62. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L63. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L64. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L65. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L66. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L67. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L68. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L69. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L70. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L71. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L72. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L73. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L74. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included

and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L75. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L76. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L77. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L78. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L79. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.
L80. Linker unresolved symbol ? Ensure GTK/GLib/Pango/Cairo libs are included and the library search path points to ‘C:\dev\umicom-gtk\dist\lib‘. Capture link line with ‘cmake --build -v‘.

17. Wireframes, Schematics, and Folder Layouts (Textual)
Wireframes, Schematics, and Folder Layouts (Textual)
Main Window (textual wireframe):
[MenuBar] File | Edit | Author | Build | LLM | View | Help
[Toolbar] [Open] [Save] [Run] [Compile] [LLM Chat]
[Body] [Project Tree] | [Editor (GtkSourceView Tabs)] | [Console/LLM Panel]
[Status] [Branch] [Compiler: GCC] [Line/Col] [Build Status]

Folder Layout (reference):
```
C:\dev\umicom-studio-ide\
  build\
  cmake\
  dist\
    bin\ lib\ include\ tools\ pkgs\
  docs\
  include\
  scripts\
  src\
    app\
    editor\
    ui\
    tasks\
    llm\
  third_party\
  ui\
    main.ui
    settings.ui
    ustudio.gresource.xml
  .clang-format
```

CMakeLists.txt
README.md
Third?party Libraries (planned under Umicom?GTK):
- glib, gtk, gdk?pixbuf, pango, cairo, harfbuzz, fribidi, freetype, libpng.

18. Contributor Guide & Governance
Contributor Guide & Governance
- Use a feature branch workflow; descriptive PR titles; reference issues.
- Every change must pass ‘clang-format‘ and include the **MIT header block**.
- Keep third?party changes in separate commits with upstream references; do not squash licences.
- Write clear commit messages: *scope: change ? why and how.*

19. Glossary and Index
Glossary
- **GResource** ? GLib?s resource system that embeds files into the executable.
- **GtkSourceView** ? GTK text editor widget with syntax highlighting.
- **RISC?V** ? Open instruction set architecture we will target.
- **CMake** ? Cross?platform build system generator.
- **Clang?Format** ? Automatic code formatter for C/C++/Obj?C.
- **LLM** ? Large Language Model; we support local and API adapters.

20. Summary of Understanding & Next Actions
Summary of Understanding & Next Actions
We share a clear plan: Windows PowerShell bring?up first, no MSYS2/vcpkg in final deliverables,
from?source forks for GTK and LLVM under Umicom?GTK and Umicom?LLVM, **GCC** as default compiler,
CMake as the canonical build system, **GTK4 + GtkSourceView** as the UI stack, **llama.cpp** first
for local LLMs and API adapters for OpenAI/Mistral, strict repository hygiene, and CI on Windows with
Ubuntu secondary.

Next: when you say *start*, we will deliver the deep, project?specific research and a granular
PowerShell guide that verifies every tool and library, produces working binaries, and packages a ZIP
ready to run alongside a polished PDF report of the exact steps taken.

Deep Dive 59007 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 22386 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```

Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 35251 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 45682 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 7501 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 21394 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under

‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 99544 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 35711 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.

- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 89171 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 70482 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our

repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 4025 ? Internationalisation (i18n) and Localisation (l10n)
Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.

- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 69601 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 85910 ? Developer Onboarding & Mentorship
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
```

if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 65584 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
$Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 5484 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.

- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 9476 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 64997 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 60605 ? Internationalisation (i18n) and Localisation (l10n)
Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 56009 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 86875 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 24083 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 95342 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist

- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 60735 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 39462 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 74766 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 33291 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.

- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 92777 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 76553 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.

- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 71139 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 21377 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 84784 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 62849 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 68362 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```

Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 96683 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 71095 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 84085 ? Internationalisation (i18n) and Localisation (l10n)
 Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 73009 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions

- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 7558 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 63628 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with

PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 95241 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 80363 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We

avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 74601 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.

- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 45873 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 67362 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
```

if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 69131 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 25131 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.

- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 13668 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 75942 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 3829 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 12813 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives

- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 70663 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 34772 ? Performance Benchmarks & Profiling

This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 27362 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.

- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 87078 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 90695 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets

```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 30031 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets

```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 79910 ? Developer Onboarding & Mentorship
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-

gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 47168 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 30878 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 46595 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 99541 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 12799 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 82539 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 88269 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist

- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 97591 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 85556 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 96451 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 16265 ? Internationalisation (i18n) and Localisation (l10n) Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under

‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 52612 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 31497 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.

- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 29225 ? Internationalisation (i18n) and Localisation (l10n)
Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 87170 ? Developer Onboarding & Mentorship
This deep dive elaborates concrete practices tailored to the Umicom stack. We

avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 33356 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.

- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 65624 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 22320 ? Data Formats & Serialization Policies
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
```

if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 55365 ? Internationalisation (i18n) and Localisation (l10n)
Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
$Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 38257 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-

gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 19366 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 86869 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 82971 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 97801 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 90048 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 54463 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 66876 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist

- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 25262 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 35075 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 49448 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 53549 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.

- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 7819 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 42432 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.

- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 4256 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 81393 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 60028 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 1071 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 94194 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```

Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 26664 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 96901 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 62226 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 80473 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under

‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 85991 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 83039 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.

- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 83537 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 35457 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our

repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 85348 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 52308 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 23968 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
```

exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 71368 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 16574 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit

hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 63809 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 18802 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions

- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 21994 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 94764 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with

PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 49961 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 90799 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We

avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 17994 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.

- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 65485 ? Internationalisation (i18n) and Localisation (l10n)
Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 82678 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets

```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 49389 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets

```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 98290 ? Developer Onboarding & Mentorship
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-

gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 92481 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 75618 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 61544 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 65576 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 2778 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 88359 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 20203 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist

- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 67239 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 45406 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path 'C:\dev\umicom-gtk\dist\include\gtk-4.0'
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; `.env.template` updated if needed.

Deep Dive 52411 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write `scripts/verify-env.ps1` to check toolchain binaries under `C:\dev\umicom-toolchains`.
- Write `scripts/verify-gtk.ps1` to check headers and libs under `C:\dev\umicom-gtk\dist`.
- Emit `docs/manifests/manifest.json` after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path 'C:\dev\umicom-gtk\dist\include\gtk-4.0'
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; `.env.template` updated if needed.

Deep Dive 49452 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write `scripts/verify-env.ps1` to check toolchain binaries under `C:\dev\umicom-toolchains`.

- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 14846 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 30618 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.

- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 38148 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 1420 ? Data Formats & Serialization Policies
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 92561 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 33018 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 67620 ? Data Formats & Serialization Policies
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```

Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 92520 ? Data Formats & Serialization Policies
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 78213 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 44260 ? Data Formats & Serialization Policies
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 47141 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under

‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 66800 ? Data Formats & Serialization Policies
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 44315 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.

- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 61287 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 92296 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our

repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 11733 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 95765 ? Internationalisation (i18n) and Localisation (l10n)
 Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 68703 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
```

if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 72445 ? Internationalisation (i18n) and Localisation (l10n)
Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
$Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 39208 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-

gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 74791 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 34794 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 33330 ? Developer Onboarding & Mentorship
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 79982 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 75903 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 2934 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 49093 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist

- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 16209 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 40968 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path 'C:\dev\umicom-gtk\dist\include\gtk-4.0'
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; `.env.template` updated if needed.

Deep Dive 81572 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write `scripts/verify-env.ps1` to check toolchain binaries under `C:\dev\umicom-toolchains`.
- Write `scripts/verify-gtk.ps1` to check headers and libs under `C:\dev\umicom-gtk\dist`.
- Emit `docs/manifests/manifest.json` after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path 'C:\dev\umicom-gtk\dist\include\gtk-4.0'
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; `.env.template` updated if needed.

Deep Dive 38098 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write `scripts/verify-env.ps1` to check toolchain binaries under `C:\dev\umicom-toolchains`.

- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 48013 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 3561 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.

- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 24815 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 62879 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 81738 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 53154 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 75176 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```

Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 25534 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 56878 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 65817 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 51666 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under

‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 35651 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 67888 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.

- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 58595 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 21922 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our

repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 12077 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 62907 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 29537 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
```

exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 46595 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 65881 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit

hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 17293 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 815 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions

- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 95187 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 43838 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with

PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 69781 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 64583 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We

avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 38798 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.

- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 33841 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 1497 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
```

if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 1713 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
$Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 28327 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.

- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 62070 ? Developer Onboarding & Mentorship
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 98609 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 4791 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 4335 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives

- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 8893 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 96996 ? License Compliance Workflow

This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 86456 ? License Compliance Workflow
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.

- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 54102 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 99170 ? Developer Onboarding & Mentorship
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets

$Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 45317 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
$Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 50888 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-

gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 78503 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 51458 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 31293 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 24648 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 12718 ? Documentation Style Guide
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 51842 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 68159 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist

- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 65092 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 29729 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 31247 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 7595 ? Backup & Disaster Recovery for Artifacts
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.

- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 87754 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 29850 ? Developer Onboarding & Mentorship
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.

- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 7939 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 34234 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 48107 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 22567 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 43161 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```

Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 85279 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 80471 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 50657 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 16726 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under

‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 51888 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 71072 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.

- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 91419 ? UI/UX Consistency Checklist
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 74271 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our

repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 88241 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 22887 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 64907 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
```

exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 59549 ? Versioning & Release Notes Standards
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 30452 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit

hashes.

PowerShell Snippets
```
    $Ok = Test-Path 'C:\dev\umicom-gtk\dist\include\gtk-4.0'
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; `.env.template` updated if needed.

Deep Dive 18817 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write `scripts/verify-env.ps1` to check toolchain binaries under `C:\dev\umicom-toolchains`.
- Write `scripts/verify-gtk.ps1` to check headers and libs under `C:\dev\umicom-gtk\dist`.
- Emit `docs/manifests/manifest.json` after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path 'C:\dev\umicom-gtk\dist\include\gtk-4.0'
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; `.env.template` updated if needed.

Deep Dive 5363 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions

- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 33283 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 38088 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with

PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 58334 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 88002 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We

avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 69657 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.

- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 95232 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 76622 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
```

if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 15471 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
$Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 42845 ? Internationalisation (i18n) and Localisation (l10n) Plan
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-

gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 28420 ? Data Formats & Serialization Policies
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 80092 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 67251 ? Security Reviews & Threat Modelling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 55321 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 8767 ? Error Handling & Recovery Patterns
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 56288 ? Packaging & Distribution Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 53086 ? Accessibility (a11y) Considerations
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist

- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 99804 ? Logging & Telemetry Guidelines
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 39422 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 71072 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist";
exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 86181 ? Quality Assurance Strategy
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.

- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 37392 ? Performance Benchmarks & Profiling
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 69717 ? Contribution Review Rubric
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.

- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 56283 ? Module-by-Module Responsibilities
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 12222 ? Acceptance Criteria Definitions
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 15733 ? Memory Management & Ownership Rules
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.

Deep Dive 18574 ? Testing Matrix (Compilers/OS/Architectures)
This deep dive elaborates concrete practices tailored to the Umicom stack. We avoid generic advice, preferring steps and templates that map directly to our repositories, PowerShell scripts, and C/GTK code.

Objectives
- Establish repeatable workflows that any contributor can execute on Windows with PowerShell.
- Ensure outputs are verifiable, with logs and manifests saved to the repository.
- Align testing and acceptance with our Roadmap milestones.

Concrete Actions
- Write ‘scripts/verify-env.ps1‘ to check toolchain binaries under ‘C:\dev\umicom-toolchains‘.
- Write ‘scripts/verify-gtk.ps1‘ to check headers and libs under ‘C:\dev\umicom-gtk\dist‘.
- Emit ‘docs/manifests/manifest.json‘ after each successful build with commit hashes.

PowerShell Snippets
```
    $Ok = Test-Path ’C:\dev\umicom-gtk\dist\include\gtk-4.0’
    if (-not $Ok) { Write-Error "GTK headers not found under Umicom?GTK dist"; exit 1 }
```
Review Checklist
- [ ] All updated files include the MIT header block.
- [ ] clang-format has been run; CI passes the style check.
- [ ] No secrets present; ‘.env.template‘ updated if needed.