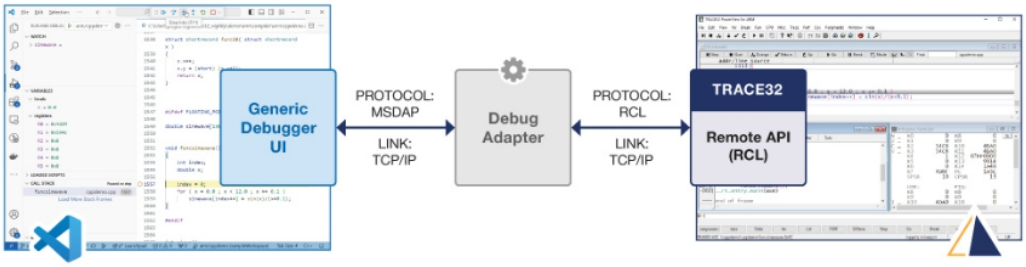
Debug Adapater Protocol (DAP) Support by TRACE32®

2026-11-19 - Comments (0) - Setup / update

Debug Adapter Protocol (DAP) Support by TRACE32®

The Debug Adapter Protocol (DAP) is a popular interface to connect integrated development environments (IDEs) to debuggers. TRACE32® supports DAP by translating between DAP messages and TRACE32®'s Remote API (RCL).



The debug adapter supports:

- Run-state control: resume and halt the processor's execution, single-step and set breakpoints (including conditional breakpoints and watchpoints).
- Source-level debugging: access local variables and inspect the stack frame.
- Register and memory access: TRACE32® access classes are supported, e.g. `CSR:` for RISC-V control and status registers.
- Disassembly view: inspect bare-metal assembly.
- Source path translations, e.g. for ELFs compiled on a build server.

Requirements

- TRACE32® PowerView 2026.09 or newer. The DAP server `t32debugadapter` is included in TRACE32® installations.
- To attach to a running PowerView instance, the Remote API must be enabled.

Starting the DAP Server

The DAP server is usually started by the IDE integration, e.g. the TRACE32® Debug extension for Visual Studio Code. It can also be started manually:

```
t32debugadapter --port=58870
```

Options:

- `--port`: Port number the DAP server listens on (default: 58870).
- `--log_to`: Logging target: none, stdout, file, or file:/path/to/log.txt (default: stdout).
- `--log_level`: Log level: info or debug (default: info).
- `--version`: Print version number and exit.

Configuration

The debug configuration is passed through the arguments of the `launch` and `attach` requests.

Launch Request Arguments

- `trace32Port`: TCP port of the TRACE32 RCL API (required, default: `20000`).
- `trace32SystemPath`: Path to the TRACE32 system directory (default: `trace32.systemPath` setting).
- `trace32Target`: TRACE32 executable for the target architecture, e.g. `t32marm` or `t32mriscv` (required).
- `trace32ConnectionScript`: Connection script executed by PowerView at startup.
- `trace32SetupScript`: PRACTICE script executed after PowerView has started.
- `trace32SetupScriptArgs`: Arguments passed to `trace32SetupScript`.
- `trace32SetupScriptTimeout`: Timeout in seconds for `trace32SetupScript`.
- `trace32SourceTranslations`: See Source Path Translations.
- `screen`: Open the PowerView GUI (default: `true`).
- `waittrace32`: Time in seconds to wait for PowerView to start (default: `1`).

Attach Request Arguments

- `trace32Port`: TCP port of the TRACE32 RCL API (required, default: `20000`).
- `trace32Node`: Host where the PowerView instance is running (default: `localhost`).
- `trace32SetupScript`: PRACTICE script executed after attaching.
- `trace32SetupScriptArgs`: Arguments passed to `trace32SetupScript`.
- `trace32SetupScriptTimeout`: Timeout in seconds for `trace32SetupScript`.

- `trace32SourceTranslations`: See Source Path Translations.

Deprecated Arguments:

- `cmm`: use `trace32SetupScript` instead
- `translateInputPath` and `translateOutputPath`: use `trace32SourceTranslations` instead

Enable the Remote API

- Configuration file `config.t32`:

```
RCL=NETTCP
PORT=20000
```

- Command line option:

```
t32marm.exe --t32-api-rcl=TCP:20000 -c config.t32
```

- PowerView command:

```
SETUP.API.RCL /Interface TCP /Port 20000.
```

If the IDE connects from another machine, the Remote API must listen on the network:

```
SETUP.API.RCL /Interface TCP /Port 20000. /RemoteHost ALLOW
```

Connection Scripts

A connection script is a PRACTICE script that connects PowerView to a debug module, simulator or other interface. With `trace32ConnectionScript`, the launched PowerView instance runs the script at startup instead of prompting for a connection.

To create a connection script:

1. Start a PowerView instance (requires release 09/2024 or newer).
2. Open the connection configuration: menu File > Connection State, or command `CONNECTION.state`.
3. Click “Create connection script” (command `CONNECTION.Wizard.CreateScript`).

Alternatively, the command `CONNECTION.Wizard.autostart` creates a connection script and selects it as autostart script.

For more information, see the PowerView User’s Guide, chapter “Connection Script Generation for Test Automation”, and the `CONNECTION` command group in the PowerView Command Reference.

Source Path Translations

If the ELF file was compiled under a different path than where the sources live in your IDE (e.g. on a build server), the debug adapter cannot find the sources. Use `trace32SourceTranslations` to map TRACE32 source paths to the paths of your IDE. Each entry is a `[trace32Path, clientPath]` pair, e.g. for an ELF compiled in `\work`:

```
"trace32SourceTranslations": [["\\work", "/home/you/src"]]
```