Debugging Setup

Relevant source files

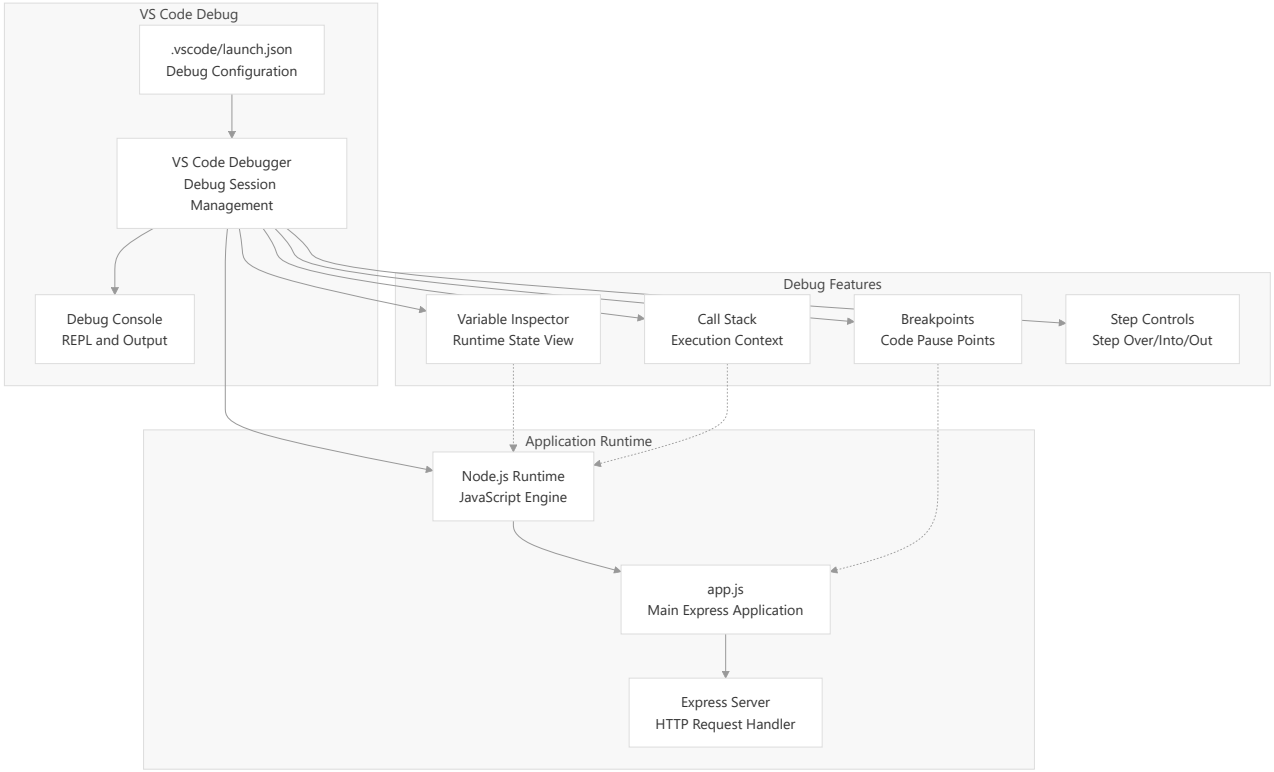
This document details the Node.js debugging configuration for PadelFlow, explaining how to set up and use the VS Code debugging environment to troubleshoot and develop the application. This covers the launch configuration, debugging workflow, and integration with the main Express application.

For information about the broader VS Code workspace configuration, see IDE Configuration. For details about the main application structure being debugged, see Server Setup and Configuration.

VS Code Debug Configuration Overview

PadelFlow includes a preconfigured debugging setup through VS Code's launch configuration system. The debugging environment is designed to launch and attach to the main Express application, providing full debugging capabilities including breakpoints, variable inspection, and step-through debugging.

Launch Configuration Structure



Sources: .vscode/launch.json 1-17

Debug Configuration Details

The launch configuration is defined as a standard Node.js debugging setup:

Configuration Property	Value	Purpose
`type`	`"node"`	Specifies Node.js debugging
`request`	`"launch"`	Launches new process instead of attaching

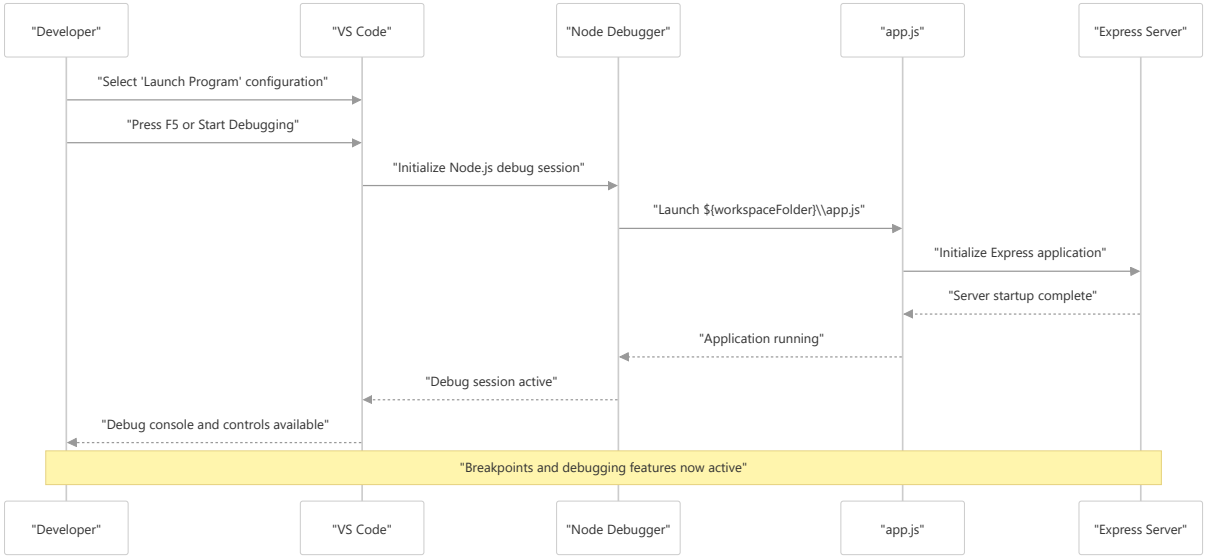
Configuration Property	Value	Purpose
`name`	`"Launch Program"`	Display name in VS Code debug menu
`skipFiles`	`["<node_internals>/**"]`	Excludes Node.js internal files from debugging
`program`	`"${workspaceFolder}\\app.js"`	Entry point for the application

The configuration targets the main application entry point at `app.js` in the workspace root, which serves as the Express server initialization file.

Sources: .vscode/launch.json 7-15

Debugging Workflow

Starting a Debug Session



The debugging process begins by launching the main `app.js` file through the Node.js runtime, establishing a debug connection that allows inspection of the entire Express application lifecycle.

Sources: .vscode/launch.json | 14

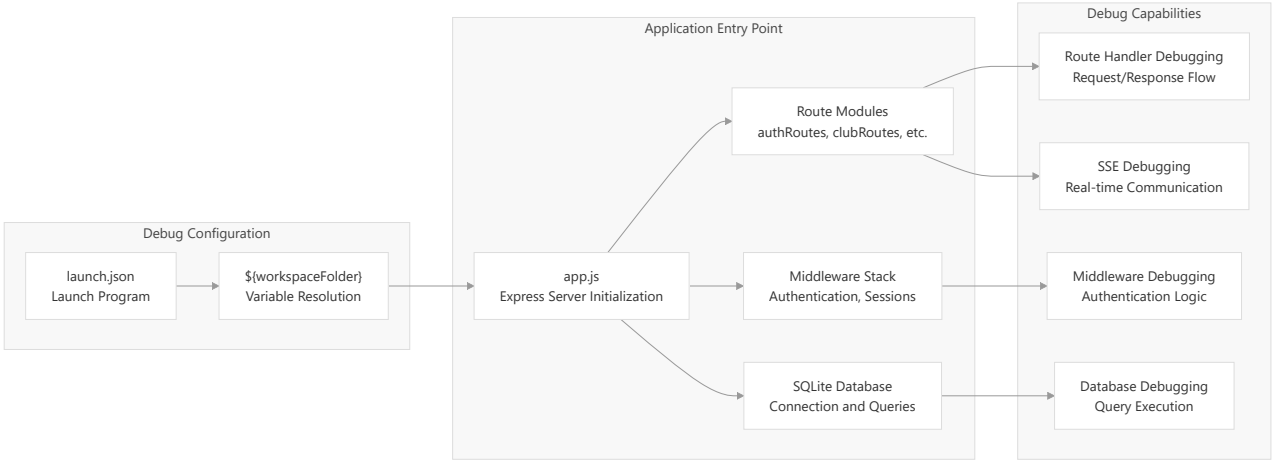
Debug Session Components

The debugging environment provides several integrated components for development:

- **Breakpoint Management**: Set breakpoints directly in source files to pause execution
- **Variable Inspection**: Examine runtime values, object properties, and application state
- **Call Stack Navigation**: View the execution path and navigate between function calls
- **Debug Console**: Interactive REPL for evaluating expressions and running commands
- **Step Controls**: Control execution flow with step over, step into, and step out operations

Integration with Application Architecture

Debug Target Mapping



The debug configuration enables comprehensive debugging across all application layers, from HTTP request handling through database operations and real-time communications.

Sources: .vscode/launch.json | 14

Debug Environment Features

Node.js Internal Filtering

The configuration includes `skipFiles` settings to exclude Node.js internal modules from the debugging session. This focuses the debugging experience on application code rather than Node.js runtime internals:

```
"skipFiles": [
    "<node_internals>/**"
]
```

This filtering improves debugging performance and reduces noise during step-through debugging by automatically stepping over Node.js core module code.

Workspace Integration

The debug configuration uses VS Code's workspace variable system with `${workspaceFolder}\\app.js` to ensure the correct application entry point is launched regardless of the workspace location. This provides portability across different development environments and file system structures.

Sources: .vscode/launch.json 11-14

Usage Instructions

Setting Up Debugging

1. **Open VS Code**: Ensure the PadelFlow workspace is loaded
2. **Access Debug View**: Use `Ctrl+Shift+D` or click the Debug icon in the Activity Bar
3. **Select Configuration**: Choose "Launch Program" from the configuration dropdown
4. **Set Breakpoints**: Click in the gutter next to line numbers to set breakpoints
5. **Start Debugging**: Press `F5` or click the green play button

Common Debugging Scenarios

Debugging Task	Recommended Approach
Route Handler Issues	Set breakpoints in route files, inspect `req` and `res` objects
Authentication Problems	Debug middleware stack, examine session data

Debugging Task	Recommended Approach
Database Query Issues	Step through database operations, inspect query parameters
Real-time Feature Problems	Debug SSE handlers, monitor event emission
Server Startup Issues	Place breakpoints in `app.js` initialization code

The debugging setup integrates seamlessly with the broader PadelFlow development environment, providing comprehensive visibility into the Express application's runtime behavior.

Sources: .vscode/launch.json 1-17