> Note: Document will be moved to [nodejs/tracing-wg](https://github.com/nodejs/tracing-wg/blob/master/wg-meetings/) after the meeting.

# Diag WG Meeting - April 2017

* Date: 2017-05-11
* Issue: <https://github.com/nodejs/diagnostics/issues/94>
* Recording: https://www.youtube.com/watch?v=YTlL1MsLbZw
* Minutes: https://docs.google.com/document/d/1pbsM1nhwcZcDpi9Dff7ngkoYgPTyBQVPfD0TPbACxpc
* Previous meeting: https://docs.google.com/document/d/1lhy1H37hsbjKljY0XYr5tt2nO5Xbo54Vf_PjBuIjvOA

## Attendees

* Thomas Watson @watson
* Michael Dawson @mhdawson
* Daniel Khan @danielkhan
* Josh Gavant @joshgav
* Andreas Madsen @AndreasMadsen
* Jan Krems @jkrems
* Eugene Ostroukhov @eugeneo
* Matthew Loring @matthewloring
* Sam Roberts @sam-github

## Today's Agenda

* build: require `--without-debugger` explicitly [node#12768](https://github.com/nodejs/node/pull/12768)
* Uniform way to trigger debugger on first line [node#12630](https://github.com/nodejs/node/issues/12630)
* inspector: JavaScript bindings for the inspector [node#12263](https://github.com/nodejs/node/pull/12263)
* Missing async callstacks on setTimeout [node#11370](https://github.com/nodejs/node/issues/11370)

### nodejs/diagnostics

* Node.js Collaboration Summit diagnostics discussion minutes [#95](https://github.com/nodejs/diagnostics/issues/95)
* \[docs\] requesting comments on diagnostics-howtos project [#92](https://github.com/nodejs/diagnostics/issues/92)

* \[trace event\] tracking issue
[#84](https://github.com/nodejs/diagnostics/issues/84)
* \[async_hooks\] tracking issue
[#29](https://github.com/nodejs/diagnostics/issues/29)

## Previous Meeting Review

* inspector: make `debug` an alias for `inspect`
[node#11441](https://github.com/nodejs/node/pull/11441)
* Switch the CLI debugger to V8 inspector
[node#11421](https://github.com/nodejs/node/issues/11421)
* \[WIP\] inspector: hint text update
[node#11207](https://github.com/nodejs/node/pull/11207)
* What will Domain be replaced with?
[node#10843](https://github.com/nodejs/node/issues/10843)

* \[trace\] tracking issue
[diagnostics#84](https://github.com/nodejs/diagnostics/issues/84)
* \[async_hooks\] tracking issue
[diagnostics#29](https://github.com/nodejs/diagnostics/issues/29)

## Minutes

### build: require `--without-debugger` explicitly
[node#12768](https://github.com/nodejs/node/pull/12768)

@refack suggests that --without-debugger be specified explicitly to ensure builders realize they're not getting any debugger when disabling SSL. Previously they got the old debugger and they may not realize that excluding SSL now excludes any debugger.

Next steps

* Continue discussion in GitHub.

### Uniform way to trigger debugger on first line
[node#12630](https://github.com/nodejs/node/issues/12630)

@joshgav: CTC recommended that `--inspect --debug-brk` be added back to accomodate IDEs, work is in progress in
<https://github.com/nodejs/node/pull/12949>.

No objections.

### inspector: JavaScript bindings for the inspector
[node#12263](https://github.com/nodejs/node/pull/12263)

@eugeneo: How to proceed? Technical comments have been addressed, more approvals are needed.

@sam-github: How do these bindings relate to `process._debug*` functions (see e.g. https://github.com/nodejs/node/pull/12777)?

`process._debug*` methods are for managing the debugger itself, whereas Inspector JS APIs are for interacting with debugger once it's *already running*.

`process._debug*` are existing APIs that were created for the old debugger. @eugeneo updated to support Inspector to address a reported issue.

The existing APIs can be used to start the debugger with: `process._debugProcess(process.pid)`.

Does that work on Windows? Yes, avoids signals on Windows.

@mhdawson: Should Inspector JS APIs be marked as experimental?

@ofrobots: These are meant to replace `vm.runInDebugContext` and that doesn't give an experimental warning.

@sam-github: The past is the past, should these be marked as experimental per our current standards?

@mhdawson: There are mixed opinions. For example, N-API is experimental and prints a warning.

@joshgav: Should we print an "experimental" warning on `require('inspector')`?
@watson: This API will be used by APM providers, and they don't want to print such warnings since it disturbs users.

@danielkhan: Warnings tend to be a distraction.

@ofrobots: Shouldn't rely on command-line flags to allow use of these APIs cause we need to replace `vm.runInDebugContext`.

@sam-github: In that case, why experimental? A: Cause we're not guaranteeing the API won't change.

@ofrobots: These are bindings to the existing Inspector API, which is quite stable now.

@sam-github: Should we mark it as stable right away?

@ofrobots: If stable already should have more docs etc.

Proposal: Land as experimental, but since we intend to move to stable quickly don't print a warning.

@mhdawson: Decision to print warning should be made case by case.

Next steps

* After reviews are in, okay to merge, no experimental warning needed.

### Missing async callstacks on setTimeout
[node#11370](https://github.com/nodejs/node/issues/11370)

@AndreasMadsen: In Chrome, you can click and get async callstacks for Web pages. When attached to Node, although you can check the box, it doesn't actually give async stacks.

We'll need to utilize async_hooks to make this work, so wanted to bring this to the attention of people working on async_hooks and Inspector.

@Fishrock123: Requires that every timer have an ID and we don't do that. Perhaps would be easier to use async IDs from async_hooks.

Next steps

* Review issue.

### Node.js Collaboration Summit diagnostics discussion minutes
[#95](https://github.com/nodejs/diagnostics/issues/95)

@watson: Discussion was about how to encourage ecosystem to make modules that are easier to instrument, how to teach them about async_hooks and using the embedder API, and how to help all module authors to emit events through a common mechanism. No concrete conclusions yet.

Could (should?) we get all diagnostic info from async_hooks? Conclusion was that there's still a lot of information that we don't get from async_hooks, such as the path to the file that's being opened.

The async_hooks init hook gives you a resource object with some properties, believe it has a file path.

@joshgav: There seem to be traces and logs outside of async context issues which also need to be tracked. Perhaps async_hooks should focus only on async context propagation?

@Fishrock123: If that's the idea why have a resource object in async_hooks?

@matthewloring: This comes up with PromiseHook too, need to include additional info like parent promise which spawned this promise.

@danielkhan: Would be good to document current state of affairs. Also good to specify a standard for module authors to write to. MongoDB's approach is a good blueprint.

How do we rationalize what to use for a shared channel? Some options:

1. Inspector through inspectorNotifications (after https://github.com/nodejs/node/pull/12263)
2. diagnostic-channel (https://github.com/Microsoft/node-diagnostic-channel)
3. async_hooks

Next steps

* @watson will open an issue summarizing current state and schedule a deep dive for stakeholders.

### \[docs\] requesting comments on diagnostics-howtos project [#92](https://github.com/nodejs/diagnostics/issues/92)

Postponing till next time, when @naugtur will discuss.

### \[trace event\] tracking issue [#84](https://github.com/nodejs/diagnostics/issues/84)

No comments today.

### \[async_hooks\] tracking issue [#29](https://github.com/nodejs/diagnostics/issues/29)

The following is Andreas’s prepared talking points included for convenience.

The too short story: async_hooks is likely going to land in node version 8.

Details for those not following closely the last couple days:

* Andreas (@AndreasMadsen) picked up lead on async_hooks PR because Trevor is busy. New PR https://github.com/nodejs/node/pull/12892
  * With help from Refael, Anna, and Jeremiah, async_hooks is now in master.
  * Some things are missing from the async_hooks PR, this is to get it merged fast and prevent merge conflicts.
    * PromiseHook, Matthew is assigned (@matthewloring).
    * C++ Embedder API (no one).
      * Documentation, Thorsten is assigned (@thlorenz). PR: https://github.com/nodejs/node/pull/12953
      * Some discussion about experimental API and its meaning (flags, warning, etc.) in https://github.com/nodejs/node/pull/12723 - we choose to let this affect the documentation PR, instead of the main PR.
      * Andreas asks to reduce API surface for now, to avoid domain mistakes.
        * Make the high-level JS Embedder API work like MakeCallback (minor change).
        * Don’t including the low-level JS Embedder API. There is two JS Embedder APIs.
        * High-level covers most embedder use cases.
        * Low-level is just for performance, node core uses this. But userland shouldn’t need it.
        * The existing API can be added in a semver-minor release if it is necessary.
* Andreas don’t have time to contribute with the last details, they needs to be assigned to other, but Andreas will be keep an eye on the progress and try to answer any questions.

@Fishrock123: Two remaining C++ issues:

1. Enable people to listen to messages from async_hooks.
2. MakeCallback used by Node addons (as opposed to the one in async-wrap.cc) needs to be tracked by async_hooks.

@watson: Will async_hooks be backported e.g. to 6.x?

@andreasmadsen: Anna has expressed interest in backporting. Not really breaking changes cause AsyncWrap has always been experimental.

@watson: Backporting would be valuable so that APM providers could use same API (async_hooks) everywhere, rather than behaving differently with older versions.

@sam-github: Needs a little longer to incubate in master.

@Fishrock123: Still considered experimental till addon MakeCallback and Promises work and docs are complete.

@mhdawson: Will this require changes in NAN and N-API too? A: Yes. @mhdawson to add to async_hooks tracking issue.

Q&A

None today.

Next meeting on Thursday May 25 at 12pm Pacific (1900 UTC).