> 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 - January 2017

* Date: 2017-01-19
* YouTube: http://youtu.be/sMkzvd9Lf74
* Hangouts on Air:
https://hangouts.google.com/hangouts/_/ytl/9unU1P2s7ypaAu9YbP-mxzklTAY1HCqDyZAiYP19S7s=?eid=100598160817214911030
* Minutes:
https://docs.google.com/document/d/146Y4TMLa0_uovREHalBsobspWSSog4N2k3TIJe_Co84
* Previous meeting:
https://github.com/nodejs/diagnostics/blob/master/wg-meetings/2016-12-02.md

## Attendees

* Josh Gavant @joshgav
* Matt Loring @matthewloring
* Ali Sheikh @ofrobots
* Mark Marron @mrkmarron
* Jason Ginchereau @jasongin
* Jeremiah Senkpiel @Fishrock123
* Richard Chamberlain @rnchamberlain
* Richard Lau @richardlau

## Agenda

### Inspector

* RFC: Support for subprocess inspection
[diagnostics#77](https://github.com/nodejs/diagnostics/issues/77)
* Support for HTTP/WS inspection
[diagnostics#75](https://github.com/nodejs/diagnostics/issues/75)
* src, lib: deprecate old debug-agent and CLI debugger
[node#10276](https://github.com/nodejs/node/pull/10276)
* inspector, doc: update hint text, add guide
[node#8978](https://github.com/nodejs/node/pull/8978)
* Bringing node-inspect into the fold
[diagnostics#67](https://github.com/nodejs/diagnostics/issues/67)
* Bring 'node-inspect' (CLI Debugger) under the Foundation
[tsc#190](https://github.com/nodejs/TSC/issues/190)
* deps: Add node-inspect
[node#10187](https://github.com/nodejs/node/pull/10187)

* Move bnoordhuis/node-heapdump to nodejs/node-heapdump
[post-mortem#40](https://github.com/nodejs/post-mortem/issues/40)

### Trace

* Trace Events trace system in Node.js
[diagnostics#53](https://github.com/nodejs/diagnostics/issues/53)
* trace_event.h tracking issue
[diagnostics#30](https://github.com/nodejs/diagnostics/issues/30)
* proposal: JS tracing APIs
[node-eps#48](https://github.com/nodejs/node-eps/pull/48)

### Async Context

* async_hooks initial implementation
[node#8531](https://github.com/nodejs/node/pull/8531)
* What will Domain be replaced with?
[node#10843](https://github.com/nodejs/node/issues/10843

### Post-Mortem

* [node-report](https://github.com/nodejs/nodereport)
* [llnode](https://github.com/nodejs/llnode)

## Previous Meeting

* https://github.com/nodejs/diagnostics/blob/master/wg-meetings/2016-12-02.md

### Trace

* Intended use cases (profiling, logging, etc.)
* JS API
* Conventions for category names and payload fields
* Trace points in core

Next steps:

* Add an "experimental" warning if --enable-tracing is specified.
* Discuss if/how to abstract from V8.
* If no objections, land initial PR (#9304).
* List next tasks in Diag repo, such as instrumenting the wraps in core, adding a JS API, and adding a callback (listener) mode.

### Inspector

* Deprecate `node --debug` (old Debugger interface)
* Deprecate CLI debugger (`node debug`)
* Replace CLI debugger with node-inspect
* Activate inspector in running process with SIGUSR1

Next steps:

* In issue for removal of old debugger node#9789 discuss what else will be effected/deprecated.
* Open a specific issue for deprecating existing CLI debugger (lib/_debugger.js, node debug).
* Open an issue in the TSC repo to bring node-inspect into the Foundation.
* Continue discussion on whether and how a CLI debugger should be bundled in default distribution.
* Reply in node#8464 that we plan to completely switch to inspector in v8.x.
* Notify client debugger projects about switch of SIGUSR1.
* Message changes to community.

### Async Context

Next steps:

* Review Embedder API added to async_hooks and start explaining to ecosystem.

## Minutes

### Inspector

*** RFC: Support for subprocess inspection [diagnostics#77](https://github.com/nodejs/diagnostics/issues/77)**

Consensus that this is a good idea, move forward with an initial implementation.

*** Support for HTTP/WS inspection [diagnostics#75](https://github.com/nodejs/diagnostics/issues/75)**

* Support Network domain in Inspector?
* Go through trace_events?

@Fishrock123: Chrome DevTools UI doesn’t provide a good interface for Node’s traffic load.

@joshgav: Consider from protocol perspective independent of presentation.

@Fishrock123: Network connections are already (able to be) wrapped by async_hooks, why an additional sink?

@ofrobots: Capture network events through trace_event first. That capture could be built on top of async_hooks.

@Fishrock123: For example, a native module which implements async_hooks for network events and sends events to trace system.

Do we want to also expose to Network domain in Inspector? To add to Inspector protocol would need trace listeners for TraceController, something @JasonGin has raised.

@ofrobots to discuss with @eugeneo.

Next steps:

*

*** src, lib: deprecate old debug-agent and CLI debugger [node#10276](https://github.com/nodejs/node/pull/10276)**

Proposal - separate deprecations into 2, land debug-agent deprecation in 7.x; CLI deprecation when node-inspect is integrated.

Will old debugger be gone in Node@8? Depends on whether we go with V8 5.7 or 5.8. 5.8 doesn't include old debugger. It also includes Turbofan and removes --expose-debug-as flag.

Do we suggest that Node@8 stay with V8 5.7 sso old debugger remains available? Yes.

If so should we still land deprecation in 7.x? No.

Get confirmation from CTC that V8 5.7 will be target for Node@8, then we can delay landing deprecation till 8.x.

Next steps

* Ask CTC to confirm target V8 version for Node@8.

*** inspector, doc: update hint text, add guide [node#8978](https://github.com/nodejs/node/pull/8978)**

@joshgav to move guide to nodejs.org following guides reorganization.

* Should we update the hint text? What should it be?
* How do we move out of experimental state? Do we have to move out before 8.x?

Since we're deprecating the old stuff in 8, seems new stuff should be out of experimental.

Need CTC vote/review to move Inspector out of experimental.

@Fishrock123: Open an issue/PR tagged ctc-agenda to bring out of experimental. Can be brought out in a minor.

What will the CTC want to know?
* That API (e.g. protocol, command-line flags) won't change radically in near future.
Current API: https://chromedevtools.github.io/debugger-protocol-viewer/v8/
Okay to add APIs.

Josh to ask ChakraCore about their needs and what it might require in current implementation.

Hint text:
* Should include link to docs.
* As long as experimental note that.
* Would be nice to include links for use with CDT and others.

Next steps

* @joshgav to move guide to nodejs.org following guides reorganization.
* Ask for CTC review to move out of experimental state preferably before 8.x. (@ofrobots)
* Josh to ask ChakraCore to review.
* Continue discussion on hint text in GH.

*** node-inspect**

  * Bringing node-inspect into the fold [diagnostics#67](https://github.com/nodejs/diagnostics/issues/67)
  * Bring 'node-inspect' (CLI Debugger) under the Foundation [tsc#190](https://github.com/nodejs/TSC/issues/190)
  * deps: Add node-inspect [node#10187](https://github.com/nodejs/node/pull/10187)

Do we want the `node inspect script.js` and/or `node debug script.js` experience?

Yes, good to have basic debugging built in a la `node debug`.

*** Move bnoordhuis/node-heapdump to nodejs/node-heapdump [post-mortem#40](https://github.com/nodejs/post-mortem/issues/40)**

@joshgav: Suggest integrating this functionality into node-inspect since Inspector provides this functionality.

@ofrobots: node-inspect may not fully address the use case here so we should consider this independently.

@Fishrock123: should this be added to the V8 module in core?

@joshgav: Should we instead factor V8 module out to indicate support for other VMs?

@Fishrock123 - just a name, could rename.

Next steps:

* Continue in GH.

### Trace

*** Tracking issues**

  * Trace Events trace system in Node.js [diagnostics#53](https://github.com/nodejs/diagnostics/issues/53)
  * trace_event.h tracking issue [diagnostics#30](https://github.com/nodejs/diagnostics/issues/30)

@joshgav to close existing issues and open new ones to track ongoing work on the following:

* add listen capability
* add JS API
* add trace points to core
* conventions for category names, payloads

* what is needed to land in a release, and as non-experimental?

*** proposal: JS tracing APIs [node-eps#48](https://github.com/nodejs/node-eps/pull/48)**

Purpose is not just for async, but for any perf diagnostics - counters, instant operations, begin/end for synchronous operations, etc.

Allows JS module authors to expose tracing data from their modules.

async_hooks is only for tracking async context.

In EP, seeking feedback on what should go in core or not?

Next steps

* Read doc and provide feedback to Jason.

### Async Context

* async_hooks initial implementation
[node#8531](https://github.com/nodejs/node/pull/8531)
* What will Domain be replaced with?
[node#10843](https://github.com/nodejs/node/issues/10843)

* docs and tests for async_hooks to be added to PR soon

@AndreasMadsen (comment):

A while ago Trevor said the following were missing:

* Remove all weak objects (will be handled in another PR to solve another issue)
* Add support to node::MakeCallback
* Implement public native API.
* Detect a change in the hooks queue when in the middle of running hooks, so the original set of hooks can finish executing.

My best guess is that only "Add support to node::MakeCallback" and "Implement public native API" is missing now, but that he wants to test the current implementation first.

### Post-Mortem

* [node-report](https://github.com/nodejs/nodereport)

Human-readable “dump.”

* [llnode](https://github.com/nodejs/llnode)

MDB story for core dumps on Linux and Mac.

### Q&A

Create a label for diag-wg-agenda.
Create a Doodle for next meeting in a few weeks.