diagnosticsPrivate - New Chrome API Proposal

You're about to propose a new app/extension API; what a marvelous road you have ahead of you! Before you venture forth, though, we'd like to get a sense for your new API and offer you some guidance on your way.

Instructions:

1. Fill out this proposal.
2. Set its sharing settings to *"Anyone with the link" "Can comment."* (Click Share->Change->"Anyone with the link" and switch "Can view" to "Can comment." Note that the "Anyone with the link" option will only be available if you've created the doc in your Chromium or personal Gmail account.)
3. Follow the rest of the Proposal Phase steps.

Proposal Date
2013-06-19

Who are the primary Eng and PM contacts for this API? (email addresses)
ikarienator, ebeach, xiaowenx

Which team will be responsible for this API? (team email address)
chrome-apps-team

What is the proposed namespace?
chrome.diagnosticsPrivate

What's a reasonable Chrome milestone target for landing most of the code on Trunk?
Chrome OS implementation in R29, the rest in R30.

Overview
We'd like to have a set of APIs useful for diagnostics tools, and initially, an API that can be used to replicate ping and traceroute functionality.

Use cases
The sendPacket API is used by a diagnostics tool to find problems in the network configuration of an environment and determine latency issues.

Do you know anyone else, internal or external, that is also interested in this API?
Whereas the initial use-case for the sendPacket API is to create a tool that figures out why the user isn't connected at all, future tools could use this to determine latencies in the routing of high-bandwidth traffic, such as needed by large video sites (e.g. the one owned by Google).

Could this API be part of the web platform?
Don't think so.

Does this API expose any functionality to the web?
No

Do you expect this API to be fairly stable? How might it be extended or changed in the future?
Yes. Though it's possible we might want to expose more parameters in the future, we're not expecting to change this anytime soon.

If multiple apps/extensions used this API at the same time, could they conflict with each others? If so, how do you propose to mitigate this problem?
There are no contention issues.

List every UI surface belonging to or potentially affected by your API:
No effect on UI.

Actions taken with app/extension APIs should be obviously attributable to an app/extension. Will users be able to tell when this new API is being used? How?
Not visible to users.

How could this API be abused?
Could potentially be used to DOS a site if enough devices are pinging together.

Imagine you're Dr. Evil Extension Writer, list the three worst evil deeds you could commit with your API (if you've got good ones, feel free to add more):
1) DOS an external site.
2) Slow down current device by transmitting a lot of traffic through ping.

What security UI or other mitigations do you propose to limit evilness made possible by this new API?
For now, let's leave this as a private API. In the future, if we expose it to the public, we can consider a throttling mechanism.

Alright Doctor, one last challenge:
Could a consumer of your API cause any permanent change to the user's system using your API that would not be reversed when that consumer is removed from the system?
No

How would you implement your desired features if this API didn't exist?
In some cases, we could try to make TCP connections instead, but won't tell us the route the traffic took to get there, nor does it work with gateways/routers that don't have TCP ports open.

Draft Manifest Changes
chrome.diagnosticsPrivate permission

Draft API spec
For the appropriate style, please refer to existing APIs.

`dictionary` **`SendPacketOptions`**
The options for sendPacket.

- `DOMString` **`ip`** The targeted IP address for the ping.
- `SendPacketType` **`type`** Type of the ping (tcp, udp, icmp).
- `long` **`ttl`** Time-to-live of the ping.
- `long` **`timeout`** Timeout in second the operation.
- `long` **`size`** Size of the payload.

`dictionary` **`SendPacketResult`**
The result of sendPacket.

- `DOMString` **`ip`** The IP address where the reply is from.
- `long` **`latency`** The latency in ms of the reply.

sendPacket
`chrome.diagnosticsPrivate.sendPacket(SendPacketOptions options, function callback)`

Send a packet of the given type with the given parameters.

Parameters
`options` (`SendPacketOptions`)
Options for the call.
`callback` (function)

Callback
The callback parameter should specify a function that looks like the following. This is called when the ping returns or times out. This callback will be called once and exactly once for each invocation of sendPacket.

```
function(SendPacketResult result) { ... };
```

If we get an error back, `chrome.runtime.lastError` will be set.

Examples of return values:

- Non-error return values (same definition of error as /bin/ping)
  - ip set to the destination IP
  - latency set to number of seconds for normal ECHO responses or 0 for timeout.
- Error return values
  - ip set to the IP address of the server that returned the error
  - latency set to 0
  - chrome.runtime.lastError set to a string, e.g.
    - Time to live exceeded
    - Destination Host Unreachable

The following are APIs we're currently considering to also be in this namespace. They are still pretty half-baked, so we're not proposing to discuss them yet. They're here right now to give an idea of where we're going with this.

startNetInternalsLog

```
chrome.diagnosticsPrivate.startNetInternalsLog(function callback);
```

Start capturing net-internals logs in the background. A diagnostics tool would use this to start logging before it runs its tests (related bug). Because running a constant log in the background uses up resources and degrades performance, this API will automatically stop logging after five minutes. In order to continue logging, please call this API again.

stopNetInternalsLog

```
chrome.diagnosticsPrivate.stopNetInternalsLog(callback());
```

Stop capturing net-internals logs. When this is called, dump the logs into a directory that the feedback report mechanism collects from (a location under /var/logs/ would be ideal).

readLogFile

```
chrome.diagnosticsPrivate.readLogFile(
  "path",
  startChar,
  callback(contents),
  errorCallback(error));
```

Return the contents of the given log file. In order to prevent abuse, a few constraints should be imposed:

- Only allow reading the contents of files in the /var/log directory and its subdirectories. If the file is not found, or not in /var/log, then return `FileError.NOT_FOUND_ERR`.
- Only read back the first 4096 characters starting with the index specified in `startChar`.

testStorage

```
chrome.diagnostics.testStorage(callback(storageInfo))
```

testMemory

```
chrome.diagnostics.testMemory(callback(memoryInfo))
```

testBattery

```
chrome.diagnostics.testBattery(callback(batteryInfo))
```

Open questions

Note any unanswered questions that require further discussion.