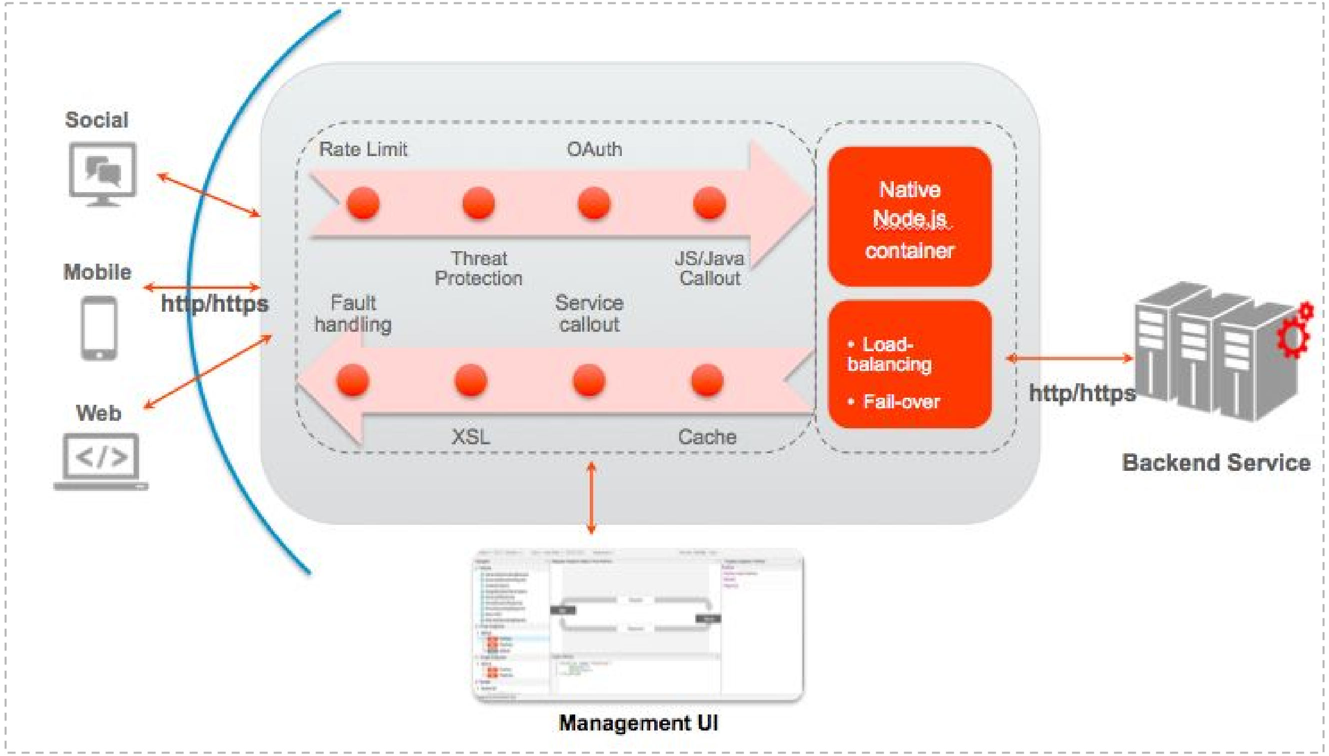
Node.js Integration with Apigee Edge

Google Cloud

Node.js on Apigee Edge



Node.js as a Target Endpoint

HTTP Target

```
<TargetEndpoint name="default">
  <Description/>
  <Flows/>
  <PreFlow name="PreFlow">
    <Request/>
    <Response/>
  </PreFlow>
  <Flows/>
  <PostFlow name="PostFlow">
    <Request/>
    <Response/>
  </PostFlow>
  <HTTPTargetConnection>
    <URL>http://mocktarget.apigee.net/</URL>
  </HTTPTargetConnection>
</TargetEndpoint>
```

apigee

Node.js as a Target Endpoint

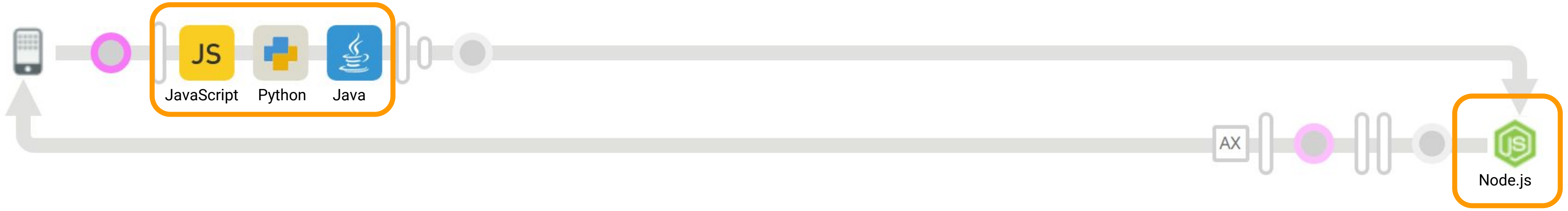
HTTP Target

```
<TargetEndpoint name="default">
  <Description/>
  <Flows/>
  <PreFlow name="PreFlow">
    <Request/>
    <Response/>
  </PreFlow>
  <Flows/>
  <PostFlow name="PostFlow">
    <Request/>
    <Response/>
  </PostFlow>
  <HTTPTargetConnection>
    <URL>http://mocktarget.apigee.net/</URL>
  </HTTPTargetConnection>
</TargetEndpoint>
```

Node.js

```
<TargetEndpoint name="default">
  <Description/>
  <FaultRules/>
  <PreFlow name="PreFlow">
    <Request/>
    <Response/>
  </PreFlow>
  <Flows/>
  <PostFlow name="PostFlow">
    <Request/>
    <Response/>
  </PostFlow>
  <ScriptTarget>
    <ResourceURL>node://hello-world.js</ResourceURL>
  </ScriptTarget>
</TargetEndpoint>
```

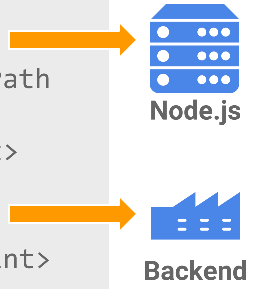
Node.js as a Target Endpoint



Node.js as a Target Endpoint

```
<ProxyEndpoint>
  ...

  <RouteRule name="node">
    <Condition>proxy.pathsuffix MatchesPath
"/queue"</Condition>
    <TargetEndpoint>node</TargetEndpoint>
  </RouteRule>
  <RouteRule name="directToBackend">
   <TargetEndpoint>backend</TargetEndpoint>
  </RouteRule>
</ProxyEndpoint>
```



Node.js as a Target Endpoint

```
<TargetEndpoint name="node">
  <Description>Node Target</Description>
  <ScriptTarget>
    <Properties>
      <Property name="success.codes">2XX,3XX,4XX</Property>
    </Properties>
    <EnvironmentVariables>
      <!-- process.env.var -->
      <EnvironmentVariable name="var">VALUE</EnvironmentVariable>
    </EnvironmentVariables>
    <Arguments>
      <Argument>argument</Argument>
    </Arguments>
    <ResourceURL>node://app.js</ResourceURL>
  </ScriptTarget>
</TargetEndpoint>
```

When to use Node.js?

When to use Node.js?

- no existing policy

When to use Node.js?

- no existing policy
- asynchronous processing logic

When to use Node.js?

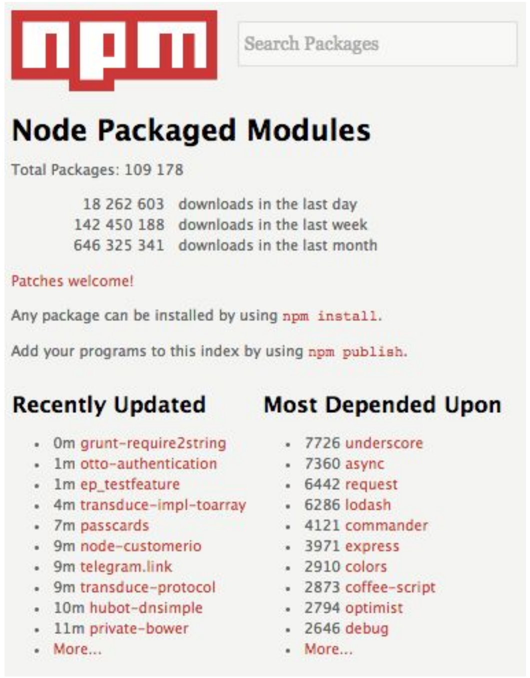
- no existing policy
- asynchronous processing logic
- non HTTP backend protocols

When to use Node.js?

- no existing policy
- asynchronous processing logic
- non HTTP backend protocols
- Common use cases:
    - Non-http backend
    - Async execution
    - Complex mashups
    - Bulk operations
    - Mockups and quick demos

When to use Node.js?

Module reuse



Node Packaged Modules

Total Packages: 109 178

```
  18 262 603   downloads in the last day
 142 450 188   downloads in the last week
 646 325 341   downloads in the last month
```

Patches welcome!

Any package can be installed by using `npm install`.

Add your programs to this index by using `npm publish`.

Recently Updated

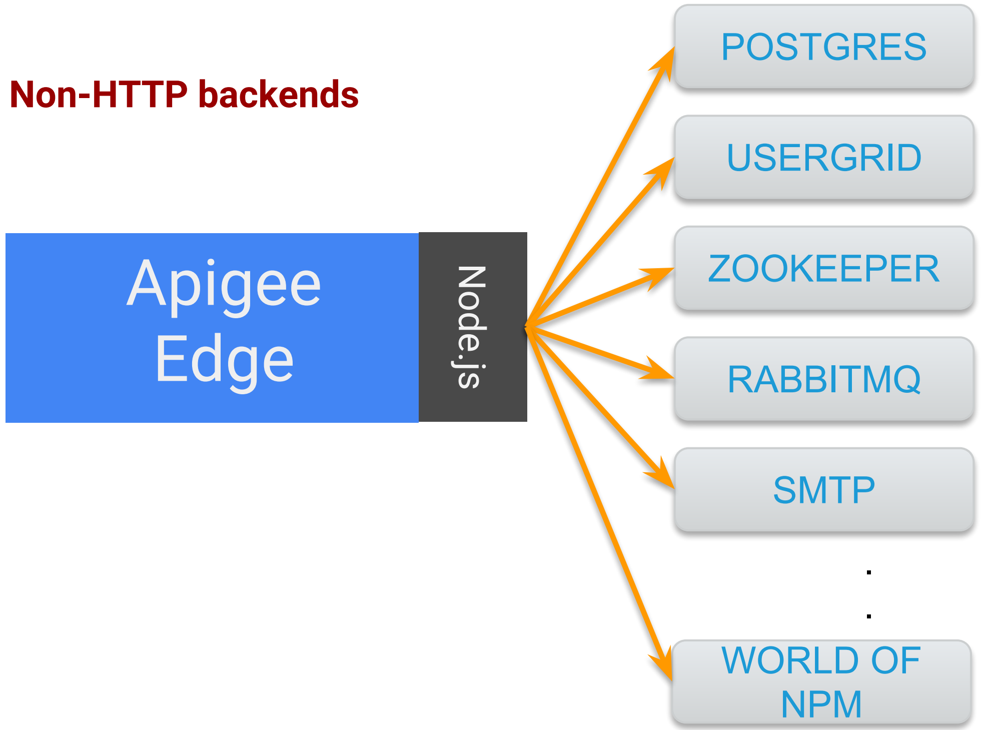
- 0m grunt-require2string
- 1m otto-authentication
- 1m ep_testfeature
- 4m transduce-impl-toarray
- 7m passcards
- 9m node-customerio
- 9m telegram.link
- 9m transduce-protocol
- 10m hubot-dnsimple
- 11m private-bower
- More...

Most Depended Upon

- 7726 underscore
- 7360 async
- 6442 request
- 6286 lodash
- 4121 commander
- 3971 express
- 2910 colors
- 2873 coffee-script
- 2794 optimist
- 2646 debug
- More...

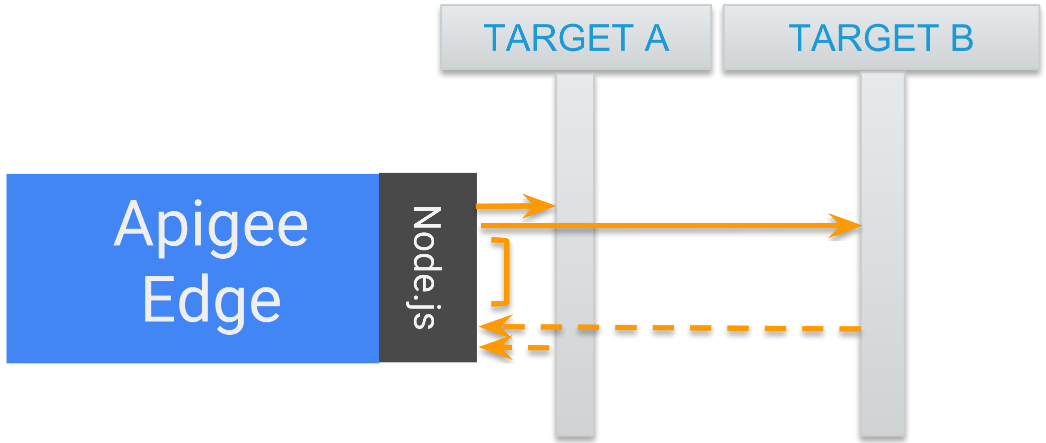
When to use Node.js?

Non-HTTP backends



When to use Node.js?

Async operations



When to use Node.js?

Module reuse - async example

```
async.waterfall([
    function(callback){
        callback(null, 'one', 'two');
    },
    function(arg1, arg2, callback){
      // arg1 now equals 'one' and arg2 now equals
'two'
        callback(null, 'three');
    },
    function(arg1, callback){
        // arg1 now equals 'three'
        callback(null, 'done');
    }
], function (err, result) {
   // result now equals 'done'
});
```

```
async.parallel([
    function(callback){
        setTimeout(function(){
            callback(null, 'one');
        }, 200);
    },
    function(callback){
        setTimeout(function(){
            callback(null, 'two');
        }, 100);
    }
],
// optional callback
function(err, results){
    // the results array will equal ['one','two'] even though
    // the second function had a shorter timeout.
});
```

apigee

When NOT to use Node.js?

When NOT to use Node.js?

- existing policy can do the job

When NOT to use Node.js?

- existing policy can do the job
- cannot execute a policy in the middle of your script.

When NOT to use Node.js?

- existing policy can do the job
- cannot execute a policy in the middle of your script.
- cannot execute full Node.js script multiple times

apigee-access

apigee-access

- open source Node.js module
- access Apigee Edge specific functionality like

apigee-access

- open source Node.js module
- access Apigee Edge specific functionality like
    - access and modify "flow variables"

apigee-access

- open source Node.js module
- access Apigee Edge specific functionality like
    - access and modify "flow variables"
    - retrieve data

apigee-access

- open source Node.js module
- access Apigee Edge specific functionality like
    - access and modify "flow variables"
    - retrieve data
    - distributed cache

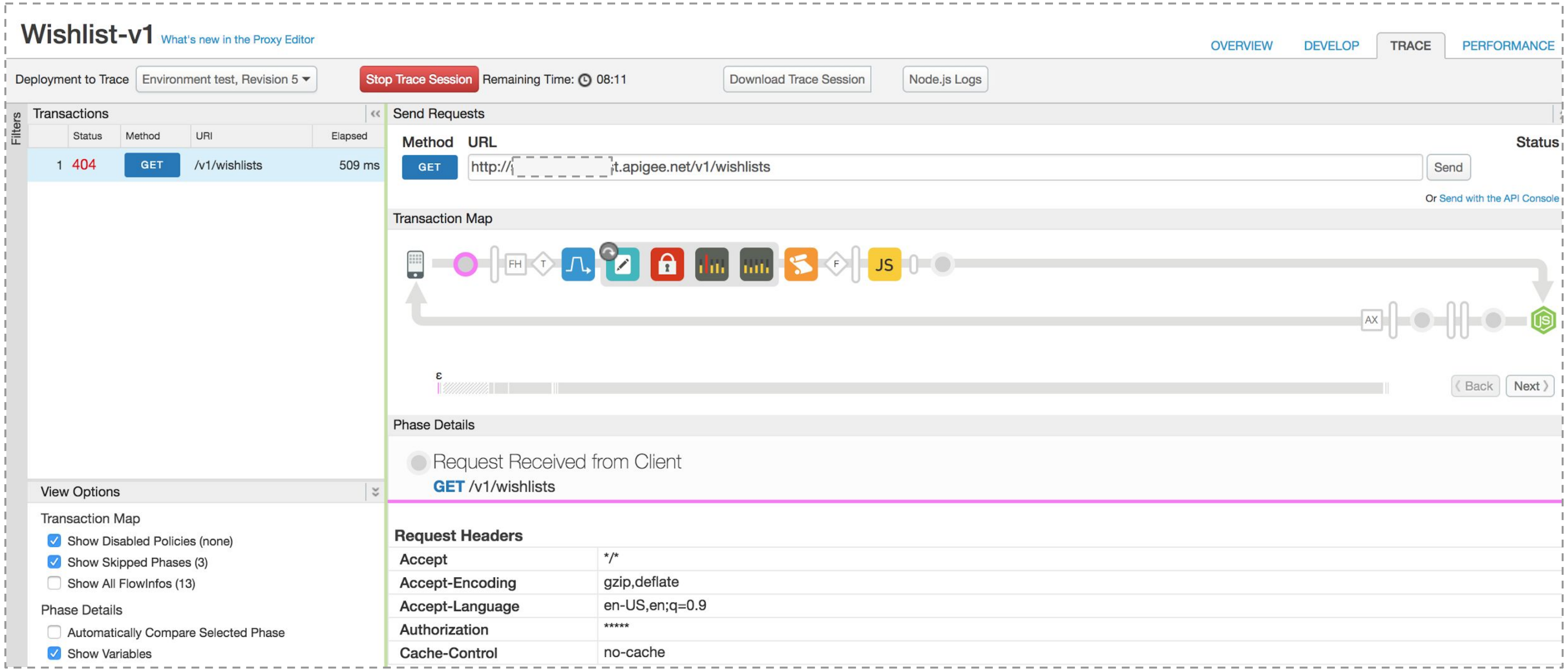
apigee-access

- open source Node.js module
- access Apigee Edge specific functionality like
  - access and modify "flow variables"
  - retrieve data
  - distributed cache
  - distributed quota service

apigee-access

- open source Node.js module
- access Apigee Edge specific functionality like
    - access and modify "flow variables"
    - retrieve data
    - distributed cache
    - distributed quota service
    - OAuth service

Debugging Node.js



Wishlist-v1 What's new in the Proxy Editor

OVERVIEW DEVELOP TRACE PERFORMANCE

Deployment to Trace: Environment test, Revision 5 | Stop Trace Session | Remaining Time: 08:11 | Download Trace Session | Node.js Logs

Transactions

	Status	Method	URI	Elapsed
1	404	GET	/v1/wishlists	509 ms

Send Requests

Method: GET
URL: http://[redacted]t.apigee.net/v1/wishlists
Send

Or Send with the API Console

Transaction Map

Phase Details

Request Received from Client
GET /v1/wishlists

Request Headers

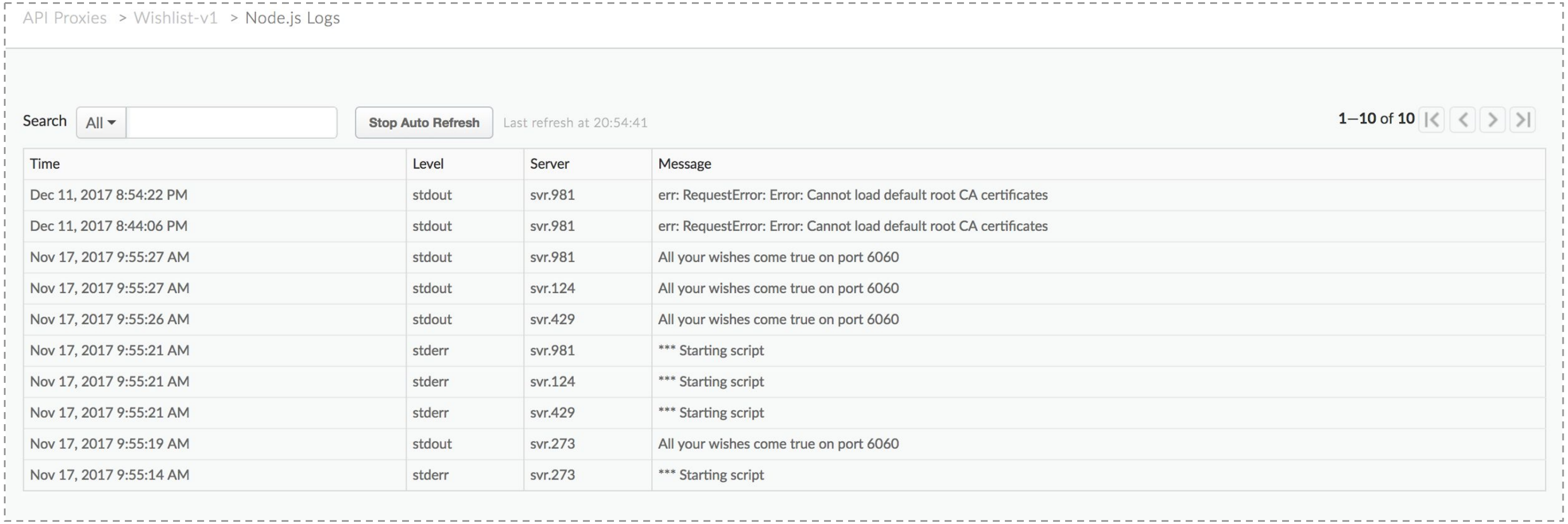
Accept	*/*
Accept-Encoding	gzip,deflate
Accept-Language	en-US,en;q=0.9
Authorization	*****
Cache-Control	no-cache

View Options

Transaction Map
- [✓] Show Disabled Policies (none)
- [✓] Show Skipped Phases (3)
- [ ] Show All FlowInfos (13)

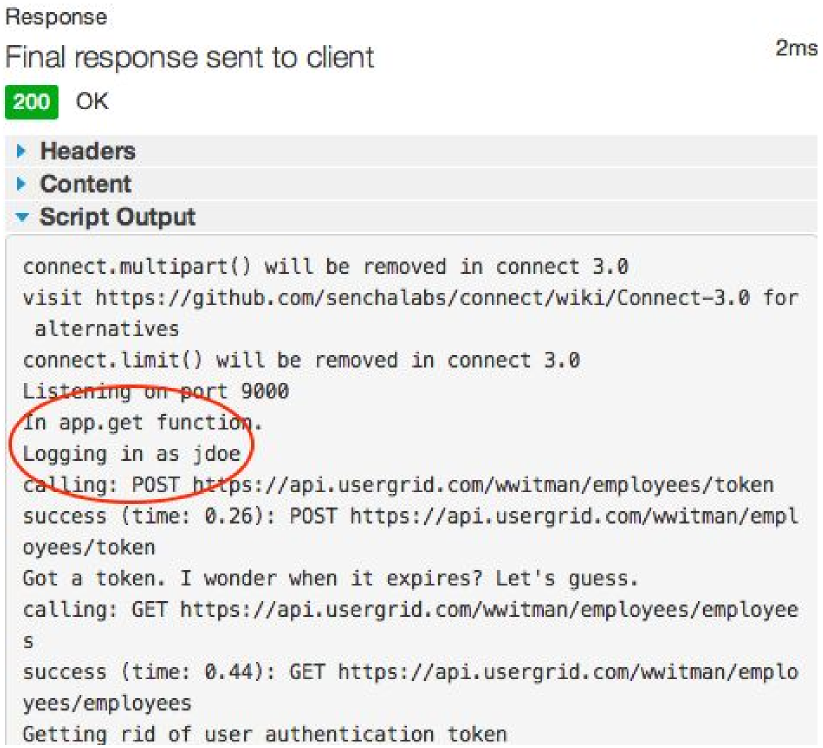
Phase Details
- [ ] Automatically Compare Selected Phase
- [✓] Show Variables

Debugging Node.js



Time	Level	Server	Message
Dec 11, 2017 8:54:22 PM	stdout	svr.981	err: RequestError: Error: Cannot load default root CA certificates
Dec 11, 2017 8:44:06 PM	stdout	svr.981	err: RequestError: Error: Cannot load default root CA certificates
Nov 17, 2017 9:55:27 AM	stdout	svr.981	All your wishes come true on port 6060
Nov 17, 2017 9:55:27 AM	stdout	svr.124	All your wishes come true on port 6060
Nov 17, 2017 9:55:26 AM	stdout	svr.429	All your wishes come true on port 6060
Nov 17, 2017 9:55:21 AM	stderr	svr.981	*** Starting script
Nov 17, 2017 9:55:21 AM	stderr	svr.124	*** Starting script
Nov 17, 2017 9:55:21 AM	stderr	svr.429	*** Starting script
Nov 17, 2017 9:55:19 AM	stdout	svr.273	All your wishes come true on port 6060
Nov 17, 2017 9:55:14 AM	stderr	svr.273	*** Starting script

Debugging Node.js



Response

Final response sent to client

2ms

200 OK

```
connect.multipart() will be removed in connect 3.0
visit https://github.com/senchalabs/connect/wiki/Connect-3.0 for
 alternatives
connect.limit() will be removed in connect 3.0
Listening on port 9000
In app.get function.
Logging in as jdoe
calling: POST https://api.usergrid.com/wwitman/employees/token
success (time: 0.26): POST https://api.usergrid.com/wwitman/empl
oyees/token
Got a token. I wonder when it expires? Let's guess.
calling: GET https://api.usergrid.com/wwitman/employees/employee
s
success (time: 0.44): GET https://api.usergrid.com/wwitman/emplo
yees/employees
Getting rid of user authentication token
```

Thank You

Node.js* runtime

- Node.js runtime on Edge is Trireme

`http://github.com/apigee/trireme`

*Node.js is a trademark of Joyent, Inc. and is used with its permission. We are not endorsed by or affiliated with Joyent.

What is Trireme

- Edge open-source project
- Set of libraries for running node.js scripts inside JVM
- Specifically designed to be embeddable within any Java program
  - “HTTP Adapter” lets it run inside existing containers
  - “Sandbox” restricts file and network I/O access

Why?

- We wanted to add node.js capabilities to our existing product which is already built using Java
- We wanted to use Java code from node.js
- We didn’t want to assemble a node.js PaaS
- We wanted script isolation – there is no way for one script to affect the heap of others
- We wanted sandboxed execution
    - Prevent script from gaining access to file system and local network
    - Limit execution time of a script – preventing infinite loops

Node.js implementation

- Node.js = JavaScript shell + native modules in C++
- Trireme exposes Java modules that mimic the interfaces of the C++ native modules in node.js.

Architecture

- One thread per Node.js application
  - Async I/O handled via NIO within that thread
- Additional thread pool for blocking operations
  - File I/O
  - DNS lookups
- Replaces native code from Node.js with Java alternatives
  - Internal modules such as “tcp_wrap”, etc.
- Implements a few popular native modules with Java code
  - “iconv”, “node_xslt” (replaced by npm trireme-xslt), etc.

Why would you care

- It is an open-source project that can be utilized outside Edge
    - If you want to embed Node.js apps inside existing Java application
    - If you want to run Node.js apps that take advantage of Java libraries you can't live without, e.g. JDBC, XML parsers, XSLT engines
- As it is the Node.js runtime used, it has significant impact on design and architecture of your solution that you need to know about
    - Node.js 0.10 is supported
    - Uses Rhino (JavaScript implementation for JVM) which only implements JavaScript 1.8
    - Trireme does not support 100% Node.js APIs

Node.js vs Trireme

Node.js

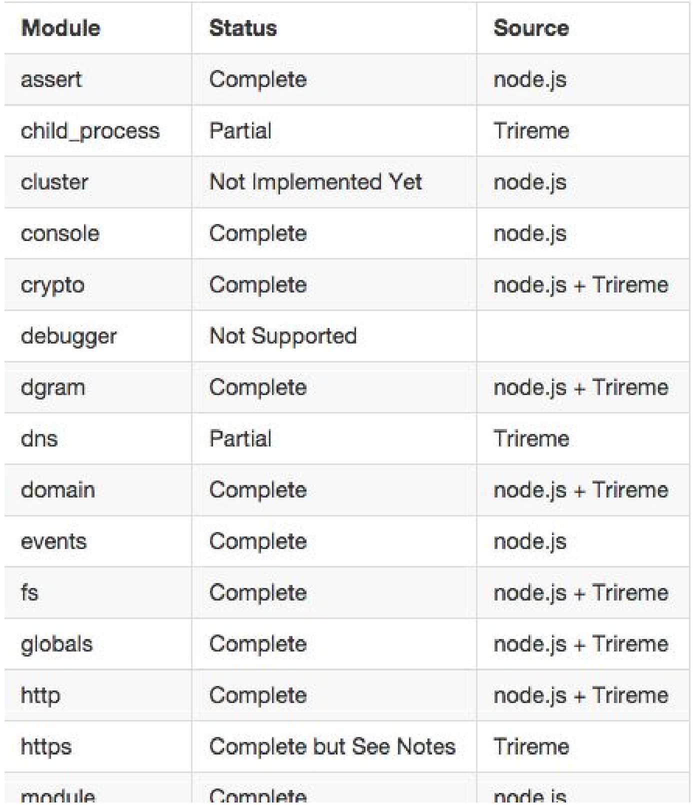
- Single-threaded event engine
  - Non-blocking TCP I/O
  - Non-blocking UDP datagrams
  - Non-blocking File I/O
  - Timers
- “Buffer” object
- Module loading system
- Utility modules
- Third-party components
  - V8 JavaScript engine (runtime)
  - OpenSSL (encryption)
  - ZLib (compression)

Trireme

- Single-threaded event engine
  - Non-blocking TCP I/O
  - Non-blocking UDP datagrams
  - Non-blocking File I/O
  - Timers
- “Buffer” object
- Module loading system
- Utility modules
- Third-party components
  - Rhino JavaScript engine (runtime)
  - Bouncy Castle (crypto, optional)
  - Java SE (compression)

Compatibility

- Can’t load native code (can’t load C code)
- https://github.com/apigee/trireme#how-complete-is-trireme



Module	Status	Source
assert	Complete	node.js
child_process	Partial	Trireme
cluster	Not Implemented Yet	node.js
console	Complete	node.js
crypto	Complete	node.js + Trireme
debugger	Not Supported	
dgram	Complete	node.js + Trireme
dns	Partial	Trireme
domain	Complete	node.js + Trireme
events	Complete	node.js
fs	Complete	node.js + Trireme
globals	Complete	node.js + Trireme
http	Complete	node.js + Trireme
https	Complete but See Notes	Trireme
module	Complete	node.js

Edge restrictions

- Script to write files or to read outside of the current local directory tree where it was deployed
- Script to listen on an incoming port (tcp_wrap to bind TCP socket without listening)
- It can open outgoing ports all it wants
- child_process
- cluster
- debugger
- dgram
- readline
- repl
- tty

When to use Node.js?

Try out of the box policies first

- Configuration over code, aka policy over code
- Speed / Agility
  - Let's implement a distributed quota
  - Use an Edge policy implement from scratch
- Quality
  - "Developed/Tested" once, "configured" everywhere
  - Maintained centrally

```
<Quota name="Quota.DailyPerApp">
    <Interval>1</Interval>
    <TimeUnit>day</TimeUnit>
    <Distributed>true</Distributed>
    <Synchronous>true</Synchronous>
    <Identifier ref="request.queryparam.apikey" />
    <Allow count="100" />
</Quota>
```

apigee

When to use Node.js?

Then consider callout policies

- Simple functionality which can be visualized as a step?
  - Get/set variable
  - Make custom/simple modifications to request/response?
  - Request/response data validations?
- Edge supports Java, JavaScript, Python as callout policies

```
//offset parameter validation
var offset =
context.getVariable('request.queryparam.offset');
if (offset != null) {
    if (offset < 1) {
        context.setVariable('errorCode', '400.02.001');
        context.setVariable('errorMessage', 'offset
query parameter value is invalid');
    }
}
```

Testing Node.js

- Testing Node.js applications
  - Grunt.js
  - Mocha
  - Chai
- TDD

Troubleshooting Node.js

- Handling dependencies and dependency conflicts
  - Express 3.7 and Connect 3.0 (deprecation notices and removal of middleware)
  - Conflicting module versions in Edge
- Handling response errors and node script failures/crashes
  - Error handling in JavaScript (try/catch)
  - Syntax errors in Node.js
  - nodejitsu forever

Troubleshooting Node.js

- The node-inspector tool
  - Browser-based Node.js debugger
  - Navigate in your source files
  - Set breakpoints (and specify trigger conditions)
  - Step over, step in, step out, resume (continue)
  - Inspect scopes, variables, object properties
- IntelliJ IDEA
  - A feature-rich GUI builder for Node.js
  - (Should be) free with the community edition

Troubleshooting Node.js

- Getting help
  - StackOverflow has an incredible wealth of Node.js knowledge
  - How to ask: http://stackoverflow.com/help/how-to-ask
- Edge docs
  - http://apigee.com/docs/api-services/content/getting-started-nodejs-apigee-edge