Logagent plugins

The architecture of logagent is modular and each input or output module is implemented as plugin for the logagent framework. Plugins are loaded on demand depending on the configurations.

How plugins work

- Logagent checks the configuration file for properties with a "module" key for the nodejs module name. External plugins needs to be installed via npm.
- Plugins are initialized with the logagent configuration from (command line arguments + configuration file) and the event emitter for logagent. Plugins should provide a start and stop method.
- Input plugins read data from a data source and emit events to logagent event emitter. This events have the identifier "data.raw" and 2 parameters:
- data - data read from a data source
- context - an object with meta data e.g. {sourceName: '/var/log/httpd/access.log'} The "context" helps other plugins to process the data coreectly, e.g. to handle multiple open files.
- Output plugins listen to "data.parsed" events and store or forward the data to the target.

Examples

Input plugin

This example implements a plugin to receive data via TCP socket with a configurable rate limit.

The plugin config file:

```
# Global options
input:
  tcp:
    module: input-tcp
    port: 45900
    bindAddress: 0.0.0.0
    sourceName: tcpTest
output:
  # print parsed logs in YAML format to stdout
  stdout: yaml
```

Node.js source code:

```
'use strict'
var split = require('split2')
var net = require('net')
var safeStringify = require('fast-safe-stringify')

/**
 * Constructor called by logagent, when the config file contains tis entry:
 * input
 *  tcp:
 *    module: megastef/logagent-input-tcp
 *    port: 4545
 *    bindAddress: 0.0.0.0
 *
 * @config cli arguments and config.configFile entries
 * @eventEmitter logent eventEmitter object
 */
function InputTCP (config, eventEmitter) {
  this.config = config.configFile.input.tcp
 this.config.maxInputRate = config.configFile.input.tcp.maxInputRate || config.maxInputRate
  this.eventEmitter = eventEmitter
}
module.exports = InputTCP
/**
 * Plugin start function, called after constructor
 *
 */
InputTCP.prototype.start = function () {
  if (!this.started) {
    this.createServer()
    this.started = true
  }
}

/**
 * Plugin stop function, called when logagent terminates
 * we close the server socket here.
 */
InputTCP.prototype.stop = function (cb) {
  this.server.close(cb)
}

InputTCP.prototype.createServer = function () {
  var self = this
  this.server = net.createServer(function (socket) {
    // Context object, the source name is used to identify patterns
   var context = { name: 'input.tcp', sourceName: self.config.sourceName || socket.remoteAddres
```

```
   socket.pipe(Throttle(self.config.maxInputRate)).pipe(split()).on('data', function emitLine
      // emit a 'data.raw' event for each line we receive
      self.eventEmitter.emit('data.raw', data, context)
      if (self.config.debug) {
        console.log(data, context)
      }
    }).on('error', console.error)
  /*
  // We could return parsed objects to the client
  // Logagent will emit "data.parsed" events
  self.eventEmitter.on('data.parsed', function (data, aContext) {
    socket.write(safeStringify(data) + '\n')
  })
  */
  })
  var port = this.config.port || 4545
  var address = this.config.bindAddress || '0. 0.0.0'
  this.server.listen(port, address)
  console.log('listening to ' + address + ':' + port)
}

// helper  to throttle bandwidth
var StreamThrottle = require('stream-throttle').Throttle
function Throttle (maxRate) {
  var inputRate = maxRate || 1024 * 1024 * 100
  var chunkSize = inputRate / 10
  if (chunkSize < 1) {
    chunkSize = 1
  }
  return new StreamThrottle({
    chunksize: chunkSize,
    rate: inputRate || 1024 * 1024 * 100
  })
}
```

Example output plugin

```
'use strict'
var prettyjson = require('prettyjson')
var safeStringify = require('fast-safe-stringify')
function OutputStdout (config, eventEmitter) {
  this.config = config
  this.eventEmitter = eventEmitter
}
```

```
OutputStdout.prototype.eventHandler = function (data, context) {
  if (this.config.suppress) {
    return
  }
  if (this.config.pretty) {
    console.log(JSON.stringify(data, null, '\t'))
  } else if (this.config.yaml) {
    console.log(prettyjson.render(data, {noColor: false}) + '\n')
  } else {
    console.log(safeStringify(data))
  }
}

OutputStdout.prototype.start = function () {
 this.eventEmitter.on('data.parsed', this.eventHandler.bind(this))
}

OutputStdout.prototype.stop = function (cb) {
 this.eventEmitter.removeListener('data.parsed', this.eventHandler)
  cb()
}

module.exports = OutputStdout
```