Gebrauchsanweisung zu »nodejsServers2«

Rolf Niepraschk, AG 7.54

2014-01-27

Einleitung

Bei »nodejsServers2« handelt es sich um ein auf »nodejs« basierendes Programm, welches als sogenannter »Dämon« auf einem Linux-Rechner gestartet wird, d. h. es läuft solange auch der Linux-Rechner läuft. »nodejsServers2« bietet einen Netzwerkzugänge in Form mehrerer http-Server auf unterschiedlichen Ports. Mit dieser Hilfe empfängt »nodejsServers2« Daten von außerhalb befindlichen Programmen (z. B. Web-Applikationen) und sendet daraufhin verschiedenartige Daten zurück. Der Sinn ist, auf diese Weise Funktionen anzubieten, die ohne »nodejsServers2« nicht oder nur in komplizierter Weise zur Verfügung stehen würden.

1. relay-Server – Port: 55555

Die Komponente »relay-Server« bietet den Zugang zu einer Vielzahl von externen auf dem Linux-Rechner installierten Programmen und Netzwerkprotokollen. Web-Applikationen sind beispielsweise damit in der Lage die sich aus der sogenannten »Same-Origin-Policy« ergebenen Einschränkungen zu überwinden oder Daten per Netzwerkzugriff zu erhalten, die normalerweise über diesen Weg nicht zugänglich sind.

Die Anforderung an den »relay-Server« geschieht immer über eine Datenstruktur im JSON-Format. Diese enthält mindestens einen mit »Action« benannten Wert, der die auszuführende Aktion auswählt. Sogenannte »externe Aktionen« beginnen mit dem Zeichen »/« und bilden den Dateipfad zu einem auf dem Rechner installierten Programm, welches gestartet werden soll. Es sind nur die im Folgenden aufgeführten Programme erlaubt. Aktionen, die nicht das Zeichen »/« enthalten, werden als »interne Aktionen« bezeichnet, da sie mit einer Ausnahme ohne Start eines besonderen Programms auskommen, also einzig die Möglichkeiten der Programmiersprache »JavaScript« in der durch »nodejs« angebotenen Form nutzen.

1.1. Externe Aktionen

Action – kennzeichnet das Programm, welches von »NodejsRelay« gestartet wird.

- `"/usr/local/bin/vxiTransceiver"`:

  Kommunikation über das VXI-11-Protokoll mit Messgeräten. Die weiteren Parameter:

  Host – IP-Adresse oder Rechnername des Messgerätes (zwingend)

Device – die Geräteadresse (zwingend)

Value – der String zum Auslösen eines bestimmten Gerätebefehls (zwingend)

VxiTimeout – Zeit in ms, die auf eine Rückmeldung vom Gerät gewartet wird (optional, Standard: 2000). Der Wert »0« kann benutzt werden, um bei fehlerhaften GPIB-Geräten einen »timeout error« zu vermeiden, wenn diese zwar Daten aber kein korrektes Ende-Signal senden.

Beispiel:

```
echo '{"Action":"/usr/local/bin/vxiTransceiver","Host":"e75481",
  "Device":"gpib0,5","Value":"*IDN?"}' | \
  curl -T - -X PUT http://localhost:55555
```

Rückgabe:

```
{
"t_start":1385556963144,"t_stop":1385556963157,
"exitCode":0,
"t__start":1385556963131,"t__stop":1385556963161,  ???
"Result":["HEWLETT-PACKARD,34970A,0,13-2-2\n"]
}
```

Die vom Gerät gesendete Antwort ist dem Kennwort »Result« zugeordnet. »exitCode« ist der Rückgabecode des Programmaufrufs (in der Shell »$?«) und »t_start«/»t_start« kennzeichnen die Dauer des Aufrufes in ms. Falls die absolute Zeit von Interesse ist, kann sie mit

```
date -d @$[1385556963157 / 1000]
```

in Sekundengenauigkeit dargestellt werden:

```
Mi 27. Nov 13:56:03 CET 2013.
```

- "/usr/bin/Rscript":

  Senden von Programmcode an die Statistik-Software »R«. Die weiteren Parameter:

  Body – R-Programmcode in Form eines Strings oder eines Arrays von Strings (zwingend). Im zweiten Falle wird aus dem Array intern ein einzelner String erzeugt, wobei nach jedem Inhalt eines Array-Elements ein Zeilenumbruch (»\n«) eingefügt wird.

  Value – Zusätzliche Parameter für den Aufruf des Programms »Rscript« (optional, String oder String-Array). Der Aufruf enthält später in jedem Falle als ersten Parameter vor den hier angegebenen den Namen der temporären Datei mit dem Programmcode.

  KeepFiles – »true« oder »1« verhindert, dass nach erfolgtem Programmaufruf das temporär angelegte Verzeichnis und dessen Inhalt gelöschte wird (optional, Standard: »false«, nur für Testzwecke).

Beispiel:

```
cat <<EOF | curl -T - -X PUT http://localhost:55555
{"Action":"/usr/bin/Rscript",
"Body":["a <- 1:10","b <- which(a > 2 & a < 8)","b"]}
EOF
```

Rückgabe:

```
{
  "t_start":1385566508241,
  "t_stop":1385566508399,
  "exitCode":0,
  "Result":"[1] 3 4 5 6 7\n"
}
```

- `"/bin/echo"`:

  Sendet den Aufrufparameter zurück (für Testzwecke). Weiterer Parameter:

  Value – Der Aufrufparameter für das Programm.

- `"/usr/bin/which"`:

  Testet, ob das als Aufrufparameter angegebene Programm existiert (für Testzwecke). Weiterer Parameter:

  Value – Der Aufrufparameter für das Programm.

1.2. Interne Aktionen

Es handelt sich um eine vom Server direkt ausführbare Aktion, bei der die Angabe des Namens eines externen Programms nicht notwendig ist bzw. es wird auf ein solches gänzlich verzichtet. Beginnt »`Action`« mit dem Zeichen »_«, so kennzeichnet sie einen internen Prozess zu administrativen Zwecken.

Action – kennzeichnet die zu erledigende Aufgabe.

- `"TEX"`:

  Bei dieser Aktion handelt es sich eigentlich auch um einen externen Programmauf. Aufgrund der Komplexität ist sie dennoch als »interne Aktionen« eingeordnet.

  Es wird aus dem übergebenen TEX-Code eine Datei erzeugt, die mit einem der unterstützten TEX-Compiler in PDF-Code (PDF-Stream) konvertiert wird.

  Body – TEX-Code in Form eines Strings oder eines Arrays von Strings (zwingend). Im zweiten Falle wird aus dem Array intern ein einzelner String erzeugt, wobei nach jedem Inhalt eines Array-Elements ein Zeilenumbruch (»\n«) eingefügt wird.

  Command – Der Name des TEX-Compilers (optional, Standard: »pdflatex«). Es werden nur TEX-Compiler unterstützt, die auf direktem Wege PDF-Dateien erzeugen, d. h. »pdflatex«, »lualatex«, »xelatex«, »pdftex«, »luatex«, »xetex«.

KeepFiles – »true« oder »1« verhindert, dass nach erfolgtem Programmaufruf das temporär angelegte Verzeichnis und dessen Inhalt gelöschte wird (optional, Standard: »false«, nur für Testzwecke).

- "TCP":
- "EMAIL":
- "RANDOM":
- "TIME":
- "_killRepeats":
- "_version":
- "_nodesVersion":
- "_environment":

2. Gitlab-Hook-Server – Port: 3420

Die Web-Applikation »GitLab« gestattet ähnlich zu dem Konkurrenzprodukt »GitHub« einen bequemen Zugriff auf das Versionskontrollsystem »GIT«. Zu jedem in »GitLab« registrierten Repositorium kann über »Settings«/»Web hooks« eine http-Adresse angegeben werden. Nach Änderung des betreffende Repositoriums (push-Aktion) wird an diese Adresse eine Datenstruktur gesendet. Sie enthält etliche Angaben zu dem Repositorium, wie Name des Repositoriums, URL für GIT-Aktionen, Nutzername, IDs der letzten Commits u. v. a. »nodejsServers2« bietet die Funktionalität eines Gitlab-Hook-Servers und ist somit in der Lage, diese Informationen von »GitLab« zu empfangen und auszuwerten. Die Adresse, die in »GitLab« angegeben werden muss, lautet im Falle des auf »a73434« laufenden »nodejsServers2« folgendermaßen:

```
http://a73434.berlin.ptb.de:3420
```

Zur Definition dessen, was nach Eintreffen der Informationen von »GitLab« zu tun ist, muss eine Datei `gitlabhook.conf` angelegt werden. Als Beispiel sei hier der Inhalt angeführt, der dazu führt, dass sich automatisch mit Änderung des Repositoriums »vaclabpage« html-Seiten, die vom Webserver ausgeliefert werden, erneuern:

```
{
  "tasks": {
    "vaclabpage": [
      "exec 1>/dev/null",
      "exec 2>/dev/null",
      "git clone %h",
      "cd %r",
      "cp -p --parents ‘git ls-files‘ /srv/www/htdocs/vaclabpage/"
    ]
  },
  "keep":false
}
```

Zur Erklärung: Unter »tasks« kann ein oder mehrere Namen von GIT-Repositorien aufgeführt werden. Jedem dieser Namen ist ein String oder ein String-Array zugeordnet. Darin enthalten sind Unix-Kommandozeilenaufrufe. »%h« ist ein Platzhalter für die zum

Clonen des GIT-Repositorium verwendbare URL. »%r« steht für den Namen des Repositoriums. Mit »"keep":false« wird festgelegt, dass temporär erstellte Verzeichnisse nicht erhalten bleiben sollen. Die Beschreibung zu dem NodeJS-Module »node-gitlab-hook« enthält weitere Hinweise zur Syntax in gitlabhook.conf.

Der konkrete Ablauf im temporären Verzeichnis des Rechners (/tmp) zur Erneuerung der Home-Page des Vakuumlabors ist also der folgende:

1. Lokales Duplikat des GIT-Repositoriums anlegen.
2. In das Verzeichnis mit dem Namen des Repositoriums wechseln.
3. Alle relevanten Dateien zum Webserver-Verzeichnis kopieren.

Es ist zu beachten, dass alle unter »nodejsServers2« laufenden Prozesse mit den Rechten des Nutzers »wwwrun« laufen.

3. Logging-Zugriff per Websocket-Protokoll – Port: 9001

Um detaillierte Informationen über den internen Ablauf beim Ansprechen der unter »nodejsServers2« laufenden Server-Prozesse zu erhalten, kann über den Port 9001 per Websocket-Protokoll Kontakt aufgenommen werden. Zu diesem Zweck steht das Kommandozeilen-Programm vlLogging zur Verfügung. Ohne Parameter nimmt es Kontakt zum lokal laufenden »nodejsServers2« auf. Wird als Parameter ein anderer Rechner (Rechnername oder IP-Adresse) angegeben, kann auch auf Informationen eines entfernten Rechners zugegriffen werden. Es wird das folgende Ausgabeformat angeboten:

```
2013-10-17 08:53:10.862 - LEVEL: [FILE_NAME:LINE_NUMBER:FUNCTION_NAME] MESSAGE
```

- LEVEL: »debug«, »error«, »warn« oder »info«
- FILE_NAME: Die Datei, in der sich der abzuarbeitende Code befindet
- LINE_NUMBER: Zeilennummer in FILE_NAME
- FUNCTION_NAME: Funktion, in der sich der abzuarbeitende Code befindet
- MESSAGE: Konkrete Informationen zum betreffenden Code-Teil.

Diese Informationen dienen der Fehlersuche im Programmcode von »nodejsServers2« und auch zur allgemeinen Beobachtung z. B. der Kommunikation mit Messgeräten. Künftig könnte der Websocket-Zugriff auch von einer Web-Applikation (Web-Browser) aus erfolgen.

Hinweise zu Verbesserungen dieses Dokuments bitte als Issue-Eintrag des git-Projektes »nodejsServers2« (siehe »GitLab«) oder per E-Mail an Rolf.Nieprashk@ptb.de.