DataAdapter für LineMetrics Cloud

Der DataAdapter ist ein Tool von LineMetrics, dass es Kunden erlaubt, Daten aus verschiedenen Quellen in die LineMetrics Cloud zu schicken.

Versionen

```
Version:           1.0.4
Datum:             26.03.2019
Verantwortlicher:  Klemens Engelbrechtsmüller <k.engelbrechtsmueller@linemetrics.com>
```

Funktionalität

- Verschiedene Datenquellen können zeitgleich ausgelesen werden
- Aus den Datenquellen können spezifische Werte ausgelesen werden
- Die Konfiguration erfolgt über ein lokales Konfigurationfile
- Die ausgelesenen Daten werden an das konfiguriere LineMetrics Konto gesendet
- Folgende Datenquellen werden unterstützt:
  - OPC UA
  - OPC DA
  - Modbus TCP
  - BACnet

Anforderungen

- JAVA JRE 8+
- Internetverbindung
- gültiges LineMetrics Konto mit konfigurierter sicherer API im LineMetrics Cloud Device Manager

Konfiguration

Die Konfiguration muss aktuell über das lokale Propertyfile `system.properties` angelegt werden. Dieses File setzt sich wie folgt zusammen: 1. LineMetrics API Zugangsdaten 2. Plugin, von welcher Quelle die Daten gelesen werden sollen 3. Items, welche Werte von der angegebenen Quelle sollen gelesen werden

API Zugangsdaten

```
linemetrics.clientid=asdf
linemetrics.clientsecret=asdf
```

Plugins

Die Datenquellen werden über Plugins realisiert und konfiguriert. Dazu muss im lokalen Konfigurationsfile zumindest ein Plugin angegeben werden.

```
dataadapter.<ID>.type=com.linemetrics.dataadapter.plugin.opcua.reader.OPCUaReaderClient
dataadapter.<ID>.host=opc.tcp://desktop-ceffcm3:51210/UA/SampleServer
dataadapter.<ID>.username=asdf
dataadapter.<ID>.password=asdf
```

Die Feld `<ID>` gibt die ID (beginnend bei 1) der Datenquelle an. Das Property `type` gibt das zu verwendende Plugin an. Im Moment werden folgende Plugins unterstützt:

- `opcua.reader.OPCUaReaderClient`: ermöglicht das intervallmäßige Lesen von Daten aus einem OPC (UA) Server
- `opcua.subscription.OPCUaSubscriptionClient`: ermöglicht das Lesen von Wertänderungen aus einem OPC (UA) Server
- `opc.OPCClient`: ermöglicht das intervallmäßige Lesen von Daten aus einem vorhandenen OPC DataAccess Server
- `opc.local.OPCClient`: ermöglicht das intervallmäßige Lesen von Daten aus einem lokalen OPC DataAccess Server
- `bacnet.reader.BacnetReaderClient`: ermöglicht das intervallmäßige Lesen von Daten aus einem Bacnet Device
- `bacnet.subscription.BacnetSubscriptionClient`: ermöglicht das Lesen von Wertänderungen aus einem Bacnet Device
- `modbus.ModbusClient`: ermöglicht das intervallmäßige Lesen von Daten aus einem Modbus Gerät

Neben den verpflichtenden Properties `type`, dass das gewünschte Plugin definiert, und `host`, dass den Endpunkt der Datenquelle beschreibt, müssen abhängig vom verwendeten Plugin verschiedene Properties für die Verbindung zur Datenquelle angegeben werden. Die zu verwendenden Properties können bei den vollständigen Beispielkonfigurationen nachgeschlagen werden.

Notiz zu `opc.local.OPCClient`: Wird dieses Plugin verwendet müssen folgende Voraussetzungen erfüllt werden - JRE 8 32Bit Version muss installiert sein - Im Pfad `C:\LM_Dataadapter\libs` muss die Datei `JCustomOpc.dll` existieren. Diese befindet sich im Repository - Der DataAdapter muss als Administrator ausgeführt werden

Items

Über die Items wird definiert welche Variablen bzw. Werte aus der Datenquelle gelesen werden sollen. Pro Datenquelle können beliebig viele Werte konfiguriert werden. Der Aufbau eines Items setzt sich wie folgt zusammen:

```
dataadapter.<ID>.item.<ITEM_ID>.namespace=2
dataadapter.<ID>.item.<ITEM_ID>.id=10848
dataadapter.<ID>.item.<ITEM_ID>.polltime=10000
dataadapter.<ID>.item.<ITEM_ID>.lm_alias=analogalias
dataadapter.<ID>.item.<ITEM_ID>.lm_customkey=mycustomkey
```

Die `<ITEM_ID>` gibt die ID des Items pro Datenquelle an (beginnend bei 1). Abhängig von der Datenquelle muss ein eindeutiger Identifier für den zu lesenden Wert angegeben werden. In diesem Fall (OPC UA) sind das die Felder `namespace` und `id`. Die Identifier für die verschiedenen Items unterscheiden sich teilweise pro Plugin. Wie die Items im jeweiligen Plugin konfiguriert werden müssen ist in der entsprechenden Beispielkonfiguration (siehe unten) veranschaulicht.

Pro Item müssen zusätzlich allgemeine Werte definiert werden, die für jedes Plugin gleich sind. Die `polltime` (in ms) gibt an in welchem Zeitintervall die Daten gelesen werden sollen bzw. in welchem Zeitintervall auf Änderungen eines Wertes gehorcht werden soll. Die Felder `lm_customkey` und `lm_alias` geben darüber Auskunft an welches Objekt in der LineMetrics Cloud die gelesenen Daten gesendet werden sollen.

Hinweis: Die `polltime` entspricht bei Verwendung des `OPCUaSubscriptionClient` dem Samplingintervalls des OPC UA Servers.

Sonstiges

Das Feld `activated_adapters` gibt an, welcher der angegebenen Datenquellen aktiv sein soll. Dieses Property kann auch mehrmals angegeben werden, in dem Fall das mehrere Datenquellen aktiv sein sollen:

```
activated_adapters=1
activated_adapters=2
```

Logging

Sämtliche Aktivitäten des DataAdapters werden in das File `system.log`, das sich im Verzeichnis des ausgeführten Programmes befindet, geloggt. Tritt ein Fehler auf, wird der Fehler zusätzlich in das File `error.log` geschrieben.

Installation

Um den DataAdapter zu installieren müssen Sie die aktuelle Version in einen beliebigen Ordner auf Ihrem Rechner kopieren. Folgende Dateien müssen vorhanden sein um das Programm ausführen zu können: - service.jar: Anwendung - start.cmd: Startscript - system.properties: Fertige Konfiguration

Ausführung

Ist der DataAdapter über `system.properties` vollständig konfiguriert kann das Programm gestartet werden. Soll das Programm auf einem Windows-Rechner ausgeführt werden, kann das Programm einfach über das Script `start.cmd`, welches sich im Installationspaket befindet, gestartet werden. Solange das Shellfenster nicht geschlossen wird, wird das Programm endlos ausgeführt.

Fehler beim Lesen oder Schreiben der Daten werden wie folgt behandelt: 1. Werden über die Konfiguration falsche LineMetrics Credentials angegeben, wird das Programm mit einer entsprechenden Fehlermeldung sofort wieder beendet. 2. Wird über die Konfiguration ein nicht vorhandenes oder fehlerhaftes Plugin konfiguriert, wird das Programm ebenfalls mit einer entsprechenden Fehlermeldung sofort wieder beendet. 3. Ist die LineMetric API vorübergehend nicht verfügbar, werden die gelesenen Daten zwischengespeichert und, sobald die API wieder verfügbar ist, an die API weitergeleitet. 4. Ist die angegebene Datenquelle (zB. OPC UA Server) vorübergehend nicht verfügbar, wird im regelmäßigem Intervall versucht erneut eine Verbindung herzustellen. Das Programm wird daher in diesem Fall nicht beendet. 5. Ist ein falsches Item konfiguriert, (zB. falscher LineMetrics Alias, nicht existierende ID in der Datenquelle) wird ein entsprechender Fehler in das File `error.log` geschrieben, das Programm jedoch nicht beendet.

Windows-Autostart

Soll der DataAdapter automatisiert beim Systemstart (von Windows) ausgeführt werden, müssen folgende Vorkehrungen getroffen werden:

1. Öffnen Sie den Menüpunkt: Windows Systemsteuerung > System und Sicherheit > Verwaltung > Aufgabenplanung
2. Erstellen Sie eine neue Aufgabe unter "Einfache Aufgabe erstellen"
3. Vergeben Sie einen Namen für ihren Task. z.B. LM DataAdapter
4. Wählen Sie aus wann der DataAdapter gestartet werden soll. z.B. Bei User Login
5. Im nächsten Schritt wählen Sie "Ein Programm starten"
6. Nun kann der Pfad zum Programm `start.cmd` im DataAdapter Ordner angegeben werden z.B. C:\Users\IEUser\Documents\LMDataAdapter\1.0.1start.cmd
7. Wichtig: Unter "Starten in" müssen Sie den Pfad zur `start.cmd` noch einmal angeben. z.B. C:\Users\IEUser\Documents\LMDataAdapter\1.0.1\
8. Nun kann das Anlegen abgeschlossen werden
9. Zusätzlich muss das Feld "Mit höchsten Berechtigungen ausführen" markiert werden

Wurde die Aufgabe erfolgreich erstellt, startet der DataAdapter beim nächsten Systemstart automatisch.

Für Entwickler

- Um das Projekt erfolgreich builden zu können muss die Bibliothek "JEasyOpc" im lokalen Repository vorhanden sein. Das Projekt kann von [https://github.com/autinitysystems/jeasyopc] geklont werden und muss danach über Maven gebuildet (mvn install) werden.
- Ebenso braucht man die Bibliothek "bacnet4j", diese kann von [https://github.com/empeeoh/BACnet4J] geklont werden
- Zusätzlich benötigt der DataAdapter das LineMetrics Java SDK. Dieses kann von [https://github.com/LineMetrics/linemetrics-sdk-java] geklont werden. Nach dem Maven Build sollte es erfolgreich importiert (mvn install) werden können.

TODO: Momentan wird eine abgeänderte Version von JEasyOpc verwendet, bei dem die JCustomOpc.dll Library mit einem absoluten Pfad geladen wird. Das wird noch geändert damit die Library direkt aus dem JAR File geladen wird.

Beispielkonfiguration

Vollständige Beispielkonfiguration für OPC UA Server, Intervall lesen

OPC UA spezifische Konfiguration: Wird das OPC UA Plugin verwendet, kann für den Adapter neben dem `type` und dem `host` optional auch `username` und `password` angegeben werden, falls der Server eine Authentifizierung erfordert. Wichtig bei den Items sind hier als Identifier die zwei Properties `namespace`, gibt den Namespace des OPC Objektes an, und `id`, gibt die ID des zu lesenden Objektes an.

```
#LineMetrics API
linemetrics.clientid=api_xxx
linemetrics.clientsecret=xxx

#Plugin
dataadapter.1.type=com.linemetrics.dataadapter.plugin.opcua.reader.OPCUaReaderClient
dataadapter.1.host= opc.tcp://desktop-ceffcm3:51210/UA/SampleServer
dataadapter.1.username=test
dataadapter.1.password=test

#Items to read - 1
dataadapter.1.item.1.namespace=2
dataadapter.1.item.1.id=10848
dataadapter.1.item.1.polltime=10000
dataadapter.1.item.1.lm_alias=analogalias
dataadapter.1.item.1.lm_customkey=mycustomkey
#Items to read - 2
dataadapter.1.item.2.namespace=2
dataadapter.1.item.2.id=10855
dataadapter.1.item.2.polltime=60000
dataadapter.1.item.2.lm_alias=textalias
dataadapter.1.item.2.lm_customkey=mycustomkey
#Activated adapters
activated_adapters=1
```

Vollständige Beispielkonfiguration für OPC UA Server, auf Änderungen horchen

OPC UA spezifische Konfiguration: Wird das OPC UA Plugin verwendet, kann für den Adapter neben dem `type` und dem `host` optional auch `username` und `password` angegeben werden, falls der Server eine Authentifizierung erfordert. Wichtig bei den Items sind hier als Identifier die zwei Properties `namespace`, gibt den Namespace des OPC Objektes an, und `id`, gibt die ID des zu

lesenden Objektes an.

```
#LineMetrics API
linemetrics.clientid=api_xxx
linemetrics.clientsecret=xxx

#Plugin
dataadapter.1.type=com.linemetrics.dataadapter.plugin.opcua.subscription.OPCUaSubscriptionCl
dataadapter.1.host= opc.tcp://desktop-ceffcm3:51210/UA/SampleServer
dataadapter.1.username=test
dataadapter.1.password=test

#Items to read
dataadapter.1.item.1.namespace=2
dataadapter.1.item.1.id=10848
dataadapter.1.item.1.polltime=10000
dataadapter.1.item.1.lm_alias=analogalias
dataadapter.1.item.1.lm_customkey=mycustomkey

#Activated adapters
activated_adapters=1
```

Vollständige Beispielkonfiguration für Remote OPC Standard Server, Intervall lesen

OPC Standard spezifische Konfiguration: Wird das OPC Plugin für einen Standard (DataAccess) Server verwendet, muss neben dem `type` und dem `host` auch die CLS-ID des OPC Servers angegeben werden. Zusätzlich muss `username` und `password` des Windowsbenutzers, der die benötigten Rechte für den OPC Server hat, angegeben werden. Bei den Items muss als Identifier die ID des OPC Tags unter `id` angegeben werden.

```
#LineMetrics API
linemetrics.clientid=api_xxx
linemetrics.clientsecret=xxx

#Plugin
dataadapter.1.type=com.linemetrics.dataadapter.plugin.opc.OPCClient
dataadapter.1.host=pcname
dataadapter.1.clsid=zzzzzzzzz-aaaa-bbbb-yyyy-xxxxxxxx
dataadapter.1.username=user
dataadapter.1.password=pw

#Items to read
dataadapter.1.item.1.id=Random.Real8
dataadapter.1.item.1.polltime=60000
dataadapter.1.item.1.lm_alias=analogalias
dataadapter.1.item.1.lm_customkey=mycustomkey

activated_adapters=1
```

Vollständige Beispielkonfiguration für Lokalen OPC Standard Server, Intervall lesen

Wird das OPC Plugin für einen lokalen OPC Server verwendet, muss neben dem `type` und dem `host` auch die `progid` des OPC Servers angegeben werden. Bei den Items muss als Identifier die ID des OPC Tags unter `id` angegeben werden.

```
#LineMetrics API
linemetrics.clientid=api_xxx
linemetrics.clientsecret=xxx

#Plugin
dataadapter.1.type=com.linemetrics.dataadapter.plugin.opc.local.OPCClient
```

```
dataadapter.1.host=localhost
#ProgID des OPC Servers
dataadapter.1.progid=progid

#Items to read
dataadapter.1.item.1.id=Random.Real8
dataadapter.1.item.1.polltime=60000
dataadapter.1.item.1.lm_alias=analogalias
dataadapter.1.item.1.lm_customkey=mycustomkey

activated_adapters=1
```

Vollständige Beispielkonfiguration für BACNet, Intervall lesen

Wird das BACNet Plugin verwendet, muss zum Einen die Konfiguration für das lokale Bacnet-Device und zum Anderen die Konfiguration für das Remote-Device angegeben werden. `remotedevice_host` gibt die IP-Adresse des gewünschten BACNet Gerätes an. Bei `remotedevice_port` wird der Port angegeben. `remotedevice_id` gibt die Geräte-ID an, und über das Feld `remotedevice_networknumber` kann, wenn nötig, die Netzwerknummer angegeben werden. Für das lokale Geräte muss folgendes angegeben werden: `localdevice_broadcast` für das anzusprechende Subnet. `localdevice_bindaddress` für die zu bindende IP-Adresse (kann auch 0.0.0.0 verwendet werden), auch hier unter `localdevice_port` der zu verwendende Port, die Geräte-ID (`localdevice_id`) und die Netzwerknummer. Bei dem Items wird als eindeutiger Identifier die `object_instance` , `object_id` und die `property_id` verlangt.

```
#LineMetrics API
linemetrics.clientid=api_xxx
linemetrics.clientsecret=xxx

#Plugin
dataadapter.1.type=com.linemetrics.dataadapter.plugin.bacnet.reader.BacnetReaderClient
dataadapter.1.remotedevice_host=192.168.1.1
dataadapter.1.remotedevice_id=11111
dataadapter.1.remotedevice_port=47808
dataadapter.1.remotedevice_networknumber=0
dataadapter.1.localdevice_broadcast=255.255.255.0
dataadapter.1.localdevice_bindaddress=192.168.1.2
dataadapter.1.localdevice_networknumber=0
dataadapter.1.localdevice_port=47809
dataadapter.1.localdevice_id=10002

#Items to read
dataadapter.1.item.1.object_instance=0
dataadapter.1.item.1.object_id=1
dataadapter.1.item.1.property_id=125
dataadapter.1.item.1.polltime=5000
dataadapter.1.item.1.lm_alias=analogalias
dataadapter.1.item.1.lm_customkey=mycustomkey

activated_adapters=1
```

Vollständige Beispielkonfiguration für BACNet, auf Änderungen horchen

Wird das BACNet Plugin verwendet, muss zum Einen die Konfiguration für das lokale Bacnet-Device und zum Anderen die Konfiguration für das Remote-Device angegeben werden. `remotedevice_host` gibt die IP-Adresse des gewünschten BACNet Gerätes an. Bei `remotedevice_port` wird der Port angegeben. `remotedevice_id` gibt die Geräte-ID an, und über das Feld `remotedevice_networknumber` kann, wenn nötig, die Netzwerknummer angegeben werden. Für das lokale Geräte muss folgendes angegeben werden: `localdevice_broadcast` für das anzusprechende Subnet. `localdevice_bindaddress` für die zu bindende IP-Adresse (kann auch 0.0.0.0 verwendet werden), auch hier unter `localdevice_port` der zu verwendende Port, die

Geräte-ID (`localdevice_id`) und die Netzwerknummer. Bei dem Items wird als eindeutiger Identifier die `object_instance` , `object_id` und die `property_id` verlangt.

```
#LineMetrics API
linemetrics.clientid=api_xxx
linemetrics.clientsecret=xxx

#Plugin
dataadapter.1.type=com.linemetrics.dataadapter.plugin.bacnet.subscription.BacnetSubscription
dataadapter.1.remotedevice_host=192.168.1.1
dataadapter.1.remotedevice_id=11111
dataadapter.1.remotedevice_port=47808
dataadapter.1.remotedevice_networknumber=0
dataadapter.1.localdevice_broadcast=255.255.255.0
dataadapter.1.localdevice_bindaddress=192.168.1.2
dataadapter.1.localdevice_networknumber=0
dataadapter.1.localdevice_port=47809
dataadapter.1.localdevice_id=10002


#Items to read
dataadapter.1.item.1.object_instance=0
dataadapter.1.item.1.object_id=1
dataadapter.1.item.1.property_id=125
dataadapter.1.item.1.polltime=5000
dataadapter.1.item.1.lm_alias=analogalias
dataadapter.1.item.1.lm_customkey=mycustomkey

activated_adapters=1
```

Vollständige Beispielkonfiguration für einen Modbus Client, Intervall lesen

Wird das Modbus TCP Plugin verwendet, muss neben dem `host` auf dem sich das Modbus Gerät befindet auch der `port` angegeben werden. Bei den Items muss als Identifier das Property `id` angegeben werden. Dieser Wert kennzeichnet den Register der gelesen werden soll.

```
#LineMetrics API
linemetrics.clientid=api_xxx
linemetrics.clientsecret=xxx

#Plugin
dataadapter.1.type=com.linemetrics.dataadapter.plugin.modbus.ModbusClient
dataadapter.1.host=192.168.0.0
dataadapter.1.port=502

#Items to read
#ID entspricht dem zu lesenden Register
dataadapter.1.item.1.id=286
dataadapter.1.item.1.polltime=5000
dataadapter.1.item.1.lm_alias=analogalias
dataadapter.1.item.1.lm_customkey=mycustomkey

activated_adapters=1
```