Einrichtung DeltaLoad

Programm „DeltaLoad“
Referenz-Implementation zum Bezug von
Delta-Updates der *Markdaten Endkundentarife*

Hintergrund

Seit dem 1. Januar 2021 steht Ihnen im Bereich Endkundentarife ausschließlich das neue Datenbankformat zur Verfügung, welches nach Implementierung des entsprechenden *Webservice Marktdaten* mit Delta-Aktualisierungen für Sie stets aktuell gehalten wird.

Die Delta-Lieferungen beziehen Sie wahlweise mittels der hier beschriebenen Referenz-Implementation „DeltaLoad“, oder mittels eines eigenen Client-Programms zum Abrufen über unseren REST-Webservice und anschließendem Einspielen in Ihre Ziel-Datenbank.

Im Download-Bereich | ene't Navigator® (enet-navigator.de) finden Sie die aktuellste Fassung des Programms „DeltaLoad“ (Linux- und Windows-Version), welches das Abrufen und transaktionssichere Einspielen der Delta-Updates in die Zieldatenbank übernimmt.

Disclaimer

Für die in diesem Dokument beschriebenen Vorgänge und angehangenen Skripte und Programme übernimmt die ene't GmbH keinerlei Haftung für verursachte Schäden an Ihrer Hard-, Software oder anderweitigen Daten. Die genannten Produkte und Drittanbieteranwendungen sind Eigentum der jeweiligen Herausgeber. Für die Installation der hier beschriebenen Umgebung sind unter Umständen zusätzliche, von Kund*innen zu beschaffende, Lizenzen notwendig.

Einrichtung DeltaLoad

Inhalt

Einrichtung DeltaLoad

Funktion von DeltaLoad

DeltaLoad ist ein in C# / .NET Core 3.1 (LTS) geschriebenes Programm der ene't GmbH, das jede*r Kund*in einsetzen kann, um seine Datenbank-Aktualisierungen in Form von Delta-Paketen über einen REST-Webservice abzurufen und damit seine Ziel-Datenbank transaktionssicher zu aktualisieren. Dies garantiert den Anwendern, immer mit den neuesten Marktdaten zu arbeiten.

Wichtiger Hinweis:

Das Programm DeltaLoad ist nur ein einzelner Baustein, um regelmäßige Delta-Updates für das neue Format *Marktdaten Endkundentarife* zu beziehen. **Ohne vorherige Anmeldung für das neue Format der *Marktdaten Endkundentarife* hat das Programm für den*die Kund*in keinen Nutzen.** Beachten Sie bitte auch unsere separaten Dokumentationen für die initiale Einrichtung von MSSQL- oder PostgreSQL-Datenbanken, die wir ebenfalls im Download-Bereich | ene't Navigator® (enet-navigator.de) bereitstellen.

Anmeldung für das neue Format *Marktdaten Endkundentarife*

Bitte wenden Sie sich an Ihre*n Kundenberater*in aus dem Vertrieb, um ein Abonnement der *Marktdaten Endkundentarife* zu beziehen. Als registrierte*r Kund*in der ene't GmbH verfügen Sie über einen eigenen Zugang für den Download-Bereich | ene't Navigator®. Die für die Inbetriebnahme des Prozesses benötigten Daten finden Sie nach dem Login unter „Datenbanken“ bzw. „Tools“.

Im Bereich „Datenbanken“ können Sie ein größeres ZIP-Paket unter der Bezeichnung „**Marktdaten MSSQL**“ bzw. "**Marktdaten PostgreSQL**" herunterladen. In dieser ZIP-Datei ist das aktuelle (initiale) Backup Ihrer neuen Ziel- bzw. Initialdatenbank enthalten sowie zur Einrichtung benötigte Skripte und eine Version von DeltaLoad.

Zur Aktualisierung dieser Datenbank stehen Ihnen ab sofort Webservices zur Verfügung. Über die Maschinenkontenverwaltung | ene't Navigator® können Sie sich ein Maschinenkonto für die Webservices *WS Marktdaten Strom* und *WS Marktdaten Gas* anlegen, indem Sie auf „Maschinenkonto hinzufügen+“ klicken. Bitte beachten Sie, dass die Zugangsdaten nur einmalig angezeigt werden. Das Maschinenkonto berechtigt Sie zur Anwendung der Delta-Updates auf die Initialdatenbank. Die Zugangsdaten müssen in die Konfiguration von DeltaLoad eingetragen werden. **Ohne gültiges Abonnement des neuen Formats der *Marktdaten Endkundentarife* und des Maschinenkontos hat das Programm DeltaLoad für Anwender keinen Nutzen.**

Im Gegensatz zur Vorgehensweise bei Formaten zu Netznutzungsentgelten erhalten Sie mit dem neuen Format „Marktdaten Endkundentarife“ einmalig eine vorbefüllte Initial-Datenbank, die dann durch Delta-Updates bei Ihnen fortwährend aktualisiert werden kann. Dies reduziert die transferierte Datenmenge deutlich gegenüber dem wiederholten Herunterladen des vollständig befüllten Datenbankformats, wie es bisher notwendig war. Technisch gesehen bestehen die Deltas allerdings nicht aus reinen Daten, sondern aus den notwendigen SQL-Befehlen, die letztlich Ihre Ziel-Datenbank aktualisieren. Hierdurch wird es uns auch erst möglich, unsere regelmäßig vorab angekündigten Strukturänderungen mit über Deltas auszurollen.

Einrichtung DeltaLoad

Ersteinrichtung in der Kundenumgebung

1. Einrichtung DeltaLoad

Um die Kundenumgebung zum kontinuierlichen Abruf der Delta-Pakete vom *Webservice Marktdaten* und die Anwendung der Deltas auf die Zieldatenbank einzurichten, entpacken Sie bitte die im Download-ZIP enthaltene Datei **`DeltaLoad_x.yy.zzzz.zip`**.

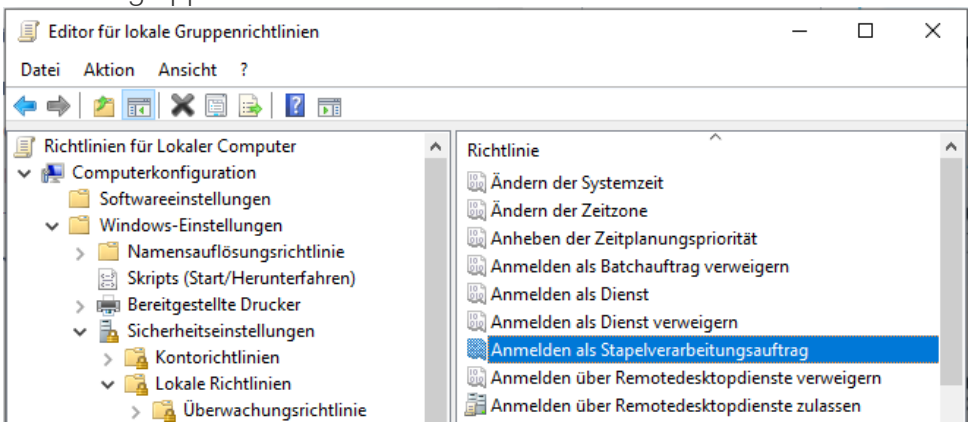
Linux

1. Legen Sie ein eigenes Konto an, in dessen Kontext das DeltaLoad ausgeführt wird. Diesem Konto geben Sie die Schreibrechte auf das Verzeichnis, in dem DeltaLoad ausgeführt werden soll (z. B. **`/home/MeinKonto/deltaload`** od. **`/opt/deltaload`**).
2. Kopieren Sie den Inhalt des Unterverzeichnisses **`linux-x64`** in das von Ihnen gewünschte Verzeichnis (z. B. **`/home/MeinKonto/deltaload`** od. **`/opt/deltaload`**).
3. Gehen Sie in das Verzeichnis und machen Sie die folgende Datei ausführbar:
   `chmod 755 DeltaLoad`
4. Stellen Sie bei Verwendung eines Verzeichnisses außerhalb des Home-Verzeichnis („~“) ggf. noch sicher, dass ausreichende Berechtigungen auf dem Verzeichnis sind (das DeltaLoad erzeugt in diesem Verzeichnis auch seine Logdateien), z. B.:
   `chmod -R 755 /opt/deltaload`

Windows

1. Kopieren Sie den Inhalt des Unterverzeichnisses **`win-x64`** in das von Ihnen gewünschte Verzeichnis (z. B. **`C:\DeltaLoad`**).

2. Legen Sie unter Windows einen Benutzer an mit dem Benutzer-Recht „Anmelden als Stapelverarbeitungsauftrag“ (über lokales „gpedit.msc“ oder vorzugsweise Domänengruppenrichtline auf Computerebene):



   Des Weiteren benötigt der Benutzer Schreib- und Leserechte auf das Verzeichnis, in dem Sie die DeltaLoad.exe legen.
3. Richten Sie im Windows Aufgabenplaner einen regelmäßig startenden Auftrag (z. B. alle 10 Minuten) zum Starten des DeltaLoad.exe an, ausgeführt unter diesem Benutzer, auch während er nicht angemeldet ist. Im Register „Einstellungen“ der Aufgabe stellen Sie bitte unter „Folgende Regel anwenden, falls die Aufgabe bereits ausgeführt wird“ die Einstellung „Keine neue Instanz starten“ sicher. Mittels des WorkMode `"service"` in der zugehörigen Konfigurationsdatei "appsettings.json" können Sie zusätzlich dafür sorgen, dass das DeltaLoad endlos Deltas verarbeitet. Das regelmäßige Neustarten des Auftrags (z.B. alle 10 Minuten, s.o.) führt somit nur dann zu einer

neuen DeltaLoad-Prozessinstanz, wenn es in der vorherigen Instanz eine Unterbrechung mit Abbruch der Verarbeitung gab.

Hinweis: Alternativ zur Einrichtung als Windows-Aufgabe können Sie zu einem späteren Zeitpunkt auch DeltaLoad.exe als echten Windows-Service gemäß Kapitel 3 installieren, sobald DeltaLoad.exe dies unterstützt. Bitte beachten Sie hierzu im Anhang die Versionshistorie des DeltaLoad.

2. Konfiguration DeltaLoad

In beiden Fällen (Linux und Windows) fahren Sie wie folgt im selben Verzeichnis fort. Wechseln Sie in das obige Verzeichnis und bearbeiten Sie die Datei **appSettings.json** mit einem Text-Editor.

Passen Sie mindestens diese Einträge in der **appSettings.json** an:

```
"ConfCrypted":     "false"
"mTokenPrefix":    "<Ihr Maschinenkonto-Benutzer-Kuerzel>"
"mTokenMain":      "<Ihr Maschinenkonto-Passwort>"
"DbServer":        "<Ihr Datenbank-Server FQDN oder IP>"
"DbPassword":      "<Ihr Passwort für den DB-Benutzer mit Schreibrechten>"
```

Optional ist noch der Datenbank-Benutzer über den Parameter `"DbUserName"` anzupassen, falls dieser abweicht.

*Wichtiger Hinweis 1: Bei Änderungen an einem der beiden Parameter **"mTokenMain"** oder **"DbPassword"** sind immer beide gleichzeitig einzutragen und ebenfalls der Parameter **"ConfCrypted"** wieder auf **"false"** zu setzen. Details hierzu finden Sie in der Versionshistorie im Anhang unter dem Punkt „Automatische Verschlüsselung sensibler Konfigurationsdaten“.*

Wichtiger Hinweis 2: Der für DeltaLoad verwendete Datenbankbenutzer muss die Datenbankberechtigungen haben, dort DDL-SQL-Befehle (Data Definition Language) auszuführen, damit auch unsere regelmäßig vorab angekündigten Strukturänderungen mit über Deltas in Ihre Ziel-Datenbank einfließen können.

Dem Eintrag `WorkMode` können Sie folgende Werte geben:

- `"service"`: wenn DeltaLoad unter Windows permanent laufen soll
  (Alternativ den Aufruf-Parameter `"-service"`)
- `"daemon"`: wenn DeltaLoad unter Linux permanent laufen soll
  (Alternativ den Aufruf-Parameter `"-daemon"`)
- `"singledelta"` oder leer (und keinen der Aufruf-Parameter `"-daemon"` oder `"-service"`): wenn DeltaLoad nur einmalig laufen und sich danach beenden soll.

Mit der Einstellung des WorkMode legen Sie somit die Arbeitsweise von DeltaLoad fest. Für einen Produktivbetrieb empfehlen wir, die Einstellung `"service"` zu verwenden. Hier wird in einem zyklischen Ablauf auf ein Delta-Update gewartet, dieses wird verarbeitet, dann wird auf das nächste Delta-Update gewartet, usw.

Einrichtung DeltaLoad

Abweichende Konfiguration für Datenbankziel MSSQL

Fall 1: DeltaLoad und der MSSQL-Server laufen auf Windows unter demselben Domänenbenutzer-Konto mit Berechtigungen zur Datenbank → In der Konfigurationsdatei werden kein Anmeldename und kein Kennwort benötigt:

```
"ConfCrypted":  "false"
[..]
"mTokenPrefix": "<Ihr Maschinenkonto-Benutzer-Kuerzel>"
"mTokenMain":   "<Ihr Maschinenkonto-Passwort>"
[..]
"DbServer":     "<Ihr Datenbank-Server FQDN oder IP>"
"DbUserName":   ""
"DbPassword":   ""
```

Fall 2: DeltaLoad oder MSSQL-Server laufen NICHT unter einem Domänenbenutzer-Konto. In diesem Fall legen Sie einen dedizierten Benutzer mit Schreibrechten in der Datenbank an (z. B. **updateuser**) und tragen diesen in der Konfigurationsdatei für den Zugriff zur Datenbank ein:

```
"ConfCrypted":  "false"
[..]
"mTokenMain":   "<Ihr Maschinenkonto-Passwort>"
[..]
"DbUserName":   "updateuser"
"DbPassword":   "<Passwort des updateuser>"
```

3. Installation von DeltaLoad als Windows Dienst (zukünftiges Feature)

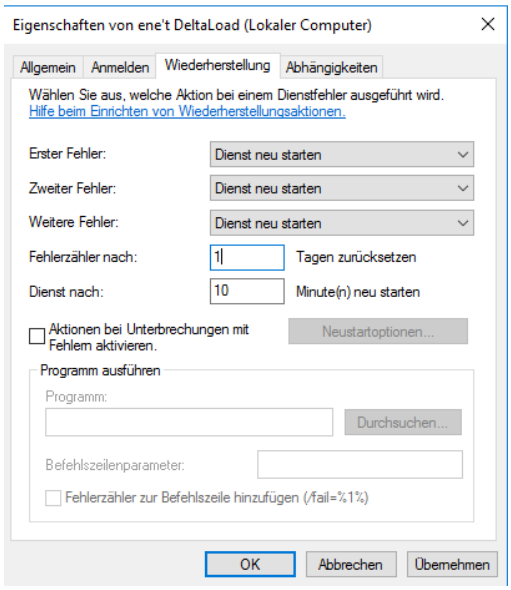
*Wichtiger Hinweis: In der vorliegenden Version des DeltaLoad wird die Verwendung als Windows-Dienst noch **nicht** unterstützt. **Die nachfolgende Beschreibung gilt daher nur für zukünftige Versionen des DeltaLoad, die dieses Feature unterstützen. Bitte beachten Sie hierzu die Versionshistorie des DeltaLoad.***

Zusammen mit dem DeltaLoad haben Sie ein Powershell-Skript namens **InstallDeltaLoadAsWindowsService.ps1** in das Verzeichnis kopiert, wo sich bereits das Programm **DeltaLoad.exe** und die Konfigurationsdatei **appSettings.json** befinden.

Zur Einrichten von DeltaLoad als Windows-Service starten Sie eine Powershell-Konsole als Administrator, wechseln in das Verzeichnis und rufen dort das o. g. Powershell-Skript auf:

```
PS C:\enet> .\InstallDeltaLoadAsWindowsService.ps1
```

Folgen Sie den weiteren Anweisungen im Skript-Verlauf. Das Skript fordert selbstständig erweiterte Rechte an und startet sich dabei neu, sofern Sie es nicht schon als Administrator gestartet haben. Nach dem Durchlauf des Installationsskripts finden Sie das DeltaLoad als Windows-Dienst unter dem Dienste-Anzeigenamen „ene't DeltaLoad“ mit entsprechendem Beschreibungstext und Starttyp „Automatisch“ wieder und können den Dienst entsprechend starten. Ein mehrmaliges Ausführen des Skripts ist unschädlich. Passen Sie die weiteren Einstellungen des Dienstes (z.B. Anmeldung unter einem Domänenbenutzer [empfohlen] oder wiederholter Wiederanlauf des Dienstes im Abbruchs-/Fehlerfall) ggf. entsprechend an:



Hinweis: *Bei der Registrierung als Windows-Dienst wird automatisch der Aufrufparameter* `"-service"` *mit angehängt. Dieser hat immer Vorrang, also auch, wenn in der* **`appsettings.json`** *als* `WorkMode "singledelta"` *eingetragen wurde.*

4. Installation von DeltaLoad als Linux Daemon

Systemd ist mittlerweile seit Jahren das standardmäßige „init“ System in RHEL/CentOS, Fedora, Ubuntu, Debian und anderen Distributionen. Ein Eintrag über `init.d` ist damit nicht mehr nötig.

Stattdessen muss hier lediglich dem Systemd eine „service unit“ Datei bekannt gemacht werden, die eine bestimmte Definition aufweist, um ein Programm als Daemon-/Service-Prozess auszuweisen.

Legen Sie zunächst einen passenden Linux User an (z.B. "**deltauser**"), unter dem das DeltaLoad laufen soll. Legen Sie danach in dessen Home-Verzeichnis ein Verzeichnis für das DeltaLoad Programm an.

Der vorgeschlagene Pfad unter Linux ist `/home/deltauser//deltaload` und wird im Folgenden **exemplarisch** verwendet. Dort werden beispielsweise alle zum DeltaLoad gehörigen Dateien hinkopiert.

Hinweis: Bitte ersetzen dieses Verzeichnis in den unten genannten beiden Dateien und den nachfolgenden Ausführungen, wenn Sie ein anderes Verzeichnis verwenden und berücksichtigen Sie Schreib- und Leserechte auf das Verzeichnis. Dort werden vom DeltaLoad u.a. Logdateien und temporäre Dateien während der Programmausführung geschrieben.

In diesem Verzeichnis sind dann zwei Dateien für die Einrichtung des DeltaLoad als Daemon unter Linux relevant:

- `deltaload.service` – service unit Datei für das DeltaLoad
- `install_deltaload_daemon.sh` – Bash-Skript zum Bekanntmachen des DeltaLoad als Daemon

Man wechselt als „root“ in das Verzeichnis `/home/deltauser/deltaload` und führt das Bash-Skript aus:

```
cd /home/deltauser/deltaload
```

Einrichtung DeltaLoad

```
./install_deltaload_daemon.sh
```

Eine erfolgreiche Registrierung als Daemon sieht dann im Skriptverlauf so aus:

```
Created symlink from /etc/systemd/system/deltaload.service to /home/deltauser/deltaload/deltaload.service.
DeltaLoad is now available as a daemon service unit 'deltaload':
deltaload.service linked

Please, assure the right configuration parameters in 'appsettings.json'
before you start the deltaload daemon service.

The daemon service can be started via command 'systemctl start deltaload'
or enabled for automatic start on boot via 'systemctl enable deltaload'.
```

Entsprechend kann nun der Daemon mittels **`systemctl start deltaload`** angestartet oder für das dauerhafte Mitstarten beim Booten mittels **`systemctl enable deltaload`** aktiviert werden.

Programmstart DeltaLoad

Hinweis: Bitte beachten Sie, dass die Synchronisationssoftware nur in einer einzigen Instanz gleichzeitig laufen sollte. So ist sichergestellt, dass alle Datenbank-Updates in gleicher Reihenfolge abgearbeitet werden. Seit Version 1.53 können zwar theoretisch mehrere DeltaLoad-Instanzen ohne Gefahr von Inkonsistenzen das gleiche Datenbankziel bedienen. Praktisch sollte man dies allerdings vermeiden, da mehrere Instanzen sich in der Regel gegenseitig bei der Delta-Verarbeitung unnötig ausbremsen.

Um lediglich die aktuelle Versionsnummer inkl. Build-Nummer des DeltaLoad auszugeben, wechseln Sie in das entsprechende Verzeichnis und führen es mit dem Kommandozeilen-Parameter "`--version`" aus:

- Unter Linux: **`./DeltaLoad --version`**
- Unter Windows: **`DeltaLoad.exe --version`**

Um die Funktion des DeltaLoad interaktiv zu testen – ohne es als Windows Aufgabenplan/Windows-Dienst/Linux Daemon zu starten – können Sie in das Verzeichnis des DeltaLoad wechseln (dies ist gleichzeitig auch das Arbeitsverzeichnis des DeltaLoad) und das Programm wie folgt einzeln aufrufen:

- Unter Linux: **`./DeltaLoad`**
- Unter Windows: **`DeltaLoad.exe`**

Ob das DeltaLoad dabei nur ein Delta verarbeitet und sich beendet oder endlos Deltas verarbeitet, hängt vom eingestellten Workmode ab.

Einrichtung DeltaLoad

Fehlt der Workmode `"service"` und `"daemon"` sowohl in der json-Config als auch als Kommandozeilen-Parameter oder wurde `"singledelta"` in der appsettings.json dort eingetragen, so verarbeitet das DeltaLoad-Programm nur ein einzelnes Delta-Update-Paket und beendet sich danach eigenständig (Exitcode 0 = OK). Dies können Sie bei dem interaktiven Startmodus sowohl in der Konsolenausgabe als auch in der erzeugten Logdatei des DeltaLoad überprüfen. Für die richtigen Einstellungen bzgl. des Loggings von DeltaLoad beachten Sie bitte den in der Versionshistorie zu Version 1.53 im Anhang genannten Abschnitt „**Umfangreiches, differenziertes Logging**".

Hinweis 1: Die Zeitspanne zwischen dem Einrichten der Initial-Datenbank (s. o.) und der Anbindung von Delta-Updates (Programm DeltaLoad oder der kundeneigene Abruf der Delta-Updates vom Webservice Marktdaten sowie dessen transaktionssicheres Verarbeiten in der Datenbank) darf eine Kalenderwoche nicht überschreiten. So ist sichergestellt, dass die noch ausstehenden Delta-Updates zeitnah aufgeholt werden können.

Hinweis 2: Alternativ zum DeltaLoad können Sie auch selbst den Webservice Marktdaten abrufen und so eigenständig für das transaktionssichere Einspielen der Deltas in Ihre Ziel-Datenbank sorgen. Die Autorisierung erfolgt über das o. g. jeweilige Maschinenkonto je Delta-Zieldatenbank.

Sollte es dennoch notwendig sein, den Restore mit einer Initial-Datenbank zu wiederholen (z.B. bei zu langer Unterbrechung der DeltaLoad-Anbindung an den Webservice, der zu alte Deltas nicht mehr vorhält), so ist bei jedem Restore ein neues Maschinenkonto für dieses neue Ziel zu erzeugen und das neue Maschinenkonto in der DeltaLoad-Konfigurationsdatei **`appsettings.json`** *(für die Neuanbindung an den Webservice Marktdaten) zu hinterlegen.*

Aktualisierung von DeltaLoad

In der initialen Bereitstellung im Bereich „Datenbanken" des Download-Bereich | ene't Navigator® (enet-navigator.de) ist eine Version von DeltaLoad bereits enthalten. **Die vollständige Beschreibung zur Einrichtung des neuen Formats inklusive des DeltaLoad-Programms finden Sie nach Registrierung für das neue Format ebenfalls im Download-Bereich unter „Tools".**

Hier gehen wir im Folgenden davon aus, dass Sie DeltaLoad bereits einmal vollständig eingerichtet haben. Daher fällt die Aktualisierung mit der bereitgestellten, neueren Version von DeltaLoad sehr einfach aus:

1a. Entpacken Sie die vorliegende Datei **`DeltaLoad_x.yy.zzzz.zip`**.

2a. Beenden Sie eine ggf. laufende Instanz des Programms DeltaLoad (Windows-Dienst „ene't DeltaLoad" bzw. Aufträge/Batches/Cronjob/Daemon mit DeltaLoad stoppen).

Unter Windows

3a. Kopieren Sie den Inhalt des Unterverzeichnisses **`win-x64`** **ohne** die Datei **`appsettings.json`** (da Sie hieran im Zielsystem bereits Anpassungen vorgenommen hatten) über Ihre vorhandene Installation von DeltaLoad (z. B. **`C:\DeltaLoad`**).

Unter Linux

3b.

i. Kopieren Sie den Inhalt des Unterverzeichnisses **`linux-x64`** **ohne** die Datei **`appsettings.json`** (da Sie hieran im Zielsystem bereits Anpassungen vorgenommen

hatten) über Ihre vorhandene Installation von DeltaLoad (z. B. **`/home/deltauser/deltaload/DeltaLoad`**)

ii. Gehen Sie in das Verzeichnis und machen Sie die folgende Datei ausführbar:
`chmod 755 DeltaLoad`

4. Kontrollieren Sie im Zielverzeichnis noch einmal Ihre Einstellungen in **`appsettings.json`** und passen Sie diese ggf. noch einmal an Ihre Bedürfnisse an.

5. Starten Sie nun wieder DeltaLoad (Windows Service/Auftrag/Cronjob/Batch/Daemon o. ä.).

Support

Kontaktieren Sie uns bitte, wenn Sie Fragen haben oder Probleme auftreten:

E-Mail: support@enet.eu
Telefon: 02433 52601-909

Einrichtung DeltaLoad

Anhang

Details zum Delta-Prozess und DeltaLoad

Bei DeltaLoad handelt es sich um ein Programm zur Ausführung in der Kommandozeile, als Windows-Service oder als Linux-Daemon. Damit rufen Sie komprimierte Delta-Updates/-Dateien der ene't *Marktdaten Endkundentarife* mit darin enthaltenen Datenbank-Update-Befehlen von einem ene't REST-Webservice ab. Diese spielen Sie dann transaktionssicher in eine MSSQL-/PostgreSQL-Datenbank des*der Kund*in ein.

Die Delta-Updates enthalten die notwendigen SQL-Statements im SQL92-Format, um die Zieldatenbank auf dem aktuellen Stand zu halten. Hierdurch ist es beispielsweise möglich, mit denselben SQL-Statements sowohl PostgreSQL als auch Microsoft SQL-Server als Zielsystem zu verwenden. Zurzeit unterstützt ene't bzgl. Initial-Datenbanken und über das DeltaLoad-Programm folgende Ziel-Datenbanksysteme:

- PostgreSQL
- Microsoft SQL Server

Für den Vorgang der Aktualisierung über Deltas können Sie die von der ene't GmbH speziell für diesen Zweck entwickelte Referenz-Implementierung – das hierin beschriebene Programm DeltaLoad – verwenden. Dieses Programm kann endlos wie ein Windows-Dienst (z. B. `DeltaLoad.exe --service`), als Linux-Daemon (z. B. `./DeltaLoad --daemon`) oder als einfaches Konsole-Programm (Verarbeitung eines einzelnen Delta-Updates mit anschließender Beendigung des Programms) in die Kundenumgebung eingebunden werden.

Das Programm DeltaLoad ist in C# unter .NET Core 3.1 (LTS) in Visual Studio 2019 entwickelt worden. Die ausführbaren Binaries sowie notwendige .NET-Core-Bibliotheken für die Lauffähigkeit unter Linux sind im ZIP inkludiert. Der Quellcode wird Kund*innen auf Anfrage bereitgestellt bzw. zu gegebener Zeit Open Source gestellt.

In unregelmäßigen Abständen wird ene't das Programm DeltaLoad überarbeiten oder erweitern und als eigenständigen Download bereitstellen.

Einrichtung DeltaLoad

Wichtige Details zu neuen Funktionen (ab Version 1.53)

Umfangreiches, differenziertes Logging:
Mittels folgendem Block in der DeltaLoad-Konfigurationsdatei **`appsettings.json`** können Sie nun genau bestimmen, welche Art von Informationen während des Programmlaufs von DeltaLoad auf Console, ins Logfile oder in das Ereignislog (nur bei Einsatz unter Windows und darf daher auch „`NoLog`" enthalten) geschrieben bzw. geloggt werden sollen:

```
"Logging": {
  "Console": {
    "LogLevel": "Information"
  },
  "File": {
    "LogLevel": "Debug",
    "LogFilePath": ".//DeltaLoad-{Date}.log",
    "RollingLogSizeBytes": "5000000"
  },
  "EventLog": {
    "LogLevel":  "NoLog"
  }
}
```

Anzugeben ist beim LogLevel das Mindestniveau, das dort geschrieben werden soll. Dabei werden folgende LogLevel (in der Reihenfolge von „Ausführlich" bis „Nur Kritisches") verwendet:

LogLevel	Kürzel in der Ausgabe	Beschreibung
`Verbose`	**VRB**	Dieses LogLevel gibt über den Debug-LogLevel hinaus noch Auskunft darüber, welche Verzweigungen oder welche Methoden wann durchlaufen werden. Beim LogLevel `"Verbose"` ist es noch wichtiger als beim LogLevel `"Debug"`, es nicht dauerhaft einzustellen, sondern nur auf Anweisung von ene't im Supportfall. Ansonsten könnten die Logdateien schnell die Festplatte vollschreiben.
`Debug`	**DBG**	Das LogLevel `"Debug"` dient der Analyse von bestimmten Sachverhalten, zumeist Fehlern/Bugs, daher werden hier alle Parameter protokolliert, die der Nachvollziehbarkeit von Ergebnissen dienen. Da mit diesem LogLevel viel und oft geschrieben wird, können Logdateien unnötig groß werden. Aus diesem Grund ist diese LogLevel sparsam zu verwenden.
`Information`	**INF**	Dies ist der Standard-LogLevel, unter dem die wichtigsten Arbeitsinformationen und Fortschrittsstatus des DeltaLoad-Prozesses ausgegeben werden.
`Warning`	**WRN**	Alles, was zu Fehlern im Programmfluss führen kann, aber nicht muss, wird unter dem LogLevel `"Warning"` ausgegeben, z. B. eine unvollständige Konfiguration, die auf ein vordefiniertes Standardverhalten zurückfällt.

Einrichtung DeltaLoad

LogLevel	Kürzel in der Ausgabe	Beschreibung
Error	**ERR**	Nur Fehlersituationen und kritische Ausnahmen – die zum Abbruch des DeltaLoad-Programmlaufs führen – werden ausgegeben. Fehlersituationen mit dem Status "Error" führen nicht zwangsweise zum Abbruch des DeltaLoad-Prozesses, z. B. wenn ein kurzweiliger Verbindungsausfall auftritt und im Wiederholversuch die Verbindung wieder verfügbar ist.
Critical	**CRI**	Nur kritische Fehler – die zum Abbruch des DeltaLoad-Programmlaufs führen – werden ausgegeben.
NoLog	*entfällt*	Es wird nichts in die Ausgabe geschrieben. Dieser Modus ist nicht für den Produktivbetrieb geeignet und sollte nur auf Anweisung von ene't im Supportfall eingestellt werden.

Fehlt ein solcher Eintrag für eine der Ausgabeziele **Console** oder **File**, so wird dort standardmäßig das LogLevel "Information" angewendet. Fehlt der Eintrag für Ausgabeziel **EventLog**, so wird dort "NoLog" angenommen.

Im Konfigurationsparameter **LogFilePath** in der Logging-Sektion **File** kann auf den Dateinamen der Logdatei Einfluss genommen werden. Dabei stellt der String „{Date}" am Ende des Basisnamens eine Besonderheit dar. In jeden Fall wird eine neue Logdatei für jeden Tag erstellt und ein Datumsstempel nach dem Muster „yyyyMMdd" an den Basisnamen bei der Erstellung angehängt. Der Literalstring „{Date}" kann am Ende des Basisnamens in der Konfigurationsdatei auch weggelassen werden, führt jedoch zu demselben Benennungsverhalten. Er dient in der Konfigurationsdatei lediglich dem besseren Verständnis über das Verhalten der Logdateien. Als Verzeichnistrenner sollte – wie im Beispiel oben gezeigt – der doppelte Slash („//") verwendet werden, der sowohl unter Windows als auch unter Linux richtig interpretiert wird, wenn der Pfad in Anführungszeichen adressiert wird (dies ist beim DeltaLoad der Fall). Der Punkt („.") im angegebenen Pfad des Logfiles steht für das Arbeitsverzeichnis des DeltaLoad (entspricht dem Standort des Programms und der Konfigurationsdatei).

Mit dem Konfigurationsparameter "RollingLogSizeBytes" kann die maximale Größe von Logdateien (in Bytes) insofern beschränkt werden, dass bei Überschreitung den Basisnamen der Logdateien jeweils eine laufende Nummer der Form „_001" angehängt werden, z. B. „DeltaLoad-20201012_001.log", „DeltaLoad-20201012_002.log" etc.

Darüber hinaus sorgt das DeltaLoad selbstständig dafür, dass seine Logdateien, die älter als 31 Tage sind (~1 Monat), gelöscht werden.

Automatische Korrektur des API-Pfades in der URI des REST-Webservers:
Der API-Pfad des REST-Webservice zum Abholen der Deltas hat sich geändert auf **/marktdaten**. Der Servername des REST-Webservice ist unverändert geblieben.

Ältere Pfade (**/delta** oder **/deltamigration**) – die nur noch für eine Übergangszeit aktiv sind – oder fehlende Pfade im Konfigurationsparameter "RestUriRoot" werden im DeltaLoad ab Version 1.53 automatisch korrigiert.

Einrichtung DeltaLoad

Bei einem falschen API-Pfad wird lediglich eine Konfigurations-Warnung beim Start von DeltaLoad mitgeloggt. Ein korrekter Parameter `"RestUriRoot"` im Bereich **`General`** in der Konfigurationdatei **`appsettings.json`** sieht künftig wie folgt aus:

```
[..]
"RestUriRoot": "https://ws.enet-navigator.de/marktdaten",
[..]
```

Zusätzliche Einstellungsparameter für sichere Datenbankverbindungen:

In der Konfigurationsdatei **`appsettings.json`** können jetzt die neuen Parameter `"SslMode"`, `"TargetDbPort"` und `"TrustAllCertificates"` im Block **`Destination`** eingefügt werden, die sich auf die Verbindung zur Kundendatenbank auswirken. Sind obige Parameter nicht vorhanden, so verhält sich DeltaLoad bzgl. der Datenbankverbindung standardmäßig wie bisher bzw. automatisch wie mit folgenden Werten.

Standardwerte bei DbType POSTGRES bzw. POSTGRESQL:

```
"Destination": {
  [..]
  "TargetDbPort": "5432",              => TCP-Port, über den der Postgres-Server angesprochen wird
  "SslMode": "disable",                => standardmäßig unverschlüsselte Verbindung zu Postgres
  "TrustAllCertificates": "false",     => standardmäßig wird keinem selbst-signierten Zertifikat vertraut
  [..]
},
```

Standardwerte bei DbType MSSQL:

```
  "TargetDbPort": "1433",              =  Port, über den der Microsoft SQL-Server angesprochen wird
  "SslMode": "true",                   => Wert für den ConnectionString-Parameter "Integrated Security"
  "TrustAllCertificates": "false",     => irrelevant für MSSQL Verbindungen
```

Mittels des Parameters `"TargetDbPort"` kann ein vom Standard abweichender TCP-Port zur Verbindung des Datenbankservers festgelegt werden (die Standardports sind oben im Beispiel dargestellt).

Folgende Werte sind für `"SslMode"` - der in erster Linie für Postgres-Verbindungen Verwendung findet – möglich:

Bei DbType POSTGRES bzw. POSTGRESQL:

Wert	Beschreibung
`Disable`	Verbindungen zum Postgres-Server erfolgen unverschlüsselt (Default)
`Prefer`	Verbindung bevorzugt via SSL/TLS verschlüsselt aufbauen, sofern der Server dies unterstützt, ansonsten auch ohne Verschlüsselung verbinden
`Require`	Verbindung zum Postgres-Server nur via SSL/TLS verschlüsselt erlauben, aber ohne explizit die Gültigkeit der SSL/TLS-Zertifikate zu prüfen (darauf vertrauen, dass das Netzwerk die Echtheit der Gegenstellen sicherstellt)

Einrichtung DeltaLoad

Bei DbType MSSQL:

Wert	Beschreibung
true	Es wird „`Integrated Security=true;`" für die Verbindung zum MSSQL-Server verwendet und entsprechend der Vertrauensstellung ein **Windows/Active-Directory-Konto zur Verbindung zum SQL-Server** verwendet (Default)
false	Der ConnectionString Term „`Integrated Security=true;`" entfällt und es können **nur einfache Microsoft SQL-Server User-Logins** verwendet werden.

Der Parameter `"TrustAllCertificates"` in der **`appsettings.json`** entspricht der Postgres ConnectionString-Option „`Trust Server Certificate`" und kann `"true"` oder `"false"` (default) gesetzt werden. Dieser Parameter kann auf `"true"` gesetzt werden, wenn der Postgres-Server ein selbst-signiertes Zertifikat verwendet, da dies ansonsten nicht als gültiges Zertifikat akzeptiert würde und die SSL/TLS-Verbindung trotzdem fehlschlägt. Dieser Parameter findet hauptsächlich in Kombination mit dem SSL-Mode `"Require"` Verwendung.

Da in DeltaLoad für Postgres-Verbindungen die Bibliothek NPGSQL verwendet wird, können Sie die obigen Parameter auch mit anderen Funktionsweisen dieser Bibliothek kombinieren, um besondere Szenarien abzubilden – beispielsweise über Umgebungsvariablen wie PGPASSFILE oder PGSSLCERT.

Weiterführende Informationen hierzu finden Sie unter: https://www.npgsql.org/doc/security.html

Automatische Verschlüsselung sensibler Konfigurationsdaten:
DeltaLoad verschlüsselt ab Version 1.53 die sensiblen Konfigurationsparameter. Es wird zwar davon ausgegangen, dass DeltaLoad nur auf Servern ausgeführt wird, dessen Zugriff auf zuständige Administratoren beschränkt ist. Damit sie aber nicht im Klartext in der Konfigurationsdatei stehen, wird bei einem (Neu-)Start von DeltaLoad eine simple Rot13-Verschlüsselung auf diesen Parametern in der Konfigurationsdatei
`appsettings.json` durchgeführt, wenn der Parameter `"ConfCrypted"` auf `"false"` gesetzt wird. Ab diesem Zeitpunkt erscheinen diese sensiblen Werte dann verschlüsselt in der Konfigurationsdatei und der Parameter `"ConfCrypted"` wechselt den Status auf `"true"`.

Folgende Konfigurationsparameter sind fortan als sensibel eingestuft:

- `mTokenMain`
- `DbPassword`

Neue oder geänderte Werte können sie also wie folgt in der Konfigurationsdatei **`appsettings.json`** hinterlegen:

```
{
  "General": {
    "ConfCrypted": "false",
    [..]
  },
  "MachineToken": {
    "mTokenPrefix": "<Benutzername des Maschinenkontos für den Webservice
                     Marktdaten>",
    "mTokenMain": "< Passwort des Maschinenkontos im Klartext>",
    [..]
  },
```

Einrichtung DeltaLoad

```
  "Destination": {
    [..]
    "DbUserName": "<neuer DB-Benutzername>",
    "DbPassword": "<neues DB-Passwort im Klartext>",
    [..]
  },
  "Logging": {
    [..]
  }
}
```

Nach dem nächsten Start von DeltaLoad steht `"ConfCrypted"` dann auf `"true"` und die betroffenen sensiblen Konfigurationsparameter enthalten die verschlüsselten Werte, die DeltaLoad als solche versteht.

Hinweis: *Nach wie vor können Sie die Werte für die Parameter `"DbUserName"` und `"DbPassword"` auch leer lassen, wenn Sie DeltaLoad unter einerm zugriffsberechtigten Benutzer (z. B. Domänenbenutzer mit entsprechender Berechtigung) ausführen.*

Differenzierter Exitcode:
Je nach Beendigungsgrund gibt das Programm DeltaLoad nun unterschiedliche, aussagekräftige Exitcodes zurück. Dieser Exitcode kann bei Bedarf (z. B. eingebettet in einem Console-Skript) abgefragt werden. Folgende Status sind möglich:

Wert	Kürzel	Beschreibung
0	**Success**	DeltaLoad wurde als Programm oder Service/Daemon normal beendet
1	**InvalidCmdParameterError**	Es gab ein Problem in den Konfigurationsparametern
2	**DeltaMismatchError**	Die zur Verarbeitung ausgewählte Delta-ID passt nicht zur letzten erfolgreich in der Datenbank verarbeiteten Delta-ID (Info-Tabelle)
3	**InputApiError**	Die Verbindung zum REST-Webservice als Quelle der Deltas konnte nicht aufgebaut werden oder meldet wiederholt Fehler
4	**DatabaseConnectionError**	Die Verbindung zur Ziel-Datenbank konnte nicht aufgebaut werden
5	**DeltaSqlUpdateError**	Ein Problem beim Einspielen des Deltas ist in der Ziel-Datenbank aufgetreten (und hat ein Rollback des aktuellen Deltas ausgelöst)
6	**UserAbort**	DeltaLoad wurde interaktiv ausgeführt und durch einen Tastendruck (z. B. „q“) oder aber durch eine Datei **`stop.txt`** normal beendet
10	**UnknownError**	Es ist ein unbekannter Fehler aufgetreten (und hat ein Rollback des aktuellen Deltas ausgelöst)

Einrichtung DeltaLoad

Änderungshistorie DeltaLoad

Version 1.61 / Dezember 2021:

- Unterstützung von abweichendem Datenbank-Schema in der Zieldatenbank
  (Parameter `"DatabaseSchema"` *in der* `appsettings.json` *mit Wert ungleich* `"ndbd"` *wird jetzt auch berücksichtigt)*
- Verbesserte Debugausgaben, verbesserte Plausibilitätsprüfung bei Pfadangaben

Bugfixes:

- Korrektur bei Kommentarfeldinhalten mit Zeilenumbrüchen

Version 1.56 / Mai 2021:

- Verbesserte Unterstützung langer Konfigurationswerte
  (z. B. besonders lange Endpunktnamen einer Cloud-Datenbank in `"TargetDbServer"`)
  (Maximale Länge von Konfigurationswerten auf 256 Zeichen erhöht)

Version 1.55 / Mai 2021:

- Unterstützung nicht-deutscher MSSQL-Zielserver bzw. User Languages zur Akzeptanz des von uns verwendeten Datumsformats
  (Dateformat `"dmy"` *vor jedem zu verarbeitenden Batch)*

Version 1.54 / Mai 2021:

- Verbesserung von Delta-Transaktionen mit langen Laufzeiten in MSSQL
  (`"CommandTimeout"` *bei MSSQL-Server als Zieldatenbank erhöht [vorher: 1.000 Sek.])*

Version 1.53 / November 2020:

- **Umfangreiches, differenziertes Logging**[*)], einstellbar über die Konfigurationsdatei **`appsettings.json`**
- Der Rest-API-Pfad für Delta-Updates wurde umbenannt in **`/marktdaten`**. Es erfolgt in DeltaLoad jetzt eine **automatische Korrektur des API-Pfades in der URI des REST-Webservers**[*)] zum Abholen der Delta-Updates bei ene't
- **Automatische Verschlüsselung sensibler Konfigurationsdaten**[*)] in der Datei **`appsettings.json`**
- **Zusätzliche Einstellungsparameter für sichere Datenbankverbindungen**[*)] in der Datei **`appsettings.json`**: `"SslMode", "TrustAllCerts", "TargetDbPort"`
- **Differenzierter Exitcode**[*)] gibt Aufschluss über den Beendigungsgrund von DeltaLoad
- Automatische Unterscheidung zwischen interaktiver und automatischer Ausführung inkl. Abbruchmöglichkeit bei Tastendruck („q“) im interaktiven Modus
- In der Sektion „General“ der **`appsettings.json`** einstellbare Parameter `"CheckInterval"` (für die Eingangsseite) und `"ConnectionTimeout"` (für die Verbindung zur Zieldatenbank) in Millisekunden
- Paralleles Befüllen eines Datenbankziels von 2 gleichzeitig laufenden DeltaLoad-Instanzen (auch von verschiedenen Rechnern aus) ist jetzt möglich. Ein Abbruch der zweiten Instanz erfolgt nicht. Die richtige Reihenfolge der Delta-Verarbeitung wird eingehalten. Wann welche DeltaLoad-Instanz welches Delta-Paket verwendet, ist aber von verschiedenen Faktoren abhängig und nicht vorherbestimmbar.
- Ausgabe der Versionsnummer beim Starten mit dem Parameter **`--version`**

[*)] Siehe Anhang-Abschnitt "Wichtige Details zu neuen Funktionen (ab Version 1.53)"

- Ausgabe einer ausführlicheren Kommandozeilen-Hilfe beim Starten mit dem Parameter **`--help`**

Bugfixes:

- Behebung von seltenen Race-Conditions, die zur außerplanmäßigen Beendigung von DeltaLoad (und Rollback des Deltas) geführt haben.

Version 1.33 / August 2020:

- Zusätzlicher Support von Microsoft SQL Server als Zieldatenbank

Version 1.32 / Juli 2020:

- Erstes öffentliches Release mit Support von Postgres als Zieldatenbank