DBTP

Data Base Transfer Protokoll
Schnittstellenbeschreibung

toposoft GmbH

16. Juni 2020

Inhaltsverzeichnis

1 Einleitung

Applikationen von toposoft benötigen Zugriff auf Stammdaten und Dateien. Prinzipiellbenötigt die jeweilige Applikation Schreibrechte auf alle beteiligten Dateien. Dies ist jedoch damit verbunden, dass der Benutzer diese Dateien direkt im Verzeichnis schreiben können muss. Dies gefährdet jedoch die Stabilität der Anwendung und die Integrität der Daten.

Es ist also ratsam, die Daten dem direkten Zugriff durch den Benutzer auf Dateiebene zu entziehen. Der Benutzer soll auf seinem Arbeitsplatz also keine Dateien sehen. Die Applikation muss demnach über ein direktes Netzwerkprotokoll auf die Dateien zugreifen. Diese liegen auf einem Server und sind direkt nur durch einen Administrator manipulierbar, der Zugriff auf den Server hat.

DBTP ist ein solches Netzwerkprotokoll und im Folgenden beschrieben.

2 Funktionsbeschreibung

Befehle werden an den DBTP-Server in Form einer URL geschickt, die den Hostnamen, den Port, ggf. den Namen einer Relation/Datei sowie die Funktion beinhaltet. Diese URL muss mit dem http-Protokoll als GET (oder ggf. POST) versendet werden. Beispiel

```
http://dbtp.server.net:8040/kerndaten?INFO
```

Das DBTP-Protokoll umfasst folgende Befehle:

Befehle für Relationen

Befehle für den allgemeinen Zugriff, die wichtigsten Befehle

- SEARCHALL - Filtert die Tupel einer Relation ahand von Suchkriterien
- GETVAL - liefert den Inhalt eines Feldes/mehrerer Felder
- SETVAL - setzt den Inhalt eines Feldes
- GETREL - liefert eine Relation als Ganzes zurück
- APPTUP - nimmt Tupel in eine Relation auf

weitere allgemeine Befehle

- INFO - liefert Eigenschaften einer Relation zurück
- CREATE - legt eine neue Relation an
- UPDTUP - aktualisiert ein Tupel

- GETTUP - liefert ein Tupel einer Relation
- DELTUP - löscht ein Tupel aus einer Relation
- UPDATEVAL - aktualisiert und liefert den Inhalt eines Feldes

Spezielle Befehle

- WRITABLE - liefert zurück, ob eine Relation schreibbar ist
- TIMESTAMP - liefert das Änderungsdatum einer Relation
- FILTERNEWER - filtert Relationen, die sich geändert haben
- DELREL - löscht eine Relation
- REWRITE - aktualisiert ein Tupel
- LOCK - sperrt ein Tupel zum Schreiben
- UNLOCK - entsperrt ein Tupel

Befehle für Dateien

- WRITABLE - liefert zurück, ob eine Datei schreibbar ist
- TIMESTAMP - liefert das Änderungsdatum einer Datei
- GETFILE - liefert den Inhalt einer Datei zurück
- PUTFILE - sendet eine Datei an den Server
- DELFILE - löscht eine Datei
- DIRECTORY - liefert den Verzeichnisinhalt mit Dateisuchmuster
- STATFILE - Dateininfo
- RENAME - benennt Dateien um

Alle Funktionen sind im Folgenden beschrieben.

Der Funktionsaufruf vom Client wird als Anfrage in Form eines HTTP-Requests an den DBTP-Server weitergegeben. Jeder Funktion sind dabei Argumente und Rückgabewerte zugeordnet. Die Argumente werden per POST im XML-Format versendet. Die Rückgabewerte werden ebenfalls im XML-Format übermittelt. Die URL besitzt neben dem Funktionsnamen keine Parameter.

Massendaten (Tupel, Relation oder Dateien), die zum Server geschickt werden, sowie Directory-Anfragen werden mittels HTTP-POST versendet (z. B. bei `PUTFILE`).

Das Ergebnis einer Anfrage ist ein mit einem http-Kopf versehendes XML-Format. Die http-Antwort ist normalerweise 200. Ist die URL fehlerhaft gebildet (z. B. unbekannte Funktion), wird 400 geliefert (Bad Request). Beim Fehlschlagen des Einloggens wird 401 geliefert (Unauthorized).

Die XML-Antwort ist in eine <DBTP>-Kennung eingeschlossen. Die Zeichenkodierung ist ISO-8859-1 (LATIN-1).

Andere Fehler, z. B. (Relation nicht gefunden) werden in der XML-Kennung <ERR> angezeigt. Liegt kein Fehler vor, fehlt diese Kennung. Mögliche Fehler sind

- NOT FOUND
- NO WRITE ACCESS
- NO READ ACCESS
- NO CREATE/DELETE ACCESS
- LOCK FAILED
- REL ALREADY EXISTS
- TUPNUM OUT OF RANGE

2.1 SEARCHALL

Filtert alle Tupel einer Relationen, die auf ein Suchmuster passen. Dieser Befehl entspricht dem SELCET-befehl aus SQL.

Das Suchmuster wird in der URL angegeben, die Inhalte können einzelne Werte, Bereiche und untere oder obere Grenzen sein.

Beispiel: `http://host:8040/zeit_geber?SearchAll&BEGINN=<2000&FToleranz=0-1`

Es werden alle Tupel aus zeit_geber gefiltert, deren Beginn vor dem 1.1.2000 und deren FToleranz zwischen 0 und 1 liegt.

Mit dem Zusatz `=XML` erhält man das Ergebnis in reinem XML: `http://host:8040/zeit_geber?SearchAll=XML&BEGINN=<2000&FToleranz=0-1`

Falls das Suchmuster auf einen Feldinhalt passen soll, der ein - enthält, muss es statt des Minus eine Tilde (~) enthalten.

Ohne Suchkriterien erhält man die gesamte Relation, als XML z.B. so:

`http://host:8040/zeit_geber?SearchAll=XML`

Werden nur bestimmte Felder des Ergebnisses benötigt, so können diese mit dem Parameter `PROJECTION` als mit Kommas getrennte Liste übergeben werden. Z.B.:

`http://host:8040/zeit_geber?SearchAll&Ort=01130810&PROJECTION=Ort,GeberNr`

Dadurch wird das Ergebnis auf die ausgewählten Felder projiziert.

2.2 WRITABLE

Liefert in der Kennung `<WRITABLE>` `True` oder `False`.

Beispiel: `<WRITABLE>True</WRITABLE>`.

2.3 TIMESTAMP

Liefert in einer `<TIMESTAMP>`-Kennung den Zeitstempel der Relation oder Datei. Der Zeitstempel ist ein 8-Zeichen-Hexadezimalwert und enthält die Sekunden ab 1.1.1970.

2.4 FILTERNEWER

Mittels POST wird eine Liste mit `<ITEM>`-Kennungen verschickt, die jeweils den Namen einer Relation und mit `@` getrennt ihren Zeitstempel enthalten. In der URL selbst wird demnach kein Relationsname verschickt.

Als Ergebnis wird eine wie oben aufgebaute Liste geliefert, die jedoch nur noch die Relationen mit deren aktuellen Zeitstempel enthält, die auf dem Server neuer als auf dem Client sind.

Die hier verwendeten Zeitstempel sind dezimal kodiert.

2.5 INFO

Liefert zurück, ob die Relation schreibbar ist, ihren Zeitstempel, ihre Struktur und die Anzahl Tupel. Beispiel:

```
<WRITABLE>True</WRITABLE>
<TIMESTAMP>4485D1A4</TIMESTAMP>
<STRUCT>ORT#20S,HISTORISCH#20S,KOMMENTAR#100S</STRUCT>
<NUMTUP>12</NUMTUP>
```

2.6 GETREL

Liefert die gesamte Relation als Binärblock zurück. Das Format ist im Anhang beschrieben. Die Daten werden Base64-kodiert in einer `<DATA>`-Kennung geschickt.

2.7 CREATE

Legt eine Relation an. Die Struktur wird mittels POST verschickt.

Beispiel:

```
<STRUCT>ORT#20S,HISTORISCH#20S,KOMMENTAR#100S</STRUCT>
```

2.8 DELREL

Die Relation wird gelöscht.

2.9 APPTUP

Ein oder mehrere Tupel werden an die Relation angehängt. Die Tupel werden mittels POST in jeweils einer `<DATA>`-Kennung verschickt. Die Tupelnummern, die der Server diesen neuen Tupeln zuweist, werden in `<TUPNUM>`-Kennungen dezimal zurückgeliefert, deren Reihenfolge der der übermittelten Tupel entspricht.

2.10 UPDTUP

Ein Tupel der Relation wird aktualisiert. Übermittelt wird, wie bei APPTUP, ein Tupel als DATA eines POST. In der URL wird als `KEY` das Schlüsselfeld angegeben, oder die Schlüsselfelder mit `+` getrennt.

Anhand des Schlüssels im übermittelten Tupel wird das passende Tupel in der Relation gesucht und darauf mit den neuen Werten überschrieben. Ist kein passendes Tupel vorhanden, wird eins angehängt.

Die Tupelnummer, die das aktualisierte Tupel hat, wird in einer `<TUPNUM>`-Kennung dezimal zurückgeliefert.

2.11 GETTUP

Die Tupelnummer des Tupels wird dezimal in einer `<TUPNUM>`-Kennung per POST übermittelt.

2.12 DELTUP

Die Tupelnummer des zu löschenden Tupels wird dezimal in einer `<TUPNUM>`-Kennung per POST übermittelt.

2.13 SETVAL

Neben dem schon angegebenen Namen der Relation werden als URL-Parameter der Name des Schlüsselfelds, der Schlüssel, der Name der Wertfelder und die Werte angegeben. Beispiel:

`http://dbtp.server.net:8040/kerndaten?SETVAL&ORT=Aachen&SPRACHE=de&PLZ=1`

Setzt in der Relation kerndaten in dem Tupel, in dem ORT auf Aachen steht, das Feld SPRACHE auf de und das Feld PLZ auf 1. Falls kein Tupel mit diesem Schlüssel vorhanden ist, wird es angelegt.

2.14 GETVAL

Neben dem schon angegebenen Namen der Relation werden als URL-Parameter der Name des Schlüsselfelds, der Schlüssel und der Name des Wertfeldes angegeben. Beispiel:

`http://dbtp.server.net:8040/kerndaten?GETVAL&ORT=Aachen&SPRACHE`

Liefert den Inhalt des Felds SPRACHE des Tupels, in dem ORT auf Aachen steht, der Relation kerndaten. Das Ergebnis wird in einer `<RET>`-Kennung geliefert.

Es ist auch möglich, den Inhalt mehrerer Wertfelder abzufragen, deren Namen mit `+` getrennt angegeben werden. Im Ergebnis sind die Werte dann abenfalls mit `+` getrennt. Beispiel:

`http://dbtp.server.net:8040/kerndaten?GETVAL&ORT=Aachen&SPRACHE+PLZ`

In der `<RET>`-Kennung würde geliefert: `de+52000`.

2.15 UPDATEVAL

Dieser Befehl dient dazu, das Aktualisieren und Abfragen zu einem atomaren Schritt zu vereinen. Neben dem schon angegebenen Namen der Relation werden als URL-Parameter der Name des Schlüsselfelds, der Schlüssel, der Name des Wertfeldes und das Delta angegeben.

Beispiel:

`http://dbtp.server.net:8040/idrel?GETVAL&TYP=MaxId&Wert&1`

Angesprochen wird das Tupel, das im Feld Typ den Wert MaxId hat. Dessen Feld Wert wird um 1 erhöht. Dieser erhöhte Wert wird nun in einer `<RET>`-Kennung geliefert.

2.16 REWRITE

Das Tupel wird binär in einer `<DATA>`-Kennung verschickt (siehe APPTUPEL) die Tupelnummer wird wie bei DELTUP übermittelt.

2.17 LOCK

Die Tupelnummer des zu sperrenden Tupels wird dezimal in einer `<TUPNUM>`-Kennung per POST übermittelt. War das Tupel bereits gesperrt, wird der Fehler LOCK FAILED geliefert. Schlägt die Aktion aus einem anderen Grund fehl, wird der entsprechende Fehler geliefert. War das Sperren erfolgreich, wird in einer `<RET>`-Kennung `OK` geliefert.

2.18 UNLOCK

Die Tupelnummer des zu entsperrenden Tupels wird dezimal in einer `<TUPNUM>`-Kennung per POST übermittelt.

2.19 SYNCGET

Es wird ein Zeitpunkt in der Anfrage übermittelt. Das Ergebnis ist ein Datenbank, die alle Relationen enthält, die sich seit diesem Zeitpunkt geändert haben.

Beispiel:

```
http://dbtp.server.net:8040/?SYNCGET&Time=8.4.2016_7:53
```

2.20 GETFILE

Liefert die gesamte Datei als Binärblock zurück. Das Format ist im Anhang beschrieben. Die Daten werden Base64-kodiert in einer `<DATA>`-Kennung geschickt. (zum Format siehe PUTFILE).

2.21 PUTFILE

Schickt eine Datei an den Server. Die Daten werden Base64-kodiert in einer `<DATA>`-Kennung geschickt.

Beispiel:

```
<?xml version="1.0">
<DBTP RELEASE="1">
   <DATA size="123" name="eine_datei.txt">
      <![CDATA[LyogWFBNICov ... 9Owo=]]>
   </DATA>
</DBTP>
```

Das Attribut `size` bezieht sich dabei auf die Größe der Datei, nicht die der kodierten Daten.

2.22 DELFILE

Die Datei wird gelöscht.

2.23 RENAME

Die Datei wird umbenannt.

Beispiel:

```
http://host:8040/result.txt?Rename&Name=ergebnis.txt
```

Die Datei result.txt wird in ergebnis.txt umbenannt.

2.24 DIRECTORY

In der URL wird, analog GETREL, der Name eines Verzeichnisses übermittelt. Im Datablock der POST-Anweisung wird das Muster als XML-Baum angegeben. Beispiel:

```
<PATTERN>F:*.dat</PATTERN>
```

Mit `F` sucht man nach Dateien, die auf das Muster passen, mit `D` Verzeichnisse.

3 Format der Anfrage

3.1 Authentifizierung

Im HTTP-Header einer Anfrage werden in einer Zeile Benutzername und Password im Format „`Authorization: Basic username:password`“ übergeben, wobei die Zeichenkette „`username:password`“ nach Base64 (RFC 3548, Kapitel 3) kodiert wird.

Beispiel:

```
GET /kerndaten?INFO HTTP/1.0
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ
```

Vom DBTP-Server wird der Benutzer mit dem angegebenen Passwort überprüft. Sollte das Passwort falsch sein oder der Benutzer nicht existieren, wird eine „HTTP-401“-Rückmeldung generiert. In einem Browser führt dies zu einer Login-Maske.

Den Benutzern sind Rechte zum Benutzen von Stammdaten zugeteilt. Ein Benutzer kann entweder

- nichts,
- nur Lesen,
- Lesen/Schreiben

4 XML-Format

XML-kodierte Antworten des DBTP-Servers beginnen mit einem HTTP-Header, gefolgt von den XML-kodierten Daten.

Beispiel:

```
HTTP/1.0 200 OK
Date: Tue, 28 Oct 2009 13:49:10 GMT
Server: DBTPD/1.0 (UNIX)
```

```
Last-Modified: Tue, 28 Oct 2003 13:49:10 GMT
Expires: Tue, 28 Oct 2009 13:49:10 GMT
Cache-Control: max-age=0
Connection: close
Content-Length: 6733
Content-Type: text/plain; charset=ISO-8859-1

<?XML version="1.0" encoding="ISO-8859-1"?>
<DBTP RELEASE="1">
 ...
</DBTP>
```

Die Zeilen sind mit einem Linefeed (ASCII-Char 10) abgeschlossen.

5 DBTP starten und einrichten

5.1 Starten und beenden

Auf dem Server wird DBTP folgendermaßen gestartet:

```
dbtpd [option] [&]
```

Das `&`-Zeichen am Ende ermöglicht bei Bedarf, dass der gestartete DBTP-Prozess als „Dämon“ im Hintergrund weiterläuft.

Beim Start werden eine Reihe von Statusmeldungen ausgegeben.

Wegen der Ausführung im Hintergund wird die Prozessnummer (`30272`) von der Shell ausgegeben.

Üblicherweise wird als DBTP-Port der Port `8040` genutzt. Ein Testbeispiel findet sich unter Kapitel 2.

Die Authentifizierung des Clients, der eine Abfrage an den DBTP-Server richtet, ist üblicherweise aktiviert (`Authentication on`).

Zum Beenden des im Hintergrund laufenden DBTP-Dämons muss man den `killall`-Befehl verwenden.

```
>killall dbtpd
```

5.2 Optionen

Beim Start kann der DBTP-Dämon anhand einiger Optionen gesteuert werden. Die folgende Übersicht wird auch ausgegeben, wenn man `dbtpd` mit nicht bekannten Optionen versucht zu starten:

```
usage: dbtpd [OPTION]
         options:
         -h, --help       : shows this list
         -v, --version    : shows the version number
         -V, --verbose    : verbose output
         -noauth          : disable authentication
         -nowrite         : disable all write access
         -p <port>        : use port <port> rather than 8030
         -koo=<XKoo,YKoo[,UTMZone] : define coordinatefields
```

Im Klartext:

- `--help`: Hilfe:
  Hier wird die Hilfe auf der Konsole ausgegeben.
- `--version`: Version:
  Zeigt die Version des DBTP-Servers an.
- `--verbose`: Verbose:
  Veranlasst den DBTP-Server Anfragen auf der Konsole auszugeben.
- `-noauth`: Keine Anmeldung:
  Mit dieser Option wird die Angabe von Benutzer und Passwort nicht mehr gefordert. Siehe Kapitel 3.1.
- `-nowrite`: Lesezugriff:
  Erlaubt nur lesenden Zugriff auf die Daten.
- `-p <port>`: Port:
  Definiert den Port, auf welchem der DBTP-Server erreichbar ist. Der Standardport ist Port 8040. Hiermit ist es möglich, mehrere DBTP-Server parallel auf verschiedenen Ports laufen zu lassen.
- `-koo`: Koordinaten:
  Definiert Feldnamen für Koordinaten. Enthält eine Relation alle diese Felder, kann der DBTP-Server die Koordinaten auf Anfrage des Clients in ein anderes Zielsystem umrechnen. (z. B. -koo=KOORDX,KOORDY,UTMZONE) Der Client kann in der Anfrage per `setkoosys=[GK,GEO,UTM]` ein bestimmtes Zielsystem anfragen. Eine Anfrage im Browser würde z. B. so aussehen: `http://127.0.0.1:8040/zeit_lage.dbf?GETREL=HTML?setkoosys=GK`. Die möglichen Koordinatensysteme sind:
  - GK: Gauß-Krüger
  - Geo: Geographisch, (Longitude, Latitude)
  - UTM: UTM (Universal Transverse Mercator)