Fews JDBC server - EOL 2022.02

> This module will be removed from the Delft-FEWS code in the 2022.02 release

Introduction

To be able to query timeseries directly using SQL statements Delft-Fews can be set up to act as a jdbc server. This can be done using an OC configuration (which will log in and automatically synchronise date with the MC, thereby assuring all data is constantly being updated), or by running this stand-alone. In the latter case the system will only export what is in the local datastore at startup.

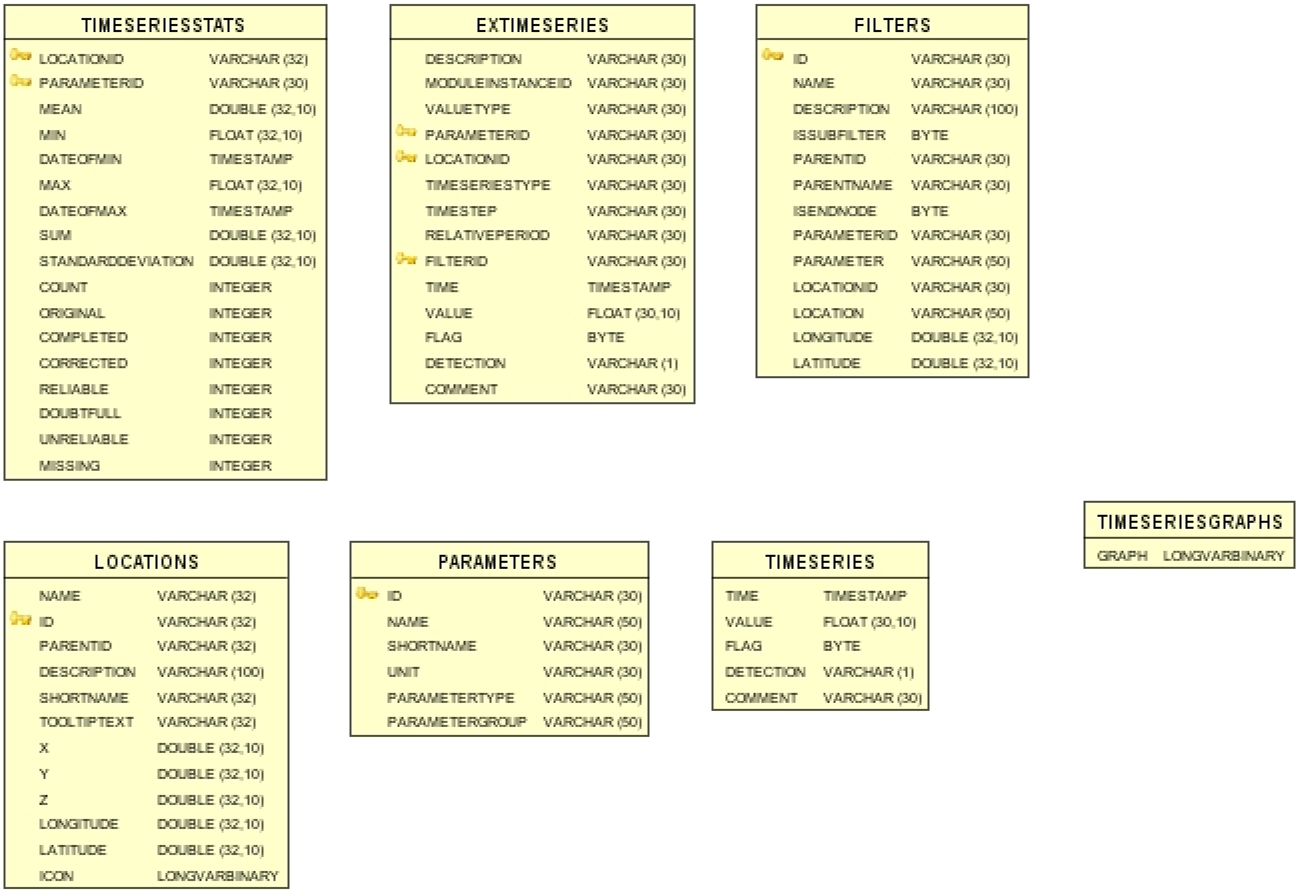
Note that the JDBC server is planned to become obsolete and to be replaced by the webservices like the PI service. Therefore no new features will be developed to the JDBC server.

Fews JDBC Interface

The JDBC Interface provides a virtual access to (virtual) FEWS tables. The JDBC server allows a client application to query the available tables. However not all SQL query statements are supported. Also the type of SQL statements allowed on a table varies per table. See the section on #SQL queries for more details.

The following information is available through the JDBC Server.

Fews JDBC data model



TIMESERIESSTATS	
🔑 LOCATIONID	VARCHAR (32)
🔑 PARAMETERID	VARCHAR (30)
MEAN	DOUBLE (32,10)
MIN	FLOAT (32,10)
DATEOFMIN	TIMESTAMP
MAX	FLOAT (32,10)
DATEOFMAX	TIMESTAMP
SUM	DOUBLE (32,10)
STANDARDDEVIATION	DOUBLE (32,10)
COUNT	INTEGER
ORIGINAL	INTEGER
COMPLETED	INTEGER
CORRECTED	INTEGER
RELIABLE	INTEGER
DOUBTFULL	INTEGER
UNRELIABLE	INTEGER
MISSING	INTEGER

EXTIMESERIES	
DESCRIPTION	VARCHAR (30)
MODULEINSTANCEID	VARCHAR (30)
VALUETYPE	VARCHAR (30)
🔑 PARAMETERID	VARCHAR (30)
🔑 LOCATIONID	VARCHAR (30)
TIMESERIESTYPE	VARCHAR (30)
TIMESTEP	VARCHAR (30)
RELATIVEPERIOD	VARCHAR (30)
🔑 FILTERID	VARCHAR (30)
TIME	TIMESTAMP
VALUE	FLOAT (30,10)
FLAG	BYTE
DETECTION	VARCHAR (1)
COMMENT	VARCHAR (30)

FILTERS	
🔑 ID	VARCHAR (30)
NAME	VARCHAR (30)
DESCRIPTION	VARCHAR (100)
ISSUBFILTER	BYTE
PARENTID	VARCHAR (30)
PARENTNAME	VARCHAR (30)
ISENDNODE	BYTE
PARAMETERID	VARCHAR (30)
PARAMETER	VARCHAR (50)
LOCATIONID	VARCHAR (30)
LOCATION	VARCHAR (50)
LONGITUDE	DOUBLE (32,10)
LATITUDE	DOUBLE (32,10)

TIMESERIESGRAPHS	
GRAPH	LONGVARBINARY

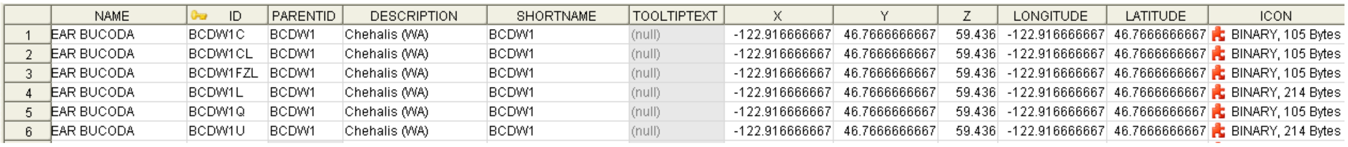
LOCATIONS	
NAME	VARCHAR (32)
🔑 ID	VARCHAR (32)
PARENTID	VARCHAR (32)
DESCRIPTION	VARCHAR (100)
SHORTNAME	VARCHAR (32)
TOOLTIPTEXT	VARCHAR (32)
X	DOUBLE (32,10)
Y	DOUBLE (32,10)
Z	DOUBLE (32,10)
LONGITUDE	DOUBLE (32,10)
LATITUDE	DOUBLE (32,10)
ICON	LONGVARBINARY

PARAMETERS	
🔑 ID	VARCHAR (30)
NAME	VARCHAR (50)
SHORTNAME	VARCHAR (30)
UNIT	VARCHAR (30)
PARAMETERTYPE	VARCHAR (50)
PARAMETERGROUP	VARCHAR (50)

TIMESERIES	
TIME	TIMESTAMP
VALUE	FLOAT (30,10)
FLAG	BYTE
DETECTION	VARCHAR (1)
COMMENT	VARCHAR (30)

Locations

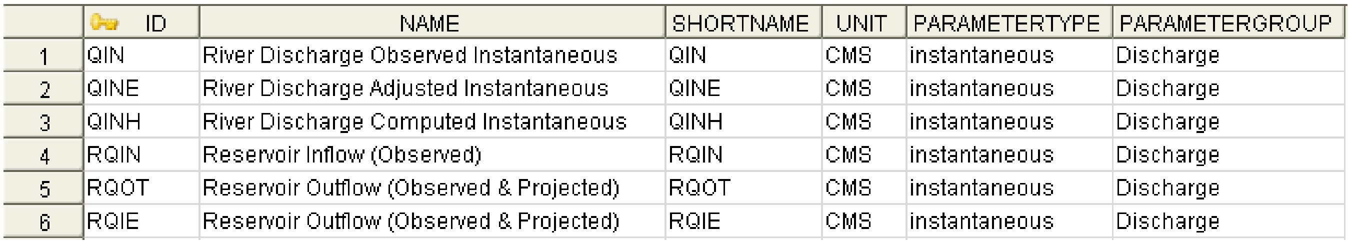
The locations table allows the client application to query the available FEWS locations.



	NAME	🔑 ID	PARENTID	DESCRIPTION	SHORTNAME	TOOLTIPTEXT	X	Y	Z	LONGITUDE	LATITUDE	ICON
1	EAR BUCODA	BCDW1C	BCDW1	Chehalis (WA)	BCDW1	(null)	-122.916666667	46.7666666667	59.436	-122.916666667	46.7666666667	BINARY, 105 Bytes
2	EAR BUCODA	BCDW1CL	BCDW1	Chehalis (WA)	BCDW1	(null)	-122.916666667	46.7666666667	59.436	-122.916666667	46.7666666667	BINARY, 105 Bytes
3	EAR BUCODA	BCDW1FZL	BCDW1	Chehalis (WA)	BCDW1	(null)	-122.916666667	46.7666666667	59.436	-122.916666667	46.7666666667	BINARY, 105 Bytes
4	EAR BUCODA	BCDW1L	BCDW1	Chehalis (WA)	BCDW1	(null)	-122.916666667	46.7666666667	59.436	-122.916666667	46.7666666667	BINARY, 214 Bytes
5	EAR BUCODA	BCDW1Q	BCDW1	Chehalis (WA)	BCDW1	(null)	-122.916666667	46.7666666667	59.436	-122.916666667	46.7666666667	BINARY, 105 Bytes
6	EAR BUCODA	BCDW1U	BCDW1	Chehalis (WA)	BCDW1	(null)	-122.916666667	46.7666666667	59.436	-122.916666667	46.7666666667	BINARY, 214 Bytes

Parameters

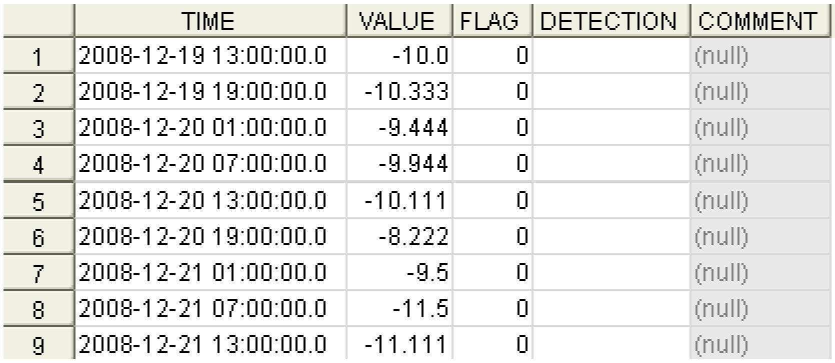
The parameters table allows the client application to query the available FEWS parameters.



	🔑 ID	NAME	SHORTNAME	UNIT	PARAMETERTYPE	PARAMETERGROUP
1	QIN	River Discharge Observed Instantaneous	QIN	CMS	instantaneous	Discharge
2	QINE	River Discharge Adjusted Instantaneous	QINE	CMS	instantaneous	Discharge
3	QINH	River Discharge Computed Instantaneous	QINH	CMS	instantaneous	Discharge
4	RQIN	Reservoir Inflow (Observed)	RQIN	CMS	instantaneous	Discharge
5	RQOT	Reservoir Outflow (Observed & Projected)	RQOT	CMS	instantaneous	Discharge
6	RQIE	Reservoir Outflow (Observed & Projected)	RQIE	CMS	instantaneous	Discharge

Timeseries

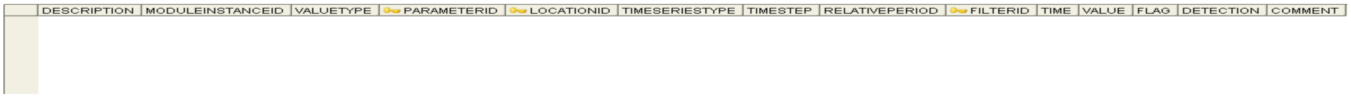
The timeseries table allows the client application to query the available FEWS timeseries. The information shown in the TimeSeries table provided by the JDBC server does not match the information of the FEWS TimeSeries table. The JDBC server provides a view of the data of a queried timeseries.



	TIME	VALUE	FLAG	DETECTION	COMMENT
1	2008-12-19 13:00:00.0	-10.0	0		(null)
2	2008-12-19 19:00:00.0	-10.333	0		(null)
3	2008-12-20 01:00:00.0	-9.444	0		(null)
4	2008-12-20 07:00:00.0	-9.944	0		(null)
5	2008-12-20 13:00:00.0	-10.111	0		(null)
6	2008-12-20 19:00:00.0	-8.222	0		(null)
7	2008-12-21 01:00:00.0	-9.5	0		(null)
8	2008-12-21 07:00:00.0	-11.5	0		(null)
9	2008-12-21 13:00:00.0	-11.111	0		(null)

ExTimeSeries

The extended timeseries table allows the client application to query the available FEWS timeseries. The information shown in the ExTimeSeries table provide by the JDBC server is similar to the information presented in the FEWS TimesSeries table. The JDBC server provides a view of the metha data of a queried timeseries.

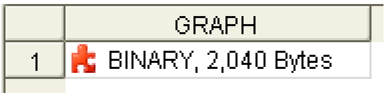


	DESCRIPTION	MODULEINSTANCEID	VALUETYPE	PARAMETERID	LOCATIONID	TIMESERIESTYPE	TIMESTEP	RELATIVEPERIOD	FILTERID	TIME	VALUE	FLAG	DETECTION	COMMENT

Note! It is currently not possible to query the ExTimeSeries due to bugs.

TimeSeriesGraphs

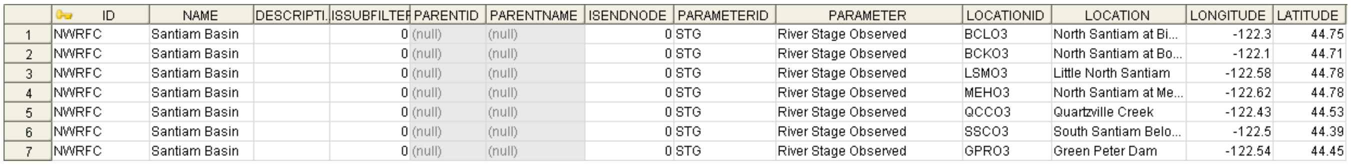
The TimeSeriesGraphs table allows the client application to retieve an image of a FEWS timeseries chart for the queried timeseries. The query returns a byte array value containing the content of a BufferedImage.



	GRAPH
1	BINARY, 2,040 Bytes

Filters

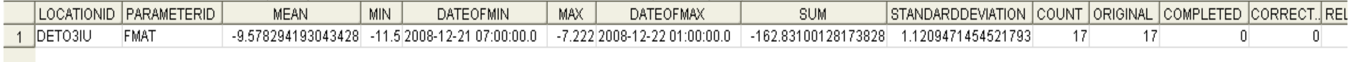
The Filters is set up as a view. This is because the Filters does not represent a FEWS table. Instead the Filters view represents the content of the FEWS configuration file 'Filters.xml'.



	ID	NAME	DESCRIPTI..	ISSUBFILTEF	PARENTID	PARENTNAME	ISENDNODE	PARAMETERID	PARAMETER	LOCATIONID	LOCATION	LONGITUDE	LATITUDE
1	NWRFC	Santiam Basin		0	(null)	(null)	0	STG	River Stage Observed	BCLO3	North Santiam at Bi...	-122.3	44.75
2	NWRFC	Santiam Basin		0	(null)	(null)	0	STG	River Stage Observed	BCKO3	North Santiam at Bo...	-122.1	44.71
3	NWRFC	Santiam Basin		0	(null)	(null)	0	STG	River Stage Observed	LSMO3	Little North Santiam	-122.58	44.78
4	NWRFC	Santiam Basin		0	(null)	(null)	0	STG	River Stage Observed	MEHO3	North Santiam at Me...	-122.62	44.78
5	NWRFC	Santiam Basin		0	(null)	(null)	0	STG	River Stage Observed	QCCO3	Quartzville Creek	-122.43	44.53
6	NWRFC	Santiam Basin		0	(null)	(null)	0	STG	River Stage Observed	SSCO3	South Santiam Belo...	-122.5	44.39
7	NWRFC	Santiam Basin		0	(null)	(null)	0	STG	River Stage Observed	GPRO3	Green Peter Dam	-122.54	44.45

TimeSeriesStats

The TimeSeriesStats is set up as a view. This is because the TimeSeriesStats does not represent a FEWS table. Instead the TimeSeriesStats view shows the results of a statistical analysis performed on the timeseries returned by the query.



	LOCATIONID	PARAMETERID	MEAN	MIN	DATEOFMIN	MAX	DATEOFMAX	SUM	STANDARDDEVIATION	COUNT	ORIGINAL	COMPLETED	CORRECT..	REL
1	DETO3IU	FMAT	-9.578294193043428	-11.5	2008-12-21 07:00:00.0	-7.222	2008-12-22 01:00:00.0	-162.83100128173828	1.1209471454521793	17	17	0	0	

Installing a FEWS JDBC Server

Windows

Step 1: Install an OC

Step 2: Delete the "Log4jConfig.xml" from the "OC" directory. When starting the application a new "Log4jConfig-JdbcServer.xml" file will be generated for logging.

Step 3: Make a new "<OC-Name>_JDBC.exe" and "<OC-Name>_JDBC.jpif" file in the \bin directory. The "<OC-Name>_JDBC.jpif" must contain the following information.

```
..\jre
-mx512m
-cp
$JARS_PATH$
nl.wldelft.fews.jdbc.FewsJdbcServer
<OC-Name>_JDBC
```

Step 4: Start the FewsJdbcServer by clicking on the <OC-Name>_JDBC.exe. The Server will start as an OC and synchronise its localDataStore with the Central Database using the synchprofiles of an OC.

Step 5: Stop the FewsJdbcServer by killing the application using the System Monitor. In the attachements an exe is provided that opens a console window. If this console window is stopped, the FEWS JDBC driver process is also stopped.

Install windows service
Follow the above listed steps to install and test the JDBC server. Finally stop the server and proceed with the next steps, based on the attached file JDBC service install.zip

Step 6: unzip the "JDBC service install.zip" to a directory at the same level as the bin and application directory, eg. like "service"
Step 7: replace in the file "run_installscript.bat" the BIN directory and the FEWs application name and directory
Step 8: run the batch file "run_installscript.bat"
Step 9: go to the services window and define the correct properties for the just installed service, like

- automatic startup
- correct user settings in login tab
- restart options after 5 minutes

Notice that the batch calls the file install_JDBC_Service.bat, that contains a list of the *.jar files in the bin directory. If these filenames have changed or the list has changed, this list should be updated. If not, running the service may not be successful. Also notice that your JAVA_HOME environment variable has been set and refers to your JRE directory. This JRE directory should not contain space characters in the name. If so, make a copy of your JRE to a directory with a name without space and set in the run_installscript.bat the JAVA_HOME variable to this new path.

Linux

Step 1: Install an OC

Step 2: Delete the "Log4jConfig.xml" from the "OC" directory. When starting the application a new "Log4jConfig-JdbcServer.xml" file will be generated for logging.

Step 3: Take the fews_jdbc.sh script file and place this one level higher than the \bin directory.

Step 4: Go to the directory where the ./fews_jdbc.sh script file is located and type **./fews_jdbc.sh <OC-Name>**.

Step 5: Stop the FEWS JDBC service by typing **exit** in the console window where the JDBC startup script was executed. An other option is to kill the process of the FEWS JDBC service.

Starting JDBC Service from FEWS Explorer

For debugging purpose it is possible to start the JDBC from the stand-alone FEWS Explorer. With the F12 key you get a list of debug options. Select "start embedded vjdbc server". The service will start and can be accessed from a database viewer.

Setting up connection in DbVisualizer

Step 1: Install DbVisualizer on your PC. Make sure it is not installed in a folder with spaces, such as "Program Files". When there is a space in the folder name, it will NOT work correctly. This is a DbVisualizer bug that can not be solved by FEWS.
Step 2: Copy the files "commons-logging-1.1.jar" and "vjdbc.jar" to a folder on your computer. These are the drivers used by DBVisualizer. Also this folder name should not contain any space characters (use the 8.3 format).
Step 3: Add a new JDBC driver to DBVisualiser:

- Start DbVisualizer
- Open the Tools menu and the Driver Manager
- Create a new driver and give it the name "vjdbc". Load the two jar files in the "User Specified" tab. * Close the Driver Manager Window.

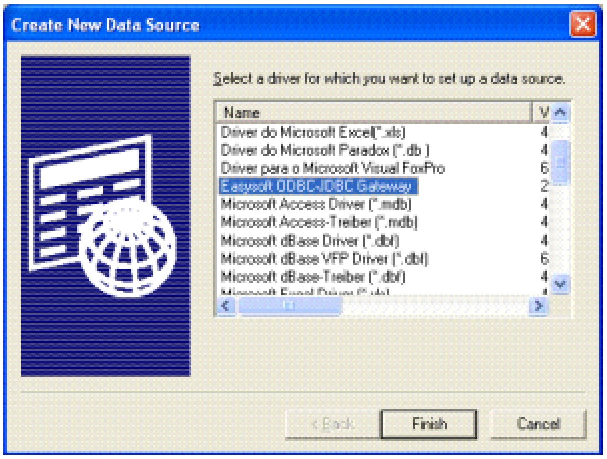
Step 4: Create a new Database Connection in DbVisualizer.

- Give it the Alias "<OC-Name> JDBC"
- Select the vjdbc driver
- Enter the database URL: "jdbc:vjdbc:rmi://<host>:2000/VJdbc,FewsDatabase" (under <host>, enter the machine where the fews jdbc application runs. You can get the IP adress by typing ipconfig in the command line of the Server). The number "2000" is the default port number, the correct port number is shown in the FEWS log file on the Server when it is started.

Setting up an ODBC-JDBC bridge

The FEWS JDBC Server has been tested with the Easysoft JDBC-ODBC bridge, this can be purchased. This allows the user to access the JDBC Server from other applications like Microsoft ACCESS that do only support ODBC. To use the JDBC driver with the ODBC-JDBC bridge, do the following:

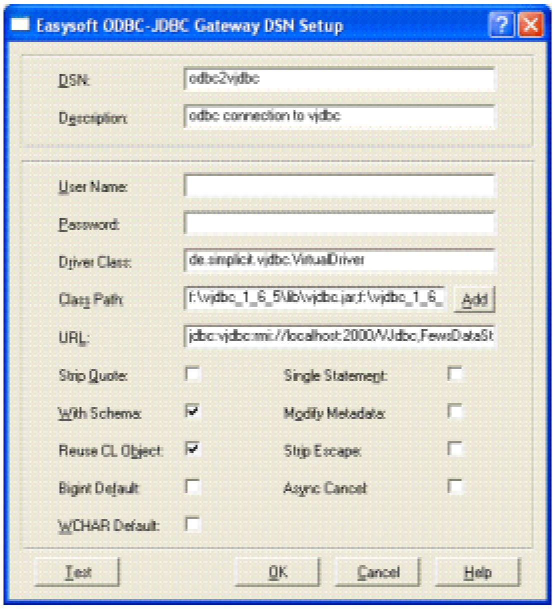
- Install the Easysoft JDBC-ODBC bridge
- Go to the Windows Start Menu -> Settings -> Control Panel -> Administrative Tools -> Data Sources (ODBC)
- Select the System DDS tab and add a new data source.



Create New Data Source

Select a driver for which you want to set up a data source.

Name	V
Driver do Microsoft Excel(*.xls)	4
Driver do Microsoft Paradox (*.db)	4
Driver para o Microsoft Visual FoxPro	6
Easysoft ODBC-JDBC Gateway	2
Microsoft Access Driver (*.mdb)	4
Microsoft Access-Treiber (*.mdb)	4
Microsoft dBase Driver (*.dbf)	4
Microsoft dBase VFP Driver (*.dbf)	6
Microsoft dBase-Treiber (*.dbf)	4



Easysoft ODBC-JDBC Gateway DSN Setup

DSN: odbc2vjdbc

Description: odbc connection to vjdbc

User Name:

Password:

Driver Class: de.simplicit.vjdbc.VirtualDriver

Class Path: f:\vjdbc_1_6_5\lib\vjdbc.jar;f:\vjdbc_1_6_

URL: jdbc:vjdbc:rmi://localhost:2000/VJdbc,FewsDataSt

Strip Quote: □ Single Statement: □

With Schema: ☑ Modify Metadata: □

Reuse CL Object: ☑ Strip Escape: □

Bigint Default: □ Async Cancel: □

WCHAR Default: □

- Make sure you add the vjdbc.jar and commons-logging-1.1.jar file to the classpath
- The url is: jdbc:vjdbc:rmi://<host>:2000/VJdbc,FewsDatabase (under <host>, enter the machine's IP or server name where the FEWS JDBC application runs)

When the FEWS JDBC application runs you can test the connection using the Test button.

JDBC server at client application

It is possible to have every client application (OC) to have also its own JDBC service.
Therefore you need to configure in systemConfigFiles\Explorer.xml a piServicePortRange. If configured, the OC automatically starts the JDBC service at the localhost.

```
<piServicePortRange start="2000" end="2001"/>
```

If a port number in the configured range is not available anymore, the service will not start.

You will see messages in the log panel like:

```
INFO - FewsJdbcServer.init - JdbcServer.Info: version: 2011.02, build: 34718 Apr3, type: stable, jre: 1.6.0_29,
mx: 518m, db: Firebird, region: ....
INFO - Log4JLogger.info - Starting RMI-Registry on port 2000
INFO - Log4JLogger.info - Binding remote object to 'VJdbc'
INFO - FewsJdbcServer.run - JDBC server started at <localhost>:2000
```

This can be used to make it possible that other programs use the JDBC service directly from the client PC.

The FEWS-JDBC Server can also be started embedded from the FEWS Explorer using F12 key.
In all cases that the service runs from the explorer the timezone settings are same as in the FEWS Explorer! This means that a real FEWS-JDBC Server and an embedded FEWS-JDBC Server started from for instance a FEWS Explorer with Dutch timezone settings, may give different timestamps on (the same) timeserie values with a shift up to 2 hours, depending on DLS conversion.

Example SQL queries

There are a number of SQL queries that can be used to retrieve data from the database. Only (read-only) statements are supported. Statements must be formatted as:

```
SELECT [DISTINCT] <select_expr> FROM TABLE_NAME [WHERE <where_condition>] [ORDER BY COLUMN_NAME [ASC |DESC]]

<select_expr>: (* | <COLUMN_NAME [, COLUMN_NAME, ...|, COLUMN_NAME, ...])

<where_condition>: COLUMN_NAME <operator> [AND <where_condition> OR <where_condition> LIKE <where_condition>]

<operator>: (= | <> | < | > <value>) | BETWEEN <value> AND <value>
```

For the Locations, Parameter and Filters table the SQL Query "Select * from <TableName>" is allowed. For the TimeSeries Table this query will return an error.

A valid query for the TimeSeries Table is as follows:

```
SELECT * from TimeSeries
WHERE moduleInstanceId = 'ImportSHEF'
AND parameterId = 'FMAT'
AND locationId = 'DETO3IL'
AND valueType = 'scalar'
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00'
AND timeSeriesType = 'external forecasting'
AND timeStep = 'unit=hour multiplier=6'
```

Or, when using filter id's:

```
SELECT time, value from TimeSeries
WHERE filterId = 'ImportSHEF'
AND parameterId = 'FMAT'
AND locationId = 'DETO3IL'
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00
```

> ⚠ ***Note*** *! When creating a query using the clause* ***time BETWEEN '2007-03-17 13:00:00' AND '2007-04-01 13:00:00'****, then it is good to realise that the* ***start time is used as system time*** *for retrieving the timeseries data. This could be important when retrieving 'external forecasting' data with an 'externalForecastTime' later than the start time.This will result in no data being returned.*

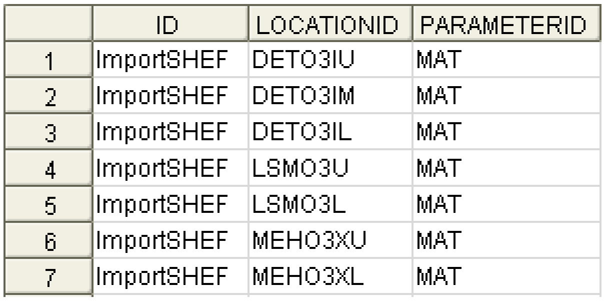
Example Locations queries

```
SELECT name, y,x from Locations ORDER BY name DESC
SELECT name, y,x from Locations WHERE X > '161000'
SELECT * from Locations where id = '46DP0003' OR id = '46DP0004'
SELECT name from Locations WHERE name <> 'Meerselsche Peel (WAM)'
SELECT id, name, y, x from Locations WHERE id LIKE '254%'
SELECT id, name, y, x from Locations WHERE name LIKE '%STUW%' or name LIKE '%Gemaal%'
```

Example Filters queries

Return all location and parameter combinations from a specific filter

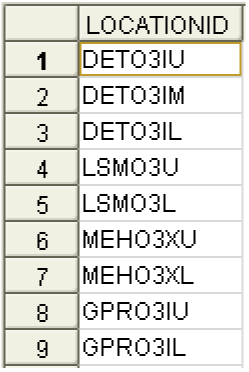
```
SELECT id, locationid, parameterid FROM filters WHERE id = 'ImportSHEF' ORDER BY location
```



	ID	LOCATIONID	PARAMETERID
1	ImportSHEF	DETO3IU	MAT
2	ImportSHEF	DETO3IM	MAT
3	ImportSHEF	DETO3IL	MAT
4	ImportSHEF	LSMO3U	MAT
5	ImportSHEF	LSMO3L	MAT
6	ImportSHEF	MEHO3XU	MAT
7	ImportSHEF	MEHO3XL	MAT

Return all locations from a specific filter

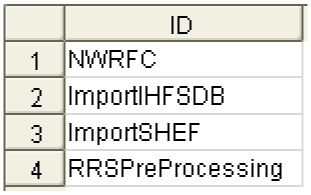
```
SELECT DISTINCT locationid FROM filters WHERE id = 'ImportSHEF'
```



	LOCATIONID
1	DETO3IU
2	DETO3IM
3	DETO3IL
4	LSMO3U
5	LSMO3L
6	MEHO3XU
7	MEHO3XL
8	GPRO3IU
9	GPRO3IL

Return a list of the main filter groups

```
SELECT DISTINCT id FROM filters WHERE issubfilter = false
```



	ID
1	NWRFC
2	ImportIHFSDB
3	ImportSHEF
4	RRSPreProcessing

Example TimeSeries queries

The Time series can be queried with or without the Filter ID. An example of a query without using the filter ID is:

```
SELECT * from TimeSeries
WHERE moduleInstanceId = 'ImportSHEF'
AND parameterId = 'FMAT'
AND locationId = 'DETO3IL'
AND valueType = 'scalar'
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00'
AND timeSeriesType = 'external forecasting'
AND timeStep = 'unit=hour multiplier=6'
AND Value BETWEEN '1.9' AND '2.0'
```

The same query with the use of a filter ID will be as follows:

```
SELECT * from TimeSeries
WHERE filterId = 'ImportSHEF'
AND parameterId = 'FMAT'
AND locationId = 'DETO3IL'
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00'
AND Value BETWEEN '1.9' AND '2.0'
```

> *Note on Time Series Queries:*
>
> - *All values are in the configured time zone of the JDBC application.*
> - *All unreliable values will not be returned in the query. The complete time step of unreliable values is missing in the returned recordset.*

Example TimeSeriesGraphs queries

The Time series can be extracted from the database as a graph (binary obejct) through the Timeseriesgraphs table. Queries with or without the Filter ID can be used, similar to the time series table. An example of a query with the use of a filter ID is:

```
SELECT * from TimeSeriesgraphs
WHERE filterId = 'ImportSHEF'
AND parameterId = 'FMAT'
AND locationId = 'DETO3IL'
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00'
```

By default the graphs have a size of 300 (width) * 200 (height) pixels. In the SQL query the width and height can also be fixed.

```
SELECT * from TimeSeriesgraphs
WHERE filterId = 'ImportSHEF'
AND parameterId = 'FMAT'
AND locationId = 'DETO3IL'
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00'
AND height = 100 AND width = 150
```

As from **201001** it is allowed to combine data from different locations and/or parameters into one graph by 'joining' them using OR-operators. Such a clause with OR-operators must be put in between brackets:

```
SELECT * from TimeSeriesgraphs
WHERE filterId = 'ImportSHEF'
AND parameterId = 'FMAT'
AND (locationId = 'DETO3IL' OR locationId = 'DETO3IL2')
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00'
AND height = 100 AND width = 150
```

```
SELECT * from TimeSeriesgraphs
WHERE filterId = 'ImportSHEF'
AND (parameterId = 'FMAT' OR parameterId = 'FMAT2')
AND locationId = 'DETO3IL'
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00'
AND height = 100 AND width = 150
```

```
SELECT * from TimeSeriesgraphs
WHERE filterId = 'ImportSHEF'
AND (parameterId = 'FMAT' OR parameterId = 'FMAT2' OR parameterId = 'FMAT3')
AND (locationId = 'DETO3IL' OR locationId = 'DETO3IL2' OR locationId = 'DETO3IL3')
AND time BETWEEN '2008-12-19 12:00:00'  AND '2008-12-23 12:00:00'
AND height = 100 AND width = 150
```

As from **201001** it is possible to optionally specify the time zone for the resulting graph; time clauses in the query remain to be specified in GMT.

Example of a graph query which will plot the data in GMT-1:

```
SELECT *
FROM TimeSeriesgraphs
WHERE filterId = 'Ott_ruw'
AND parameterId = 'H.diepte'
AND (locationId = '10.H.59'  OR locationId = '15.H.20')
AND time BETWEEN '2008-05-01 00:00:00'  AND '2008-05-01 10:00:00'
AND height = 500 AND width = 750 AND timezone='GMT-1';
```

Example code

Here follows some example code of how client applications can set up a connection to a JDBC server hosted by a FEWS OC.

Setting up a connection in JAVA

No special jars other than the ones provided by the JRE are required.

```
Connection con = DriverManager.getConnection("jdbc:vjdbc:rmi://localhost:2000/VJdbc,FewsDatabase");

//example get Locations
Statement stmt = con.createStatement();
ResultSet set = stmt.executeQuery("SELECT * from Locations");

while (set.next()) {
    System.out.print("Name:" + set.getString("name"));
    System.out.print(", Id:" + set.getString("Id"));
    System.out.print(", X:" + set.getString("x"));
    System.out.print(", y:" + set.getString("y") + "\n");
}
```

Miscellaneous

Using a different port number (available 200901)

By default the port number of the JDBC Server is 2000. It is possible to use a different port number when starting the application. In the global.properties a property can be added like this:

JdbcServerPort=2078

This will start the JDBC Server on port 2078.

Rolling Barrel

When the FEWS JDBC Server is started, the OC rolling barrel configuration will not be used. Instead the Rolling Barrel will run once a day at 02:00 GMT. After the FEWS Rolling Barrel, the compact Database script (only for MS ACCESS databases) will also be executed automatically.

(Java) JDBC Clients, Timezones and DayLightSaving conversion

FEWS stores timeseries with timestamp in GMT, without DayLightSaving (DLS) conversion.

JDBC Client applications like DBVisualizer adopt timezone settings from the (local) Operating System.
This means that data is converted (from FEWS GMT) to local timezone. When DLS conversion is active, a query on data from the night that DLS is switched (zomertijd to wintertijd, when clock is set back a hour) results in 'double' timeseries records between 2:00 and 3:00 AM.

The JVM for the JDBC client (like DBVisualizer) can be started with an extra commandline option, and forces timezone setting for the JVM rather than adopting it from the local OS. This commandline option looks like:
-Duser.timezone=GMT
or
-Duser.timezone=GMT+1
or
-Duser.timezone=GMT-5
and so on...

When starting DBVisualizers JVM with -Duser.timezone=GMT results are in GMT, without DLS conversion.

Another noticeable issue:
The FEWS-JDBC Server, started as described above, in a standalone manner, has a (hardcoded) timezonesetting of GMT.
The FEWS-JDBC Server can also be started embedded from the FEWS Explorer using F12 key. In the latter case it runs in the timezone set for the FEWS Explorer!
This means that a standalone FEWS-JDBC Server and an embedded FEWS-JDBC Server started from for instance a FEWS Explorer with Dutch timezone settings, may give different timestamps on (the same) timeserie values with a shift up to 2 hours, depending on DLS conversion.

Known issues

- java.sql.SQLException: java.net.MalformedURLException: no protocol....
  This is an exception that occurs due to a bug in DBVisualizer. Check whether DBVisualizer OR the vjdbc drivers are located in directories that contain spaces in their path. Move them to a directory path without spaces to solve this issue.