robbiblubber.org

JDBC Interop API

TECHNICAL REFERENCE

Version 0.1

Table of Contents

Introduction

The **JDBC Interop API** provides a named pipe ddp communication API for JDBC database access. This interface is used by the robbiblubber.org .NET JDBC data access implementation.

Technical Concepts

In the following, the main concepts of the JDBC Interop server implementation will be explained. This documentation focuses on the implementation of the Interop server and will not illustrate the mechanisms of named pipes, ddp, or encoding.

Interop Server

The Interop server is provided as a JAR-file named Robbiblubber.Extend.SQL.JDBC.Interop.jar. It requires an established set of named pipes and expects the pipe base name as command line argument.

JDBC Interop server uses the robbiblubber.org Named Pipe Library for Java to communicate with named pipes. It will access two named pipes by the same name and an appended index (/0 as inbound and /1 as outbound channel).

Passing Commands

The Interop server will listen for command strings which will be processed. After processing the Interop server will always send a response message.

Each valid command string must contain an **op** field, specifying the base command and may contain various command-specific data fields.

Each response message will contain a **success** field containing a Boolean value indicating if the command has been executed successfully. If the command has failed the response will also provide a **message** field containing an error message. Successfully completed commands may contain command-specific data fields.

Interop API Commands

Interop API Commands are invoked as ddp messages over named pipes. Parameters are passed to the method as ddp fields. The result is a ddp string.

The following sections describe the **Interop API Commands** in detail.

open

Opens a database connection.

```
[returns ddp(success, message)]
ddp(op=open;
    cstr=Connection_String;
    class=Class;
    uid=UserName;
    password=Password;)
```

Syntax

Parameters

`cstr`	(*string*) (*required*) JDBC connection string.
`class`	(*string*) Optionally defines a Java class to load before obtaining a connection from the driver manager.
`uid`	(*string*) Database user name if required or not specified in connection string.
`password`	(*string*) Database password if required or not specified in connection string.

Return Value

`success`	(*bool*) (*required*) Will be *true* if successful, otherwise *false*.
`message`	(*string*) Returns an error message if an error has occurred.

close

Closes the database connection.

```
[returns ddp(success, message)]
ddp(op=close;)
```

Syntax

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`message` (*string*) Returns an error message if an error has occurred.

stcreate

Creates a statement.

```
[returns ddp(success, h, message)]
ddp(op=stcreate;
    sql=SQL;)
```

Syntax

Parameters

`sql` (*string*) (*required*) SQL text.

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`h` (*string*) Returns a statement handle.

`message` (*string*) Returns an error message if an error has occurred.

bind

Binds a parameter to a statement.

```
[returns ddp(success, message)]
ddp(op=bind;
    h=Handle;
    name=Name;
    type=Type;
    value=Value;)
```

Syntax

Parameters

`h`	(*string*) (*required*) Statement handle.
`name`	(*string*) (*required*) Parameter name.
`type`	(*string*) (*required*) Parameter type.
`value`	(*string*) (*required*) Parameter value.

Return Value

`success`	(*bool*) (*required*) Will be *true* if successful, otherwise *false*.
`message`	(*string*) Returns an error message if an error has occurred.

execute

Executes a non-SQL statement.

```
[returns ddp(success, result, message)]
ddp(op=execute;
    h=Handle;)
```

Syntax

Parameters

`h` (*string*) (*required*) Statement handle.

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`result` (*int*) Returns the number of records affected.

`message` (*string*) Returns an error message if an error has occurred.

query

Executes a SQL statement and returns a result set.

```
[returns ddp(success, h, message)]
ddp(op=query;
    h=Handle;)
```

Syntax

Parameters

`h` (*string*) (*required*) Statement handle.

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`h` (*string*) Returns a result set handle.

`message` (*string*) Returns an error message if an error has occurred.

single

Executes a SQL statement and returns a single value.

```
[returns ddp(success, result, type, message)]
ddp(op=single;
    h=Handle;)
```

Syntax

Parameters

`h`	(*string*) (*required*) Statement handle.

Return Value

`success`	(*bool*) (*required*) Will be *true* if successful, otherwise *false*.
`result`	(*string*) Returns the result value.
`type`	(*string*) Returns the result data type.
`message`	(*string*) Returns an error message if an error has occurred.

stclose

Closes a statement.

```
[returns ddp(success, message)]
ddp(op=stclose;
    h=Handle;)
```

Syntax

Parameters

`h` (*string*) (*required*) Statement handle.

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`message` (*string*) Returns an error message if an error has occurred.

rmeta

Gets result set metadata.

```
[returns ddp(success, type[], name[], message)]
ddp(op=rmeta;
    h=Handle;)
```

Syntax

Parameters

`h` (*string*) (*required*) Result set handle.

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`eof` (*bool*) Will be *true* if the result set has been read to its end, otherwise *false*.

`type[]` (*string*) Returns a comma-separated array of data types for this result set.

`name[]` (*string*) Returns a comma-separated array of column names.

`message` (*string*) Returns an error message if an error has occurred.

read

Reads a record from a result set.

```
[returns ddp(success, eof, value[], message)]
ddp(op=read;
    h=Handle;)
```

Syntax

Parameters

`h` (*string*) (*required*) Result set handle.

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`eof` (*bool*) Will be *true* if the result set has been read to its end, otherwise *false*.

`value[]` (*string*) Returns a comma-separated array of values for this record.

`message` (*string*) Returns an error message if an error has occurred.

rclose

Closes a result set.

```
[returns ddp(success, message)]
ddp(op=rclose;
    h=Handle;)
```

Syntax

Parameters

`h` (*string*) (*required*) Result set handle.

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`message` (*string*) Returns an error message if an error has occurred.

trstart

Starts a transaction.

```
[returns ddp(success, message)]
ddp(op=tstart;
    il=Isolation_Level;)
```

Syntax

Parameters

`il`	(*int*) Desired isolation level:	0 (TRANSACTION_DEFAULT_BEHAVIOR) 1 (TRANSACTION_READ_UNCOMMITTED), 2 (TRANSACTION_READ_COMMITTED), 4 (TRANSACTION_REPEATABLE_READ), 8 (TRANSACTION_SERIALIZABLE).

Return Value

`success`	(*bool*) (*required*) Will be *true* if successful, otherwise *false*.
`message`	(*string*) Returns an error message if an error has occurred.

commit

Commits a transaction.

```
[returns ddp(success, message)]
ddp(op=commit;)
```

Syntax

Return Value

`success`	(*bool*) (*required*) Will be *true* if successful, otherwise *false*.
`message`	(*string*) Returns an error message if an error has occurred.

rollback

Rolls back a transaction.

```
[returns ddp(success, message)]
ddp(op=rollback;)
```

Syntax

Return Value

`success` (*bool*) (*required*) Will be *true* if successful, otherwise *false*.

`message` (*string*) Returns an error message if an error has occurred.

Dependencies

The JDBC Interop API uses the robbiblubber.org Utility Suite for Java and the robbiblubber.org Named Pipe Library for Java.

License

The JDBC Interop API and the JDBC Interop server are under MIT License. You can get a copy at https://bitbucket.org/robbiblubber/robbiblubber.extend.sql.jdbc.interop.java/downloads/License.txt.

Obtaining

The complete sources are available at
https://bitbucket.org/robbiblubber/robbiblubber.extend.sql.jdbc.interop.java/.