# Table of Contents

1. Introduction ................................................................. 3

2. Responses ......................................................................... 4
   2.1. Generic response ...................................................... 4
   2.2. SQL response ........................................................... 4
   2.3. Fetch response ......................................................... 4
   2.4. Slice response ........................................................ 5

3. Databases ......................................................................... 6
   3.1. Attach ................................................................. 6
       3.1.1. Identification .................................................. 6
       3.1.2. Attachment .................................................... 7
   3.2. Detach ................................................................. 8
   3.3. Create ................................................................. 8
   3.4. Drop ................................................................. 9
   3.5. Database information request ................................ 9
   3.6. Disconnect ......................................................... 10

4. Transactions ..................................................................... 11
   4.1. Start transaction ..................................................... 11
   4.2. Commit transaction ................................................ 11
   4.3. Rollback transaction .............................................. 11
   4.4. Commit retaining .................................................. 12
   4.5. Rollback retaining .................................................. 12
   4.6. Prepare .............................................................. 12
       4.6.1. Simple prepare .............................................. 12
       4.6.2. Prepare with message .................................. 13
   4.7. Transaction information request ............................ 13

5. Statements ..................................................................... 15
   5.1. Allocate .............................................................. 15
       5.1.1. Deviations for protocol version 11 .................. 15
   5.2. Free ................................................................. 15
       5.2.1. Deviations for protocol version 11 .................. 15
   5.3. Prepare .............................................................. 16
       5.3.1. Deviations for protocol version 11 .................. 16
   5.4. Describe ............................................................ 17
   5.5. Describe bind (input parameters) ............................ 17
   5.6. Execute ............................................................. 17
   5.7. Rows affected by query execution ......................... 19
   5.8. Fetch ............................................................... 19
   5.9. Set cursor name ................................................... 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.10. Information request</td>
<td>20</td>
</tr>
<tr>
<td>6. Blobs</td>
<td>22</td>
</tr>
<tr>
<td>6.1. Create/Open</td>
<td>22</td>
</tr>
<tr>
<td>6.2. Get segment</td>
<td>22</td>
</tr>
<tr>
<td>6.3. Put segment</td>
<td>23</td>
</tr>
<tr>
<td>6.4. Seek</td>
<td>23</td>
</tr>
<tr>
<td>6.5. Cancel</td>
<td>23</td>
</tr>
<tr>
<td>6.5.1. Deviations for protocol version 11</td>
<td>24</td>
</tr>
<tr>
<td>6.6. Close</td>
<td>24</td>
</tr>
<tr>
<td>6.6.1. Deviations for protocol version 11</td>
<td>24</td>
</tr>
<tr>
<td>7. Arrays</td>
<td>25</td>
</tr>
<tr>
<td>7.1. Get slice</td>
<td>25</td>
</tr>
<tr>
<td>7.2. Put slice</td>
<td>25</td>
</tr>
<tr>
<td>8. Services</td>
<td>27</td>
</tr>
<tr>
<td>8.1. Attach</td>
<td>27</td>
</tr>
<tr>
<td>8.2. Detach</td>
<td>27</td>
</tr>
<tr>
<td>8.3. Start</td>
<td>27</td>
</tr>
<tr>
<td>8.4. Query service</td>
<td>28</td>
</tr>
<tr>
<td>9. Events</td>
<td>29</td>
</tr>
<tr>
<td>9.1. Connection request</td>
<td>29</td>
</tr>
<tr>
<td>9.2. Queue events</td>
<td>30</td>
</tr>
<tr>
<td>9.3. Cancel events</td>
<td>30</td>
</tr>
<tr>
<td>10. Reading row data</td>
<td>31</td>
</tr>
<tr>
<td>Appendix A: External Data Representation (XDR)</td>
<td>32</td>
</tr>
<tr>
<td>Appendix B: Data types</td>
<td>33</td>
</tr>
<tr>
<td>Appendix C: Revision history</td>
<td>34</td>
</tr>
</tbody>
</table>
Chapter 1. Introduction

This document describes the Firebird wire protocol. Most of the information was obtained by studying the Firebird source code and implementing the wire protocol in the Firebird .NET provider and Jaybird (Firebird JDBC driver).

The protocol is described in the form of the message sent by the client and received from the server. The described protocol is Firebird/Interbase protocol version 10. Earlier (Interbase) versions of the protocol are not in scope for this document. Changes in later protocol versions are described in notes below the description of the relevant version 10 message (currently only version 11 is partially described).

This document is not complete. It is advisable to consult the Interbase 6.0 API Guide for additional information on subjects like parsing the status vector, information request items, and the meaning of operations.

Unless otherwise indicated, a client request must be flushed to the server for processing. For some operations the flush can be deferred, so it is sent together with a different operation. Versions 11 and higher of the wire protocol explicitly support (or even require) deferring of operations, including deferring the read of the response.
Chapter 2. Responses

The wire protocol has a limited set of responses. Some operations have a specific response, which is described together with the operation. Most operations however use one (or more) of the responses described in this section. The meaning and content depend on the operation that initiated the response.

2.1. Generic response

Int32  
Operation code

If operation equals op_response:

Int32  
Object handle

Int64  
Object ID

Buffer  
Data (meaning depends on the operation).

Byte[]  
Status vector

Information about parsing the status vector can be found in the Interbase 6.0 API Guide in the documentation set. It might also be advantageous to look at the sources of the Firebird .NET provider or Jaybird.

2.2. SQL response

Int32  
Operation code

If operation equals op_sql_response:

Int32  
Message count

Buffer  
Response data (meaning depends on the operation).

2.3. Fetch response
Int32
Operation code

If operation equals op_fetch_response:

Int32
Status

A value of 0 is the success value.

End of cursor is indicated with a non-zero status.

A status with value of 100 means that there are no more rows.

Int32
Count of rows following response

The data rows are not in a buffer as described in Data types, but as a sequence of data rows, see Reading row data.

2.4. Slice response

Int32
Operation code

If operation equals op_slice:

Int32
Slice length

Int32
Slice length

Buffer
Slice data
Chapter 3. Databases

3.1. Attach

Attachments to a database are done in two steps, first identification (connect) to the server, then attachment to a database.

3.1.1. Identification

Performs the initial handshake and protocol selection.

**Client**

Int32
  - Operation code (op_connect)

Int32
  - Operation code (op_attach)

Int32
  - Version (CONNECT_VERSION2)

Int32
  - Architecture type (eg arch_generic = 1).

**String**

  - Database path or alias

Int32
  - Count of protocol versions understood (eg 1)

**Buffer**

  - User identification

The next block of data declares the protocol(s) that the client is willing or able to support. It should be sent as many times as protocols are supported (and specified as *Count of protocol versions understood*), values depend on the protocol.

Int32
  - Protocol version (PROTOCOL_VERSION10)

Int32
  - Architecture type (eg arch_generic = 1)

Int32
  - Minimum type (eg ptype_rpc = 2)
Int32
    Maximum type (eg ptype_batch_send = 3)

Int32
    Preference weight (eg 2)

**Server**

Int32
    Operation code

If operation equals op_accept:

Int32
    Protocol version number accepted by server

Int32
    Architecture for protocol

Int32
    Minimum type

### 3.1.2. Attachment

Attaches to a database.

**Client**

Int32
    Operation code (op_attach)

Int32
    Database object id (0)

String
    Database path or alias

Buffer
    Database parameter buffer

*Table 1. Example of parameters sent in the DPB*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>isc_dpb_version1</td>
<td>Version (must be first item!)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isc_dpb_dummy_packet_interval</td>
<td>Dummy packet interval</td>
<td>120</td>
<td>*</td>
</tr>
<tr>
<td>isc_dpb_sql_dialect</td>
<td>SQL dialect</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value</td>
<td>Optional</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>isc_dpb_lc_ctype</td>
<td>Character set</td>
<td>UTF8</td>
<td></td>
</tr>
<tr>
<td>isc_dpb_sql_role_name</td>
<td>User role</td>
<td>RDB$ADMIN*</td>
<td></td>
</tr>
<tr>
<td>isc_dpb_connect_timeout</td>
<td>Connection timeout</td>
<td>10</td>
<td>*</td>
</tr>
<tr>
<td>isc_dpb_user_name</td>
<td>User name</td>
<td>SYSDBA</td>
<td></td>
</tr>
<tr>
<td>isc_dpb_password</td>
<td>User password</td>
<td>masterkey</td>
<td></td>
</tr>
</tbody>
</table>

**Server**

**Generic response** — where the *Object handle* is the database handle.

### 3.2. Detach

Detaches from the database. After detach the connection is still open, to disconnect use **Disconnect** (op_disconnect).

**Client**

Int32

Operation code (op_detach)

Int32

Database handle

**Server**

**Generic response**

### 3.3. Create

Create a database. Create is similar to **Attachment** (op_attach).

**Client**

Int32

Operation code (op_create)

Int32

Database object id (0)

String

Database path

**Buffer**

Database parameter buffer
3.4. Drop

Drops the currently attached database.

Client

Int32

Operation code (op_drop_database)

Int32

Database handle

Server

Generic response

3.5. Database information request

Requests database or server information.

Client

Int32

Operation code (op_info_database)

Int32

Database handle

Int32

Incarnation of object (0)

Buffer

Requested information items

Int32

Length of buffer available for receiving response (too small may lead to receiving a truncated buffer, which necessitates requesting information again).

The buffer in the response is sized to the actual length of the response (upto the declared available length), so specifying a larger than necessary size does not inflate the response on the wire.

Server

Generic response — where Data holds the requested information.
3.6. Disconnect

Client

Int32
   Operation code (op_disconnect)

No response, remote socket close.
Chapter 4. Transactions

4.1. Start transaction

Starts a transaction with the transaction options specified in the transaction parameter buffer.

Client

Int32
   Operation code (op_transaction)

Int32
   Database handle

Buffer
   Transaction parameter buffer

Server

Generic response — where Object handle is the new transaction handle.

4.2. Commit transaction

Commits an active or prepared transaction.

Client

Int32
   Operation code (op_commit)

Int32
   Transaction handle

Server

Generic response

4.3. Rollback transaction

Rolls back an active or prepared transaction.

Client

Int32
   Operation code (op_rollback)
4.4. Commit retaining

Commits an active or prepared transaction, retaining the transaction context.

Client

Int32

Operation code (op_commit_retaining)

Int32

Transaction handle

Server

Generic response

4.5. Rollback retaining

Rolls back an active or prepared transaction, retaining the transaction context.

Client

Int32

Operation code (op_rollback_retaining)

Int32

Transaction handle

Server

Generic response

4.6. Prepare

Performs the first stage of a two-phase commit. After prepare a transaction is in-limbo until committed or rolled back.

4.6.1. Simple prepare

Client
4.6.2. Prepare with message

Associates a message (byte data) with the prepared transaction. This information is stored in RDB$TRANSACTIONS and can be used for recovery purposes.

Client

- Int32: Operation code (op_prepare)
- Int32: Transaction handle

Server

Generic response

4.7. Transaction information request

This is similar to Database information request.

Client

- Int32: Operation code (op_transaction_info)
- Int32: Database handle
- Int32: Incarnation of object (0)

Buffer
- Requested information items
Int32

Length of buffer available for receiving response (too small may lead to receiving truncated buffer).

**Generic response** — where *Data* holds the requested information.
Chapter 5. Statements

5.1. Allocate

Allocates a statement handle on the server.

**Client**

Int32

- Operation code (op_allocate_statement)

Int32

- Database handle

**Server**

Generic response — where *Object handle* is the allocated statement handle.

5.1.1. Deviations for protocol version 11

An allocate can only be sent together with a Prepare operation.

5.2. Free

Frees resources held by the statement.

**Client**

Int32

- Operation code (op_free_statement)

Int32

- Statement handle

Int32

- **Option** | **Description**
  - DSQL_close | Closes the cursor opened after statement execute.
  - DSQL_drop | Releases the statement handle.

**Server**

Generic response

5.2.1. Deviations for protocol version 11

Request flushing and response processing must be deferred.
## 5.3. Prepare

### Client

Int32
- Operation code (op_prepare_statement)

Int32
- Transaction handle

Int32
- Statement handle

Int32
- SQL dialect

String
- Statement to be prepared

Buffer
- Describe and describe bind information items

*Example of requested information items*
- isc_info_sql_select
- isc_info_sql_describe_vars
- isc_info_sql_sqllda_seq
- isc_info_sql_type
- isc_info_sql_sub_type
- isc_info_sql_length
- isc_info_sql_scale
- isc_info_sql_field
- isc_info_sql_relation

Int32
- Target buffer length (32768)

### Server

**Generic response** — where Data holds the statement description (matching the requested information items)

### 5.3.1. Deviations for protocol version 11

The statement handle can no longer be allocated separately. The initial *Allocate* operation **must** be sent together with the first prepare operation. When allocating and preparing together, the value of
the statement handle of the prepare must be 0xFFFF (invalid object handle). The responses must be processed in order: first allocate response, then prepare response.

Once a statement handle has been allocated, it can be reused by sending a prepare with the obtained statement handle.

5.4. Describe

Describe of output parameters of a query is done using the statements information request message

Example of requested information items:

- `isc_info_sql_select`
- `isc_info_sql_describe_vars`
- `isc_info_sql_sqlda_seq`
- `isc_info_sql_type`
- `isc_info_sql_sub_type`
- `isc_info_sql_length`
- `isc_info_sql_scale`
- `isc_info_sql_field`
- `isc_info_sql_relation`

5.5. Describe bind (input parameters)

Describe of input parameters of a query is done using the statements information request message

Example of requested information items:

- `isc_info_sql_select`
- `isc_info_sql_describe_vars`
- `isc_info_sql_sqlda_seq`
- `isc_info_sql_type`
- `isc_info_sql_sub_type`
- `isc_info_sql_length`
- `isc_info_sql_scale`
- `isc_info_sql_field`
- `isc_info_sql_relation`

5.6. Execute

Client
<table>
<thead>
<tr>
<th>Operation</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>op_execute</td>
<td>DDL and DML statements.</td>
</tr>
<tr>
<td>op_execute2</td>
<td>Stored procedures.</td>
</tr>
</tbody>
</table>

- **Int32**: Operation code

- **Int32**: Statement handle

- **Int32**: Transaction handle

If the statement has input parameters:

- **Buffer**: Parameters in BLR format

- **Int32**: Message number (0) ?

- **Int32**: Number of messages (1) ?

If not statement has no input parameters:

- **Buffer**: Empty (length only 0)

- **Int32**: Message number (0) ?

- **Int32**: Number of messages (0) ?

If the statement is a stored procedure and there are output parameters:

- **Buffer**: Output parameters in BLR format

- **Int32**: Output message number (0) ?

**Server**
Int32

Operation code

If operation equals op_sql_response:

SQL response

if not:

Generic response

5.7. Rows affected by query execution

Obtain the rows affected by a query is done using the statements information request message

List of requested information items

• isc_info_sql_records

5.8. Fetch

Client

Int32

Operation code (op_fetch)

Int32

Statement handle

Buffer

Output parameters in BLR format

Int32

Message number

Int32

Message count/Fetch size (200)

Server

Int32

Operation code

If operation equals op_fetch_response:

Fetch response.

If not:

Generic response.
### 5.9. Set cursor name

**Client**

- **Int32**
  - Operation code (op_set_cursor)
- **Int32**
  - Statement handle
- **String**
  - Cursor name (null terminated)
- **Int32**
  - Cursor type (0).

  ![]() Reserved for future use

**Server**

**Generic response**

### 5.10. Information request

This is similar to Database information request.

**Client**

- **Int32**
  - Operation code (op_info_sql)
- **Int32**
  - Statement handle
- **Int32**
  - Incarnation of object (0)

**Buffer**

- Requested information items

**Server**

**Generic response** — where Data holds the requested information.
Information about how to parse the information buffer sent by the Firebird server can be found in the Interbase 6.0 documentation set.
Chapter 6. Blobs

6.1. Create/Open

Client

Int32
  Operation code

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>op_create_blob</td>
<td>Creates a new blob</td>
</tr>
<tr>
<td>op_create_blob2</td>
<td>Creates a new blob with a blob parameter buffer</td>
</tr>
<tr>
<td>op_open_blob</td>
<td>Opens an existing blob</td>
</tr>
<tr>
<td>op_open_blob2</td>
<td>Opens an existing blob with a blob parameter buffer</td>
</tr>
</tbody>
</table>

Buffer

Blob parameter buffer (not allowed with op_create_blob and op_open_blob, required with op_create_blob2 and op_open_blob2)

Int32
  Transaction handle

Int64
  Blob ID

Server

Generic response — where:

a. Object handle is the blob handle

b. Object id is the blob id (only for op_create_blob / op_create_blob2, garbage for op_open_blob / op_open_blob2)

6.2. Get segment

Client

Int32
  Operation code (op_get_segment)

Int32
  Blob handle

Int32
  Segment length (max length = 32768)
6.3. Put segment

Client

Int32
   Operation code (op_batch_segments)

Int32
   Blob handle

Buffer
   Blob Segments

Server

Generic response — where Data is the blob segment.

6.4. Seek

Client

Int32
   Operation code (op_seek_blob)

Int32
   Blob handle

Int32
   Seek mode (0)

Int32
   Offset

Server

Generic response — where Object handle is the current position.

6.5. Cancel

Cancels and invalidates the blob handle. If this was a newly created blob, the blob is disposed.
Client

Int32
   Operation code (op_cancel_blob)

Int32
   Blob handle

Server

Generic response — no useful information in response

6.5.1. Deviations for protocol version 11

Request flushing and response processing must be deferred.

6.6. Close

Closes and invalidates the blob handle.

Client

Int32
   Operation code (op_close_blob)

Int32
   Blob handle

Server

Generic response — no useful information in response

6.6.1. Deviations for protocol version 11

Request flushing and response processing must be deferred.
Chapter 7. Arrays

7.1. Get slice

Client

Int32
  Operation code (op_get_slice)

Int32
  Transaction handle

Int64
  Array handle

Int32
  Slice length

Buffer
  Slice descriptor (SDL)

String
  Slice parameters (Always an empty string)

Buffer
  Slice (Always empty)

Server

Slice response

7.2. Put slice

Client

Int32
  Operation code (op_put_slice)

Int32
  transaction handle

Int64
  Array handle (0)

Int32
  Slice length
Buffer
  Slice descriptor (SDL)

String
  Slice parameters (Always an empty string)

Int32
  Slice length

Buffer
  Slice data

Server

Generic response — where Object id is the array handle.
Chapter 8. Services

8.1. Attach

Client

Int32
  Operation code (op_service_attach)

Int32
  Database object ID (0)

String
  Service name
    For local connections: service_mgr
    For remote connections: hostname:service_mgr

Buffer
  Service parameter buffer

Server

Generic response — where Object handle is the services manager attachment handle.

8.2. Detach

Client

Int32
  Operation code (op_service_detach)

Int32
  Services manager attachment handle

Server

Generic response

8.3. Start

Client

Int32
  Operation code (op_service_start)
Chapter 8. Services

8.4. Query service

Client

Int32
   Services manager attachment handle

Int32
   Incarnation of object (0)

Buffer
   Services parameter buffer

Server

Generic response

— where Data contains the requested information.
Chapter 9. Events

9.1. Connection request

**Client**

Int32

Operation code (op_connect_request)

Int32

Connection type (P_REQ_async)

Int32

Partner identification (0)

**Server**

Int32

Attachment handle

Int16

Port number

This is part of the sockaddr_in structure.

It is not in XDR format

Int16

Socket family

This is part of the sockaddr_in structure.

It is not in XDR format

Byte[4]

IP Address

This is part of the sockaddr_in structure.

It is not in XDR format

Byte[8]

Zeroes

This is part of the sockaddr_in structure.

It is not in XDR format
Chapter 9. Events

9.2. Queue events

**Client**

Int32

Operation code (op_que_events)

Int32

Database handle

**Buffer**

Events parameter buffer

Int32

Ast function address

Int32

Ast parameters function address

Int32

Local event id

**Server**

**Generic response** — where Object handle holds the remote event id.

9.3. Cancel events

**Client**

Int32

Operation code (op_cancel_events)

Int32

Database handle

Int32

Local event id

**Server**

**Generic response**
Chapter 10. Reading row data

TODO: Processing row data
Appendix A: External Data Representation (XDR)

The Firebird wire protocol uses XDR for exchange messages between client and server.
Appendix B: Data types

Int32
   Integer 32-bits

Int64
   Integer 64-bits

Buffer
   Type  Description
   Int32  Length
   Byte[] Buffer data

Byte[]
   An array of bytes

String
   A text string (Read/Written as a buffer)
Appendix C: Revision history

Revision History

0.1 31 May 2004  First draft for review.

0.2 02 Jun 2004  Fixed issues reported by Paul Vinkenoog.

0.3 03 Jun 2004  Added new subsections to the Statements section.

0.4 05 Jun 2004  Fixed issues reported by Paul Vinkenoog.

0.5 06 Jun 2004  Fixed issues reported by Paul Vinkenoog.

0.6 07 Jun 2004  Added events system documentation.

0.7 16 Jun 2004  Modified document ID to wireprotocol.

0.8 17 Jun 2004  Added two new segmented lists.

0.9 18 Jun 2004  • Improved segmentedlist usage.
                  • Fixed rendering of important tags.

0.1 19 Jun 2004  Changed rendering of important tags using Paul Vinkenoog fix.

0.1 20 Jun 2004  • Added new segmentedlist.
                  • Updated Statements.Prepare documentation.
                  • Updated Statements.Execute documentation.
                  • Updated Blobs.GetSegment documentation.
                  • Updated Blobs.Seek documentation.

0.1 21 Jun 2004  Updated services information.

0.1 13 Sep 2014  Updated and expanded protocol information

0.1 04 Aug 2020  M Conversion to Asciidoc, minor copy-editing

0.1 2020 R