# 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
# Appendix A: External Data Representation (XDR)

## 11. Reading row data

### 10. Events

- 10.1. Connection request
- 10.2. Queue events
- 10.3. Cancel events

### 11. Reading row data
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 (e.g. arch_generic = 1).

String

Database path or alias

Int32

Count of protocol versions understood (e.g. 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 (e.g. arch_generic = 1)

Int32

Minimum type (e.g. ptype_rpc = 2)
Int32
  Maximum type (e.g. ptype_batch_send = 3)

Int32
  Preference weight (e.g. 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><strong>Parameter</strong></td>
<td><strong>Description</strong></td>
<td><strong>Value</strong></td>
<td><strong>Optional</strong></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
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.

4.7. Transaction information request

This is similar to Database information request.
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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSQL_close</td>
<td>Closes the cursor opened after statement execute.</td>
</tr>
<tr>
<td>DSQL_drop</td>
<td>Releases the statement handle.</td>
</tr>
</tbody>
</table>

**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_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

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 statement information request message

*Example of requested information items*

- `isc_info_sql_select`
- `isc_info_sql_describeVars`
- `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 statement information request message

*Example of requested information items*

- `isc_info_sql_select`
- `isc_info_sql_describeVars`
- `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*
Int32
  Operation code

<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
  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) ??

Buffer
  Parameter values

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 statement 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:
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

Int32
Requested information items buffer length

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)
Int32
  Data segment (0)

Server

Generic response — where Data is the blob segment.

6.3. Put segment

Client

Int32
  Operation code (op_batch_segments)

Int32
  Blob handle

Buffer
  Blob Segments

Server

Generic response

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.
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.

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. Batches

Statement batches were introduced in protocol v16 (Firebird 4.0).

8.1. Create

Client

Int32
  Operation code (op_batch_create)

Int32
  Statement handle

Buffer
  BLR format of batch messages

Int32
  Message length

Buffer
  Batch parameters buffer

Server

Generic response

8.2. Send messages

Client

Int32
  Operation code (op_batch_msg)

Int32
  Statement handle

Int32
  Number of messages

Buffer
  Batched values (formatted message repeats 'Number of messages' times)

Server

Generic response
8.3. Execute batch

**Client**

Int32
Operation code (op_batch_exec)

Int32
Statement handle

Int32
Transaction handle

**Server**

Int32
Operation code

If operation equals op_batch_cs:

**Batch completion state**

Int32
Statement handle

Int32
Total records count

Int32
Number of update counters (records updated per each message)

Int32
Number of per-message error blocks (message number in batch and status vector of an error processing it)

Int32
Number of simplified per-message error blocks (message number in batch without status vector)

**Buffer**

Update counters (records updated per each message), array of Int32, length is equal to "Number of update counters" field in packet.

**Buffer**

Detailed info about errors in batch (for each error server sends number of message (Int32) and status vector in standard way (exactly like in op_response). Number of such pairs is equal to "Number of per-message error blocks" field in packet.
Buffer

Simplified error blocks (for each error server sends number of message (Int32) w/o status vector). Used when too many errors took place. Number of elements is equal to "Number of simplified per-message error blocks" field in packet.

Otherwise:

Generic response

8.4. Release batch

Client

Int32
Operation code (op_batch_rls)

Int32
Statement handle

Server

Generic response

8.5. Cancel batch

Client

Int32
Operation code (op_batch_cancel)

Int32
Statement handle

Server

Generic response

8.6. Sync batch

Introduced in v17 (Firebird 4.0.2).

Client

Int32
Operation code (op_batch_sync)
8.7. Set default blob parameters

Client

Int32
  Operation code (op_batch_set_bpb)

Int32
  Statement handle

Buffer
  Default BLOB parameters buffer

Server

Generic response

8.8. Register existing blob

Client

Int32
  Operation code (op_batch_regblob)

Int32
  Statement handle

Int64
  Existing BLOB ID

Int64
  Batch temporal BLOB ID

Server

Generic response

8.9. Stream of BLOB data

Client

Int32
  Operation code (op_batch_blob_stream)
Int32
Statement handle

Buffer
BLOB stream

This stream is a sequence of blob records. Each blob records contains:

Int32
Record length

The following three fields are called BLOB header

Int64
Batch temporal BLOB ID

Int32
BLOB size

Int32
BLOB parameters buffer size

Buffer
BLOB parameters buffer

Buffer
BLOB data (length - BLOB size bytes)

BLOB headers and records in a stream need not match, i.e. one record may contain many BLOBs and BLOB may stretch from one record to next.

Server

Generic response
Chapter 9. Services

9.1. 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.

9.2. Detach

Client

Int32
  Operation code (op_service_detach)

Int32
  Services manager attachment handle

Server

Generic response

9.3. Start

Client

Int32
  Operation code (op_service_start)
9.4. Query service

**Client**

Int32
Service manager attachment handle

Int32
Incarnation of object (0)

Buffer
Services parameter buffer

**Server**

Generic response

---

**Server**

Generic response — where Data contains the requested information.
Chapter 10. Events

10.1. 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
10.2. 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.

10.3. Cancel events

Client

Int32
Operation code (op_cancel_events)

Int32
Database handle

Int32
Local event id

Server

Generic response
Chapter 11. 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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>31 May 2004</td>
<td>First draft for review.</td>
</tr>
<tr>
<td>0.2</td>
<td>02 Jun 2004</td>
<td>Fixed issues reported by Paul Vinkenoog.</td>
</tr>
<tr>
<td>0.3</td>
<td>03 Jun 2004</td>
<td>Added new subsections to the Statements section.</td>
</tr>
<tr>
<td>0.4</td>
<td>05 Jun 2004</td>
<td>Fixed issues reported by Paul Vinkenoog.</td>
</tr>
<tr>
<td>0.5</td>
<td>06 Jun 2004</td>
<td>Fixed issues reported by Paul Vinkenoog.</td>
</tr>
<tr>
<td>0.6</td>
<td>07 Jun 2004</td>
<td>Added events system documentation.</td>
</tr>
<tr>
<td>0.7</td>
<td>16 Jun 2004</td>
<td>Modified document ID to wireprotocol.</td>
</tr>
<tr>
<td>0.8</td>
<td>17 Jun 2004</td>
<td>Added two new segmented lists.</td>
</tr>
<tr>
<td>0.9</td>
<td>18 Jun 2004</td>
<td>• Improved segmentedlist usage.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed rendering of important tags.</td>
</tr>
<tr>
<td>0.1</td>
<td>19 Jun 2004</td>
<td>Changed rendering of important tags using Paul Vinkenoog fix.</td>
</tr>
<tr>
<td>0.1</td>
<td>20 Jun 2004</td>
<td>• Added new segmentedlist.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Statements.Prepare documentation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Statements.Execute documentation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Blobs.GetSegment documentation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Blobs.Seek documentation.</td>
</tr>
<tr>
<td>0.1</td>
<td>21 Jun 2004</td>
<td>Updated services information.</td>
</tr>
<tr>
<td>0.1</td>
<td>13 Sep 2014</td>
<td>Updated and expanded protocol information</td>
</tr>
<tr>
<td>0.1</td>
<td>04 Aug 2020</td>
<td>M Conversion to AsciiDoc, minor copy-editing</td>
</tr>
<tr>
<td>0.1</td>
<td>26 Dec 2021</td>
<td>AP Document batch execution</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
</tr>
</tbody>
</table>

39