TIMESTAMP WITH TIME ZONE

A timestamp represents a fixed point in time (an event) and is the same event regardless of whether it is expressed in local time, UTC or some other time zone. The Firebird TIMESTAMP data type is used to record such an event in a database. Prior to FB4, the database designer had to state whether this was in a fixed time zone, such as GMT, or in some local time dependent on where the database was deployed. From FB4 onwards. A new TIMESTAMP WITH TIME ZONE data type is available.

A TIMESTAMP WITH TIME ZONE data type is always saved in the database as a GMT timestamp plus a time zone id, regardless of whether it was originally input as GMT or as a local time in the context of some time zone.

Converting a timestamp to a standard time zone (e.g. GMT) makes sense before storing it in a database. The database is then portable between time zones and timestamps can be readily compared and ordered. Indeed, the only reason you would want to include time zone information in the database is so that the database records the original way that the timestamp was expressed and so that it can be rendered in its original format, if so desired.

Firebird requires access to a time zone database in order to perform the conversion to and from GMT (see README.timezone_database).

Entering a Timestamp with Time Zone Value

The date and time part of a TIMESTAMP WITH TIME ZONE are the same as a TIMESTAMP without a time zone. In FB4, a timestamp's time zone is separated from the timestamp by white space and can be given in three alternative formats:

1. As a time zone database name e.g. Europe/London or EST5EDT.
2. As an offset to GMT
3. As an alias to an offset to GMT (e.g. EST or Eastern Standard Time). *(These are also in the time zone database, but have no daylight savings time info with them)*.

In all cases, it is possible to convert the timestamp to GMT and, indeed this is done when saving the timestamp to the database.

However, consider the following:

- 01.JUL.2020 7:00 America/New_York
- 01.JUL.2020 7:00 EST5EDT
- 01.JUL.2020 06:00 EST
- 01.JUL.2020 07:00 -04:00

These are all different ways of entering the same timestamp. On the other hand, a reader only knows that they are the same if they also know that Daylight Savings Time is in effect for America/New_York time zone on 1st July and that EDT is 4 hours behind GMT. While it may be argued that any kid knows this – is this true for some less well known parts of the world.

When it comes to rendering a TIMESTAMP WITH TIME ZONE in a report, simply reflecting back the original input data is probably not a useful approach.

Rendering a TIMESTAMP WITH TIME ZONE

The problem is that while Firebird is very flexible when it comes for timestamp with time zone data entry – which is a good thing – simply rendering a timestamp in exactly the same format as it was entered is not necessarily a good thing if timestamps in many different formats are rendered on the same report. The reader may well struggle to compare the results.

Firebird's default is to return a timestamp in the same format as it was entered. However, the report designer has to think carefully about whether the original format should be used or, alternatively, whether all timestamps should be rendered in a common format in order to ensure both readability and comparison without the reader having to keep a time zone database in their head.

Note that not all input formats convey the same information. In particular, the time zone offset form is independent of time zone name and whether or not daylight savings time is in effect. For example “01.JUL.2020 07:00 -04:00” is EDT in New York, but could be AST in New Brunswick in the Canada/Atlantic time zone.

Furthermore, considering that:

- A Time Zone Name, such as America/New_York tells you the offset from GMT only when looking up the time zone database using the timestamp (including date) as the key. It is thus “difficult” for the human reader to compare timestamps in different time zones when using this format.
- A time zone offset alias, such as EST, is independent of daylight savings time. However, it is still necessarily for the reader to know how different time zone aliases relate to GMT when comparing timestamps in different time zones.

Either GMT, or local time plus a time zone offset are probably the only realistic common formats for rendering a time zone if unconstrained input for time zone identity is permitted and human readability is desired.

Assuming that the above is true then when decoding a time zone with timestamp, the client needs to know the time part in either local time or GMT plus the offset from GMT. The timestamp can then be rendered in local time plus the offset from GMT, or as GMT. A human reader can then readily compare them. The time zone name is additional information and only of interest when you want to show how the timestamp was originally entered.

IBX thus provides three alternative ways of rendering a TIMESTAMP WITH TIME ZONE as a string:

1. Default: Timestamp in local time with a time zone offset from GMT.
2. Timestamp in GMT (no time zone offset).
3. Timestamp with time zone as originally input.

Reading a TIMESTAMP WITH TIME ZONE From a Firebird Database

FB4 provides two TIMESTAMP WITH TIME ZONE formats for the over the wire protocol:

- In the standard format (which was the only format available in FB4 beta 1), an encoded date, GMT time and time zone id is provided.
- In the extended format (available only in the FB4 development version so far), an encoded date, GMT time, offset from GMT and time zone id is provided.

The rationale for the extended format was to allow for situations where the time zone database was not available to the client and hence all time zone database computations have to be performed by the server. However, it also gives exactly the right information for rendering a timestamp in a common time zone offset format and without any further client side work.

The low level Firebird API provides two API calls that can be used to decode the data received from the server.

- The IUtil.decodeTimeStampTzEx API call is used to decode the extended format and returns the date and local time plus the time zone name as input (if a time zone offset was input then the offset is returned as a text string). IBX uses this to get the local time and time zone name. It also extracts the offset and time zone id from the received data.
- IUtil.decodeTimeStampTz is used to decode the standard format. This also returns the date and local time plus the time zone name as input. In order to determine the offset, IBX uses the (TIMESTAMP WITHOUT TIME ZONE) API calls IUtil.decodeDate and IUtil.decodeTime to get the timestamp in GMT. The offset can then be derived. This exploits the fact that the time zone is an extension to the original timestamp over the wire encoding and that the time part returned “over the wire” is in GMT. The time zone id may also be extracted from the received data.

IBX thus returns the same information regardless which format is used for the over the wire protocol and this includes the all important offset from GMT. It can thus render each of the three formats identified above.