Livebase 6.3.0
Upcoming Release Notice

Rome, May 22nd 2026

On **May 26, 2026**, FHOSTER will release a new version of Livebase, 6.3.0. This new version will generate cloudlets whose GraphQL API will handle Date, Time, and Datetime attributes differently than previous versions, and will not be fully backward compatible.

This may require some specific changes to the code using the Livebase cloudlet API **if these cloudlets are rebuilt** after the release of version 6.3.0 of the platform.

In this document, we describe the differences that will be introduced in the GraphQL API of cloudlets generated by version 6.3.0 of the Livebase platform, and provide instructions on how to adapt the code using these APIs.

Preliminary release (version 6.2.1)

As a preliminary security step, we released an interim version of Livebase (6.2.1) on 24 April. This changed the platform's behavior when restarting a previously stopped cloudlet, eliminating the risk of it being accidentally regenerated.

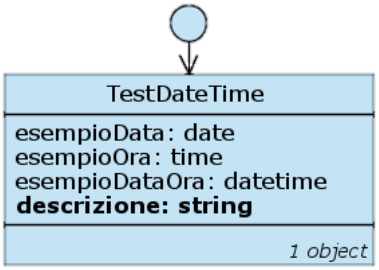
Until previous version of Livebase (6.2.0), when restarting a cloudlet generated with an old version of the platform, runtime regenerated the cloudlet's GraphQL API if it detected a more recent, compatible generator.

Starting with version 6.2.1, Livebase will simply restart the cloudlet without ever regenerating it, even if a more recent, compatible generator is available; the cloudlet will only be regenerated following a change in its model or an explicit rebuild command.

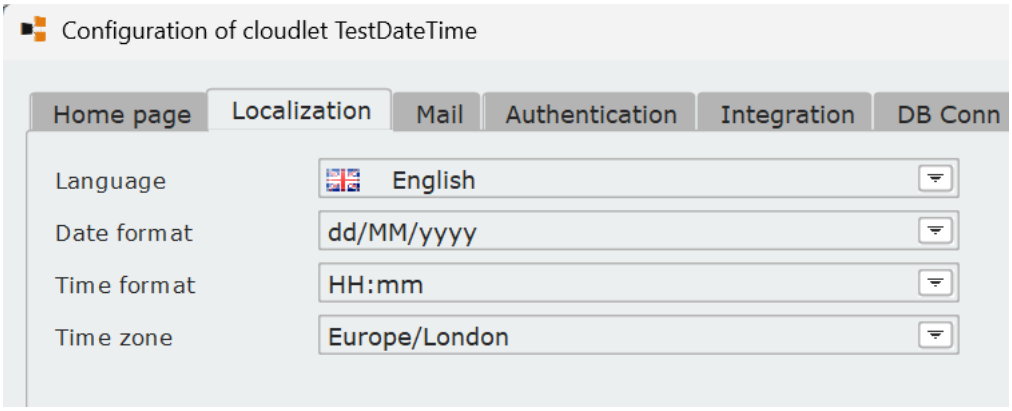
Reference scenario

The examples in this document are taken from the GraphQL API of a custom-built sample cloudlet with the following characteristics:

• The model from which the cloudlet has been generated contains a class configured as follows:



- The cloudlet's localization configuration is as follows:



- • The user running the sample GraphQL queries has the same localization settings as the cloudlet.

Current behavior

In this section, we describe the behavior of the GraphQL API of the sample cloudlet generated with the current version of Livebase (6.2.1).

Mutation

In mutations (write queries), the values provided for Date, Time and Datetime attributes must be supplied according to the format configured for the user performing the mutation. The currently supported formats do NOT include the timezone, and Livebase therefore uses the timezone set for the user.

Request

```
mutation {
  TestDateTime___create(data:{
    descrizione:"Esempio DateTime",
    esempioData:"13/05/2026",
    esempioOra:"12:40",
    esempioDataOra:"01/07/2026 9:15"
  }) {
    descrizione,
    esempioData,
    esempioOra,
    esempioDataOra
  }
}
```

Response

```
{
  "data": {
    "TestDateTime___create": {
      "descrizione": "Esempio DateTime",
      "esempioData": "13/05/2025",
      "esempioOra": "12:40",
      "esempioDataOra": "01/07/2026 09:15"
    }
  }
}
```

Queries without format options

In (read-only) queries without format options, the values returned for Date, Time, and Datetime attributes will be in the format and timezone set for the user running the query.

Request

```
query  {
  TestDateTime___getPage{
    items {
      descrizione,
      esempioData,
      esempioOra
      esempioDataOra
    }
  }
}
```

Response

```
{
  "data": {
    "TestDateTime___getPage": {
      "items": [
        {
          "descrizione": "Esempio DateTime",
          "esempioData": "13/05/2026",
          "esempioOra": "12:40",
          "esempioDataOra": "01/07/2026 09:15"
        }
      ]
    }
  }
}
```

Queries with format options

In (read-only) queries with format options, the values returned for Date, Time and Datetime attributes will be in the format specified by the options given, but always in the timezone set for the user.

Request

```
query  {
  TestDateTime___getPage{
    items {
      descrizione,
      esempioData(format:"yyyy-MM-dd"),
      esempioOra(format:"HH.mm"),
      esempioDataOra(format:"yyyy-MM-dd HH.mm.ss")
    }
  }
}
```

Response

```
{
  "data": {
    "TestDateTime___getPage": {
      "items": [
        {
          "descrizione": "Esempio DateTime",
          "esempioData": "2026-05-13",
          "esempioOra": "12.40",
          "esempioDataOra": "2026-07-01 09.15.00"
        }
      ]
    }
  }
}
```

Queries with search filters

You can specify filters in queries to narrow down search results. For information on how to use these filters and their syntax, please refer to the official Livebase documentation. Any date/time values included in the expressions used in the filters must be in the format specified for the user running the query, similar to what we saw earlier for mutations.

Request

```
query  {
  TestDateTime___getPage(options: {
    filter: {
      esempioData___eq:"13/05/2026"
    }
  }){
    items {
      descrizione,
      esempioData,
      esempioOra
      esempioDataOra
    }
  }
}
```

Response

```
{
  "data": {
    "TestDateTime___getPage": {
      "items": [
        {
          "descrizione": "Esempio DateTime",
          "esempioData": "13/05/2026",
          "esempioOra": "12:40",
          "esempioDataOra": "01/07/2026 09:15"
        }
      ]
    }
```

```
  }
}
```

Future behavior

In this section, we describe the behavior of the GraphQL API of the sample cloudlet, generated with version 6.3.0, which will be released at 12.00pm on May 26, 2026.

Mutation with new standard format

In mutations (write queries), the values for Date, Time and Datetime attributes must be provided according to the standard **ISO 8601** format. For Time and Datetime attributes, the timezone may be specified. If the time zone is not specified (it is optional by default), Livebase will assume it is the time zone of the user performing the mutation

Request

```
mutation {
  TestDateTime___create(data:{
    descrizione:"Esempio DateTime",
    esempioData:"2026-05-13",
    esempioOra:"12:40:00", #without tz, system uses user's setting (+1)
    esempioDataOra:"2026-07-01T09:15:00Z" #with tz stated
  }) {
    descrizione,
    esempioData,
    esempioOra,
    esempioDataOra
  }
}
```

Response

```
{
  "data": {
    "TestDateTime___create": {
      "descrizione": "Esempio DateTime",
      "esempioData": "2026-05-13",
      "esempioOra": "12:40:00+01:00", #In output TZ always stated
      "esempioDataOra": "2026-07-01T10:15:00+01:00" #Specified user's TZ
    }
  }
}
```

Mutation with old, retrocompatibile format (deprecated)

For backward compatibility, in mutations it will still be possible to provide values for Date, Time and Datetime attributes in the format configured for the current user, and Livebase will interpret them as relative to the user's configured timezone.

Request

```
mutation {
  TestDateTime___create(data:{
    descrizione:"Esempio DateTime",
    esempioData:"13/05/2026",
    esempioOra:"12:40",
    esempioDataOra:"01/07/2026 9:15"
  }) {
    descrizione,
    esempioData,
    esempioOra,
    esempioDataOra
  }
}
```

Response

```
{
  "data": { #In output date/time in formato ISO8601
    "TestDateTime___create": {
      "descrizione": "Esempio DateTime",
      "esempioData": "2026-05-13",
      "esempioOra": "12:40:00+01:00",
      "esempioDataOra": "2026-07-01T09:15:00+01:00"
    }
  }
}
```

Queries without format options

In queries (read-only), the values returned for Date, Time and Datetime attributes will be in the standard **ISO 8601** format including timezone, and the timezone specified will always be the one set for the user.

Request

```
query  {
  TestDateTime___getPage{
    items {
      descrizione,
      esempioData,
      esempioOra,
      esempioDataOra
    }
```

```
  }
}
```

Response

```
{
  "data": {
    "TestDateTime___getPage": {
      "items": [
        {
          "descrizione": "Esempio DateTime",
          "esempioData": "2026-05-13",
          "esempioOra": "12:40:00+01:00",
          "esempioDataOra": "2026-07-01T10:15:00+01:00"
        }
      ]
    }
  }
}
```

Queries with format option

In (read-only) queries with format options, the values returned for Date, Time, and Datetime attributes will be in the format specified by the specified options, but always within the timezone set for the user.

In this case, the cloudlet generated with Livebase version 6.3.0 will behave identically to that of cloudlets generated with previous versions of the platform.

Queries with search filters

For queries with search filters, the rules outlined for mutations apply. We therefore recommend using the ISO8601 format, although the old format is still supported.

Request

```
query  {
  TestDateTime___getPage(options: {
    filter: {
      esempioData___eq:"2026-05-13" #or "13/05/2026"
    }
  }){
    items {
      descrizione,
      esempioData,
      esempioOra
      esempioDataOra
    }
  }
```

```
}
```

Response

```
{
  "data": {
    "TestDateTime___getPage": {
      "items": [
        {
          "descrizione": "Esempio DateTime",
          "esempioData": "2026-05-13",
          "esempioOra": "12:40:00+01:00",
          "esempioDataOra": "2026-07-01T10:15:00+01:00"
        }
      ]
    }
  }
}
```

Migration guidelines

A backwards compatibility issue may arise in software components that execute GraphQL **queries** to obtain the value of Date, Time, or Datetime attributes from the GraphQL API of cloudlets rebuilt with Livebase version 6.3.0 (i.e., rebuilt on or after May 26, 2026). This issue will only occur if the GraphQL queries do not contain format options: in this case, the values returned by the API will no longer follow the format set for the user, but rather the **ISO 8601** standard.

To restore compatibility in these cases, there are two options:

- Modify the code that executes the query to correctly interpret the values returned for Date, Time, and Datetime attributes in ISO 8601 format rather than the format set for the user. This solution is recommended because it makes the integration of the software component that executes the query immune to any subsequent changes to the user or cloudlet's localization settings.

- add a format option to the query for each Date, Time, and Datetime attribute that matches the one set for the user. This solution is not recommended because it makes the software component integration vulnerable to any subsequent changes to the user's or cloudlet's localization settings.

We also recommend to modify all the GraphQL **mutations** involving Date, Time, or Datetime attributes in order to provide values formatted according to the the **ISO 8601** standard, rather than according the user localization settings. This will make the software component integration not vulnerable to any subsequent changes to the user's or cloudlet's localization settings.