

Firebird and REST (PDF)

PDF Download

Today, you often need to handle REST requests. If you haven't come across this yet:

"Representational state transfer (REST) is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web. REST defines a set of constraints for how the architecture of an Internet-scale distributed hypermedia system, such as the Web, should behave. The REST architectural style emphasizes the scalability of interactions between components, uniform interfaces, independent deployment of components, and the creation of a layered architecture to facilitate caching components to reduce user-perceived latency, enforce security, and encapsulate legacy systems."

IBEBlock can handle REST requests and the data received, just by using functions provided by IBEScript. The following example illustrates that any REST API can be used based on IBExpert scripting without any real development knowledge.

The following is an example of how to use a REST API to get the exact GPS location of, for example, train stations in Germany.

Example 1

- Create an IBEBlock script that monitors a table somewhere in the database using `ibec_waitforevent`.
- When a table gets a new entry (for example a city name for the train station), it should stop waiting because the event is posted by a trigger, gets the data from the REST API, puts the results into the database in native format, the trigger will then convert the native format to a readable text format or individual records and stores all this in the database.

For this example, I am using <https://data.deutschebahn.com/dataset/api-fahrplan.html> and will use the free version of their REST API, for sample locations:
<https://developer.deutschebahn.com/store/apis/info?name=Fahrplan-Free&version=v1&provider=DBOpenData#/>

This is a RESTful webservice to request a railway journey - free plan with restricted access (max. 10 requests per minute). However, you can register to use a less restricted version which requires an access token.

As a first step, you can use an already existing database and extend it with the following, or create your own sample from scratch:

1. Create a sample table which contains data and which we will monitor:

```
CREATE TABLE PLZ (  
    PLZ  VARCHAR(255) NOT NULL,  
    ORT  VARCHAR(255)  
);
```

2. Create a table trigger to post an event so that data is inserted, and set the necessary context variables to pass to the user session:

```
CREATE OR ALTER TRIGGER PLZ_AI0 FOR PLZ
ACTIVE AFTER INSERT POSITION 0
AS
begin
    RDB$SET_CONTEXT('USER_SESSION', 'LAST_ORT', new.ORT);
    RDB$SET_CONTEXT('USER_SESSION', 'LAST_PLZ', new.PLZ);
    POST_EVENT 'new_ort';
END
```

3. Create additional tables to store raw data received from the REST API, and to store single records parsed from the REST API request:

```
CREATE TABLE DEUTSCHEBAHN (
    ID      INTEGER GENERATED BY DEFAULT AS IDENTITY,
    JSON    BLOB SUB_TYPE 1 SEGMENT SIZE 80,
    PLZ     VARCHAR(255)
);

CREATE TABLE DEUTSCHEBAHN_GPS (
    ID      VARCHAR(255),
    NAME    VARCHAR(255),
    LON     FLOAT,
    LAT     FLOAT
);
```

Now create an IBEBlock script to handle the job: If you want to connect to your database and start the script in the current connection, you can use `ibec_GetDefaultConnection()`. Alternatively use something like the following, adjusting the database path and authentication data:

```
ibec_CreateConnection(__ctFirebird, 'DBName="localhost:c:\DB\REST.FDB";
                                ClientLib=fbclient.dll;
                                user=SYSDBA; password=masterkey; names=UTF8;
                                sqldialect=3');
```

The next step should be to wait for a posted event:

```
Res = ibec_WaitForEvent(FBSrc, 'new_ort', 0);
```

If an event is received, fetch the data from the context variables, and execute the REST call to the REST API:

```
if (Res is not null) then begin
    ort = RDB$GET_CONTEXT('USER_SESSION', 'LAST_ORT');
    plz = RDB$GET_CONTEXT('USER_SESSION', 'LAST_PLZ');

    url  = 'URL=https://api.deutschebahn.com/freeplan/v1/location/' ||
```

```
ort;
```

Now execute the function to get data from the REST API, and, if successful, insert into a table as raw data:

```
ibec_http_Get(:sess);

scode = ibec_http_StatusCode(:sess);
sdesc = ibec_http_Data(:sess);
if (scode = 200) then begin
    INSERT INTO DEUTSCHEBAHN (PLZ, JSON) VALUES (:plz, :sdesc);
```

It will always return an array of data, but not exactly as standard JSON, so that needs to be modified prior to parsing, using our JSON functions:

```
sdesc = '{"Res" : {  "Items" : ' + sdesc + '}}';
```

Parsing is then done by also using our JSON functions:

```
JsonRoot = ibec_json_Parse(sdesc);

try
    ArrayNode = @Json.SelectNode(JsonRoot, 'res\items',
@Json.SEARCH_IGNORE_CASE); -- case insensitive search
    begin
        iCount = ibec_json_ChildCount(ArrayNode);
        for i = 0 to (iCount - 1) do
            begin

                ChildNode = ibec_json_SelectNode(ArrayNode, i,
__jsonSearchByIndex);
                NType = ibec_json_NodeType(ChildNode);
                if (NType = __jsonObject) then begin

                    N = ibec_json_SelectNode(ChildNode, 'name', 0);
                    vName = ibec_json_GetNodeValue(N, TRUE);
                    id = ibec_json_SelectNode(ChildNode, 'id', 0);
                    vID = ibec_json_GetNodeValue(id, TRUE);
                    lon = ibec_json_SelectNode(ChildNode, 'lon', 0);
                    vLon = ibec_json_GetNodeValue(lon, TRUE);
                    lat = ibec_json_SelectNode(ChildNode, 'lat', 0);
                    vLat = ibec_json_GetNodeValue(lat, TRUE);

                    try
                        INSERT INTO DEUTSCHEBAHN_GPS (id, name, lat, lon) VALUES (:vID,
:vName, :vLat, :vLon);
                    except
                        end;
```

At the end, don't forget to free objects and close the session, as well as close the database

connection:

```
        finally
            ibec_json_Free(JsonRoot);
        end

        ibec_http_CloseSession(:sess);
    end
end;
finally
    ibec_CloseConnection(FBSrc);
end;
```

So, in this context, the complete script will look as follows when ready to execute:

```
execute ibeblock
as
begin
    try
        FBSrc = ibec_GetDefaultConnection();
    /*
ibec_CreateConnection(__ctFirebird,'DBName="localhost:c:\DB\REST.FDB";
                        ClientLib=fbclient.dll;
                        user=SYSDBA; password=masterkey; names=UTF8;
sql dialect=3'); */

    Res = ibec_WaitForEvent(FBSrc, 'new_ort', 0);
    if (Res is not null) then begin
        ort = RDB$GET_CONTEXT('USER_SESSION', 'LAST_ORT');

        plz = RDB$GET_CONTEXT('USER_SESSION', 'LAST_PLZ');

        url = 'URL=https://api.deutschebahn.com/freeplan/v1/location/' ||
ort;
        sess = ibec_http_OpenSession(:url);

        ibec_http_Get(:sess);

        scode = ibec_http_StatusCode(:sess);
        sdesc = ibec_http_Data(:sess);
        if (scode = 200) then begin
            INSERT INTO DEUTSCHEBAHN (PLZ, JSON) VALUES (:plz, :sdesc);

            sdesc = '{"Res" : {  "Items" : ' + sdesc + '}}';
            JsonRoot = ibec_json_Parse(sdesc);
            try
                ArrayNode = @Json.SelectNode(JsonRoot, 'res\items',
@Json.SEARCH_IGNORE_CASE); -- case insensitive search
```

```

        iCount = ibec_json_ChildCount(ArrayNode);
        for i = 0 to (iCount - 1) do
            begin
                ChildNode = ibec_json_SelectNode(ArrayNode, i,
__jsonSearchByIndex);
                NType = ibec_json_NodeType(ChildNode);
                if (NType = __jsonObject) then begin
                    N = ibec_json_SelectNode(ChildNode, 'name', 0);
                    vName = ibec_json_GetNodeValue(N, TRUE);
                    id = ibec_json_SelectNode(ChildNode, 'id', 0);
                    vID = ibec_json_GetNodeValue(id, TRUE);
                    lon = ibec_json_SelectNode(ChildNode, 'lon', 0);
                    vLon = ibec_json_GetNodeValue(lon, TRUE);
                    lat = ibec_json_SelectNode(ChildNode, 'lat', 0);
                    vLat = ibec_json_GetNodeValue(lat, TRUE);
                    try
                        INSERT INTO DEUTSCHEBAHN_GPS (id, name, lat, lon) VALUES
(:vID, :vName, :vLat, :vLon);
                    except
                        end;
                    end;
                end;
            finally
                ibec_json_Free(JsonRoot);
            end
        end;
        ibec_http_CloseSession(:sess);
    end

finally
    commit;
    ibec_CloseConnection(FBSrc);
end;
end

```

To test the example posted above, you can insert some data in the PLZ table, or alternatively execute a query to insert a record:

```
INSERT INTO PLZ (PLZ, ORT) VALUES ('01067', 'Dresden');
```

The previous example shows you how to use Firebird events, and it can be modified to traverse existing records in the PLZ table, returning data and inserting it into the database. So instead of using an event to inform us to trigger data retrieval, we can use a loop, in this case for 30 rows in our table:

```

execute IBEBlock
as
begin
    try
        Stmt = 'select plz, ort from plz rows 30';
        for execute statement :Stmt into :plz, :ort do
            begin

```

```
ibec_Progress('data ' + plz + ' - ' + ort);

url    = 'URL=https://api.deutschebahn.com/freeplan/v1/location/' ||
ort;

sess   = ibec_http_OpenSession(:url);

ibec_http_Get(:sess);

scode = ibec_http_StatusCode(:sess);
sdesc = ibec_http_Data(:sess);
if (scode = 200) then begin
    INSERT INTO DEUTSCHEBAHN (PLZ, JSON) VALUES (:plz, :sdesc);

    sdesc = '{"Res" : {  "Items" : ' + sdesc + '}}';
    JsonRoot = ibec_json_Parse(sdesc);
    try
        ArrayNode = @Json.SelectNode(JsonRoot, 'res\items',
@Json.SEARCH_IGNORE_CASE); -- case insensitive search

        iCount = ibec_json_ChildCount(ArrayNode);
        for i = 0 to (iCount - 1) do
            begin
                ChildNode = ibec_json_SelectNode(ArrayNode, i,
__jsonSearchByIndex);
                NType = ibec_json_NodeType(ChildNode);
                if (NType = __jsonObject) then begin
                    N = ibec_json_SelectNode(ChildNode, 'name', 0);
                    vName = ibec_json_GetNodeValue(N, TRUE);
                    id = ibec_json_SelectNode(ChildNode, 'id', 0);
                    vID = ibec_json_GetNodeValue(id, TRUE);
                    lon = ibec_json_SelectNode(ChildNode, 'lon', 0);
                    vLon = ibec_json_GetNodeValue(lon, TRUE);
                    lat = ibec_json_SelectNode(ChildNode, 'lat', 0);
                    vLat = ibec_json_GetNodeValue(lat, TRUE);
                    try
                        INSERT INTO DEUTSCHEBAHN_GPS (id, name, lat, lon) VALUES
(:vID, :vName, :vLat, :vLon);
                    except
                        end;
                    end;
                end;
            end;
        finally
            ibec_json_Free(JsonRoot);
        end
    end;
    ibec_http_CloseSession(:sess);
end

finally
    commit;
```

```
end;
end
```

If you would like some more simple examples, there are some Fake Online REST API for Testing and Prototyping available here <http://www.dummy.restapiexample.com/>, so you can play with those like:

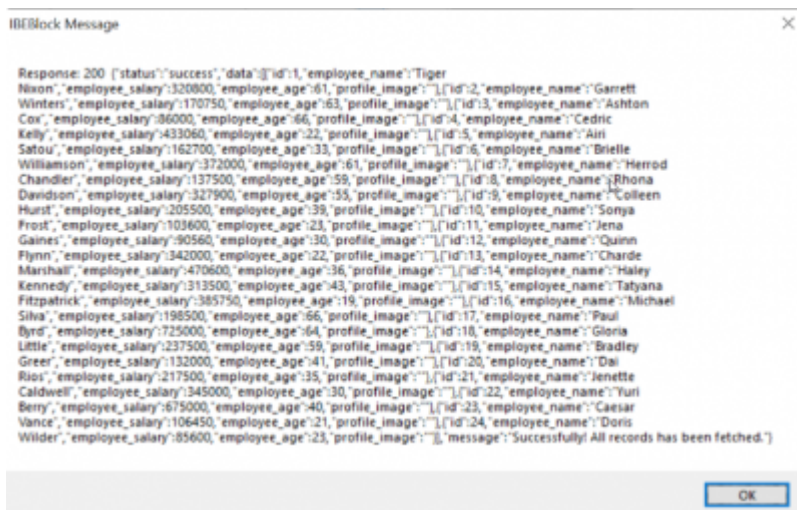
Example 2

```
execute IBEBlock
as
begin
    url    = 'URL=http://dummy.restapiexample.com/api/v1/employees';
    sess   = ibec_http_OpenSession(:url);

    ibec_http_Get(:sess);
    scode  = ibec_http_StatusCode(:sess);
    sdesc  = ibec_http_Data(:sess);
    if (scode = 200) then
        ibec_ShowMessage('Response: ' || :scode || '
' || ibec_UTF8ToAnsiString(:sdesc));
    else if (scode = 429) then
        ibec_ShowMessage('Too many requests');

    ibec_http_CloseSession(:sess);
end
```

This will return the following:



Or execute similar for a single record:

```
execute IBEBlock
as
begin
    url    = 'URL=http://dummy.restapiexample.com/api/v1/employee/1';
    sess   = ibec_http_OpenSession(:url);
```

```
ibec_http_Get(:sess);

scode = ibec_http_StatusCode(:sess);
sdesc = ibec_http_Data(:sess);
if (scode = 200) then
    ibec_ShowMessage('Response: ' || :scode || '
' || ibec_UTF8ToAnsiString(:sdesc));
else if (scode = 429) then
    ibec_ShowMessage('Too many requests');
ibec_http_CloseSession(:sess);

end
```

will return the employee with id=1, for example:



Please note that this REST testing service can be under a heavy load, and if the status code is 429 there will be an error message saying “Too many requests”.

This is a minimal working example that can be extended further, with some ideas posted in the more complex examples above. We have shown that any REST API can be used based on IBExpert scripting, without any in-depth knowledge in development and the results are simply accessible in the database.

From:
<http://ibexpert.com/docu/> - IBExpert

Permanent link:
<http://ibexpert.com/docu/doku.php?id=01-documentation:01-06-white-papers:firebird-and-rest>

Last update: **2023/06/20 17:12**

