Beckhoff PLC (TwinCAT 3)

PLC programming by using OOP approach

In this paper, we are going to have a look OOP programming more deeply and SQL injection which is a code injection technique that might destroy your database. Beside, while the technology is being developed in the software area, security of a program is getting more important. That’s why I believe that we need to think about and evaluate security in any piece of code we have created.

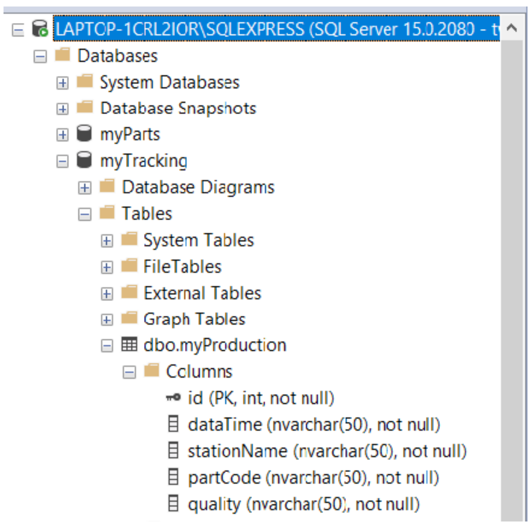
What we will do;

- Configuration of Connection
- Insert Method
- SQL Injection
- Sanitizing Input

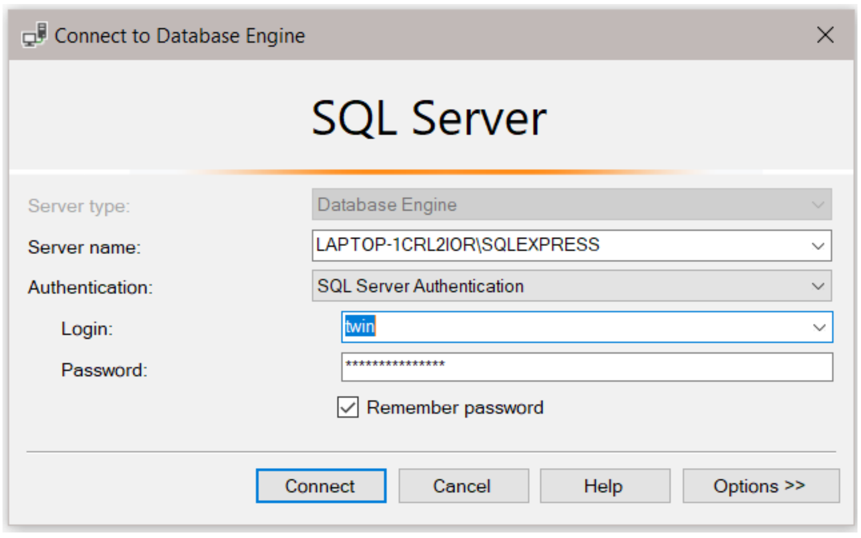
Configuration

In TwinCAT, we don’t have to implement a connection method for databases because we can use TF6420 database server configurator and create a connection, then we could use this connection by adding it to DB configuration pool. It means that TwinCat always has a connection with DB, so we don’t need to do anything to get a connection.

In example, we are using MS SQL Server, our database’s name is myTracking and we have 5 columns which are id, dataTime, stationName, partCode and quality.



```
⊟ LAPTOP-1CRL2IOR\SQLEXPRESS (SQL Server 15.0.2080 - t
  ⊟ Databases
    ⊞ System Databases
    ⊞ Database Snapshots
    ⊞ myParts
    ⊟ myTracking
      ⊞ Database Diagrams
      ⊟ Tables
        ⊞ System Tables
        ⊞ FileTables
        ⊞ External Tables
        ⊞ Graph Tables
        ⊟ dbo.myProduction
          ⊟ Columns
              id (PK, int, not null)
              dataTime (nvarchar(50), not null)
              stationName (nvarchar(50), not null)
              partCode (nvarchar(50), not null)
              quality (nvarchar(50), not null)
```



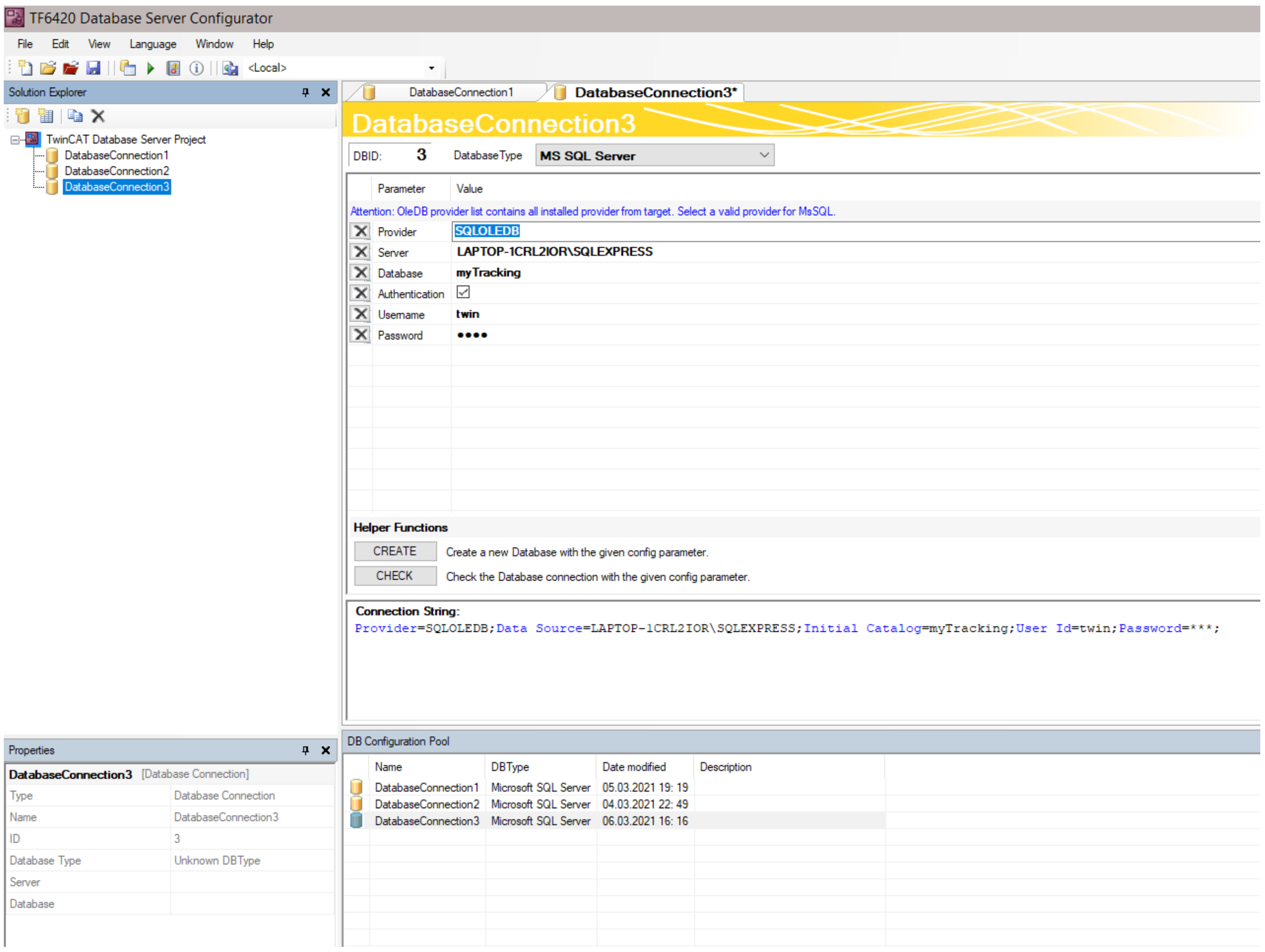
Connect to Database Engine

SQL Server

Server type:	Database Engine
Server name:	LAPTOP-1CRL2IOR\SQLEXPRESS
Authentication:	SQL Server Authentication
Login:	twin
Password:	***************

☑ Remember password

After creating the database and columns, we need to open TF6420 and create a connection. At this point, we support some information to TF6420.



TF6420 Database Server Configurator

```
⊟ TwinCAT Database Server Project
    DatabaseConnection1
    DatabaseConnection2
    DatabaseConnection3
```

DatabaseConnection3

DBID: 3 DatabaseType MS SQL Server

Attention: OleDB provider list contains all installed provider from target. Select a valid provider for MsSQL.

Parameter	Value
Provider	SQLOLEDB
Server	LAPTOP-1CRL2IOR\SQLEXPRESS
Database	myTracking
Authentication	☑
Username	twin
Password	●●●●

Helper Functions

CREATE	Create a new Database with the given config parameter.
CHECK	Check the Database connection with the given config parameter.

Connection String:

```
Provider=SQLOLEDB;Data Source=LAPTOP-1CRL2IOR\SQLEXPRESS;Initial Catalog=myTracking;User Id=twin;Password=***;
```

Properties

DatabaseConnection3 [Database Connection]

Type	Database Connection
Name	DatabaseConnection3
ID	3
Database Type	Unknown DBType
Server	
Database	

DB Configuration Pool

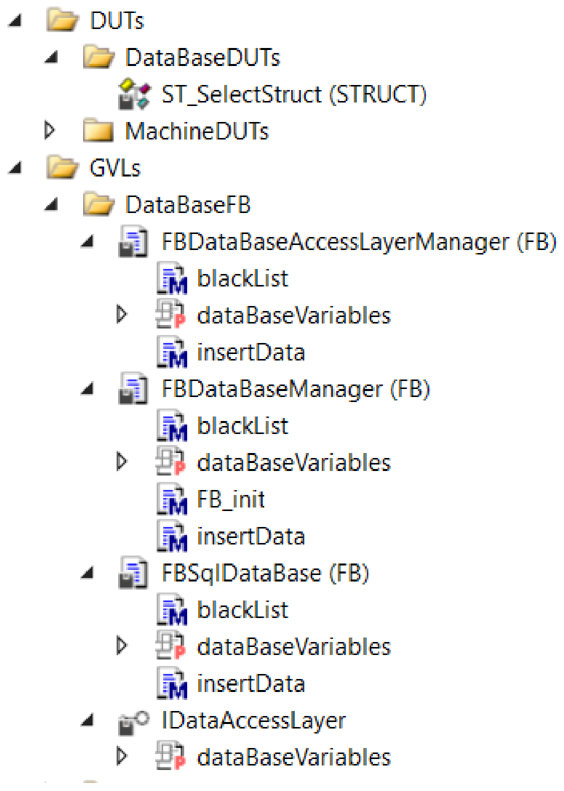
Name	DBType	Date modified	Description
DatabaseConnection1	Microsoft SQL Server	05.03.2021 19: 19	
DatabaseConnection2	Microsoft SQL Server	04.03.2021 22: 49	
DatabaseConnection3	Microsoft SQL Server	06.03.2021 16: 16	

DBID defines which DB we are going to connect because we might have more than one DB and we will use this DBID inside TwinCAT to connect to correct DB.

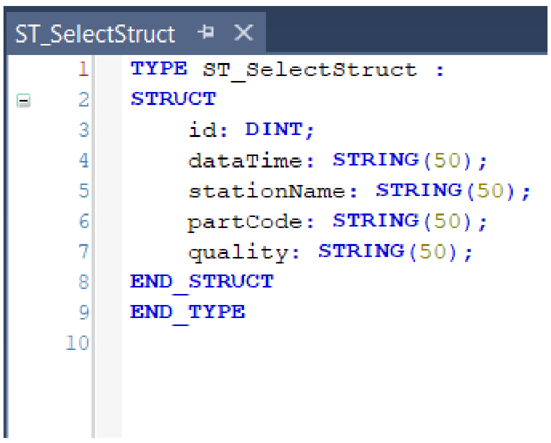
Insert Method

In the last paper, we have looked abstract function blocks, inheritance and polymorphic structure, so we don’t review here again. For fully understanding, please check my first paper;

(https://www.linkedin.com/posts/murattoprak_opp-in-plc-activity-6772583672387256320-c3Bo).

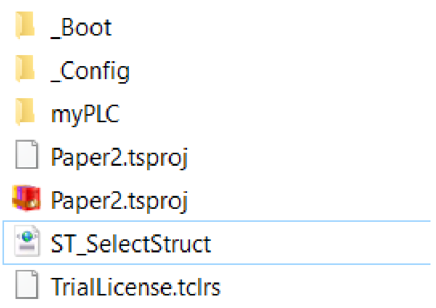


```
DUTs
  DataBaseDUTs
    ST_SelectStruct (STRUCT)
  MachineDUTs
GVLs
  DataBaseFB
    FBDataBaseAccessLayerManager (FB)
      blackList
      dataBaseVariables
      insertData
    FBDataBaseManager (FB)
      blackList
      dataBaseVariables
      FB_init
      insertData
    FBSqlDataBase (FB)
      blackList
      dataBaseVariables
      insertData
    IDataAccessLayer
      dataBaseVariables
```



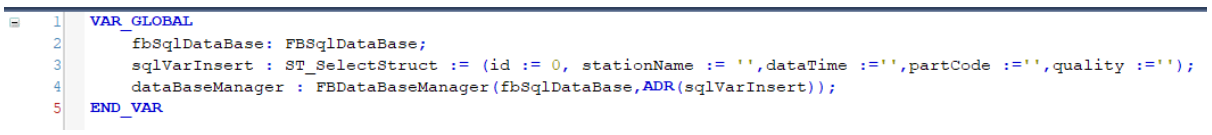
ST_SelectStruct

```
TYPE ST_SelectStruct :
STRUCT
    id: DINT;
    dataTime: STRING(50);
    stationName: STRING(50);
    partCode: STRING(50);
    quality: STRING(50);
END_STRUCT
END_TYPE
```



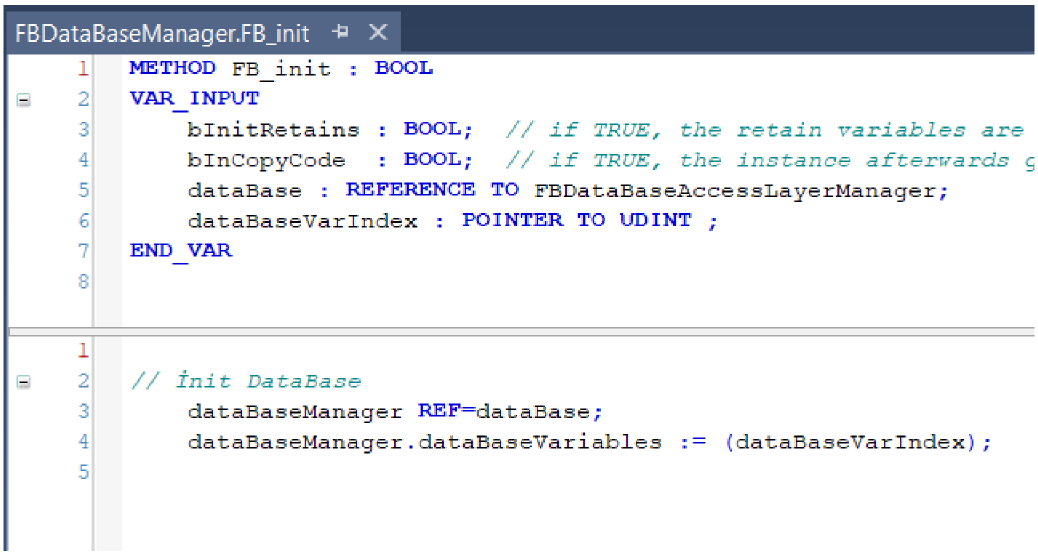
```
_Boot
_Config
myPLC
Paper2.tsproj
Paper2.tsproj
ST_SelectStruct
TrialLicense.tclrs
```

In this case, we are going to insert some data. That’s why we have additional things inside the program. These are IdataAccesLayer interface which has property with name of databaseVariables and ST_SelectStruct which are exported from TF6420 and imported to TwinCAT, but you don’t have to do in this way because it doesn’t affect anything, so if you want, you can create your own structure.



```
VAR_GLOBAL
    fbSqlDataBase: FBSqlDataBase;
    sqlVarInsert : ST_SelectStruct := (id := 0, stationName := '',dataTime :='',partCode :='',quality :='');
    dataBaseManager : FBDataBaseManager(fbSqlDataBase,ADR(sqlVarInsert));
END_VAR
```

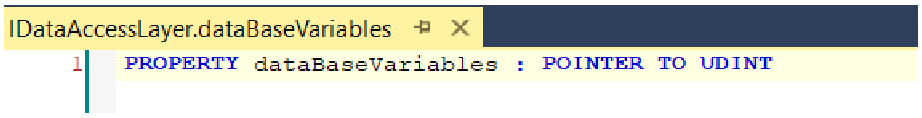
In addition, we need to give the address of our structure which stored our data to FBDataBaseManager not to break solid programming standard. After a while, our needs can be changed, so we will just give the address of new structure.



```
METHOD FB_init : BOOL
VAR_INPUT
    bInitRetains : BOOL;  // if TRUE, the retain variables are
    bInCopyCode  : BOOL;  // if TRUE, the instance afterwards g
    dataBase : REFERENCE TO FBDataBaseAccessLayerManager;
    dataBaseVarIndex : POINTER TO UDINT ;
END_VAR

```

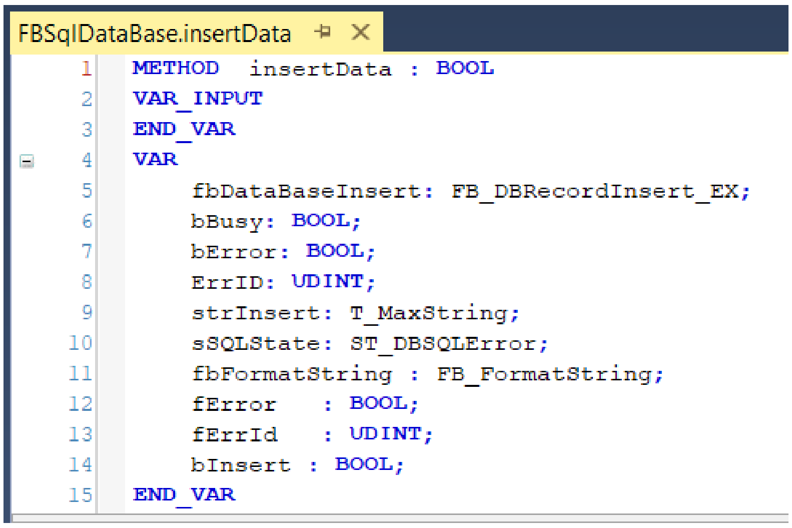
```

// init DataBase
    dataBaseManager REF=dataBase;
    dataBaseManager.dataBaseVariables := (dataBaseVarIndex);

```

Inside the init function, we just set the address of dataBaseVariables because we defined dataBaseVariables as a pointer.



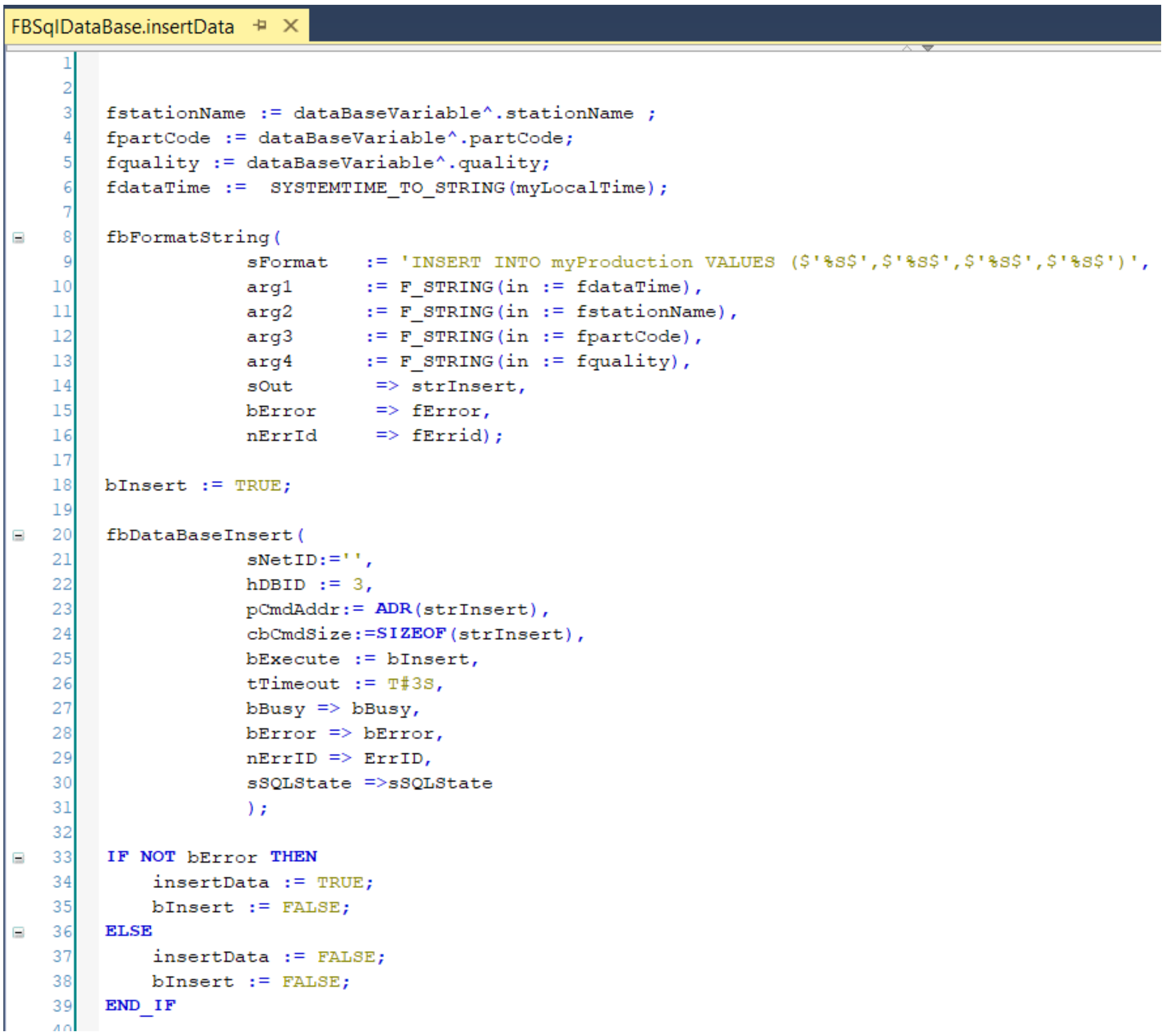
```
PROPERTY dataBaseVariables : POINTER TO UDINT
```

In the insertData method, FB_DBRecordInsert_EX is used for inserting data to the database and this function inserts one data set in one execution. Also FB_FormatString is used for formatting strInsert variables. These two functions are elements of Tc_Database and Tc2_Utilities libraries. Please don’t forget to add libraries.



```
METHOD  insertData : BOOL
VAR_INPUT
END_VAR
VAR
    fbDataBaseInsert: FB_DBRecordInsert_EX;
    bBusy: BOOL;
    bError: BOOL;
    ErrID: UDINT;
    strInsert: T_MaxString;
    sSQLState: ST_DBSQLError;
    fbFormatString : FB_FormatString;
    fError    : BOOL;
    fErrId    : UDINT;
    bInsert : BOOL;
END_VAR
```

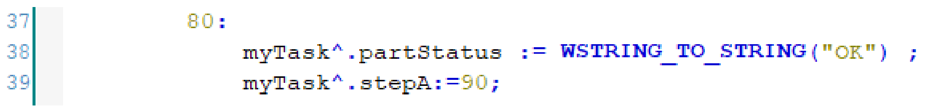
Every $'%S$' refers to one argument and we send id to database because id is increased one in every insertion by database itself. Furthermore, hDBID input of FB_DBRecordInsert_EX is constant 3, which is exactly the same value with TF6420. When bExecute input is given, the function inserts one data set and if there is no error in insertion, the method returns True, otherwise False.


FBSqlDataBase.insertData

```


fstationName := dataBaseVariable^.stationName ;
fpartCode := dataBaseVariable^.partCode;
fquality := dataBaseVariable^.quality;
fdataTime :=  SYSTEMTIME_TO_STRING(myLocalTime);

fbFormatString(
            sFormat   := 'INSERT INTO myProduction VALUES ($'%S$',$'%S$',$'%S$',$'%S$')',
            arg1      := F_STRING(in := fdataTime),
            arg2      := F_STRING(in := fstationName),
            arg3      := F_STRING(in := fpartCode),
            arg4      := F_STRING(in := fquality),
            sOut       => strInsert,
            bError     => fError,
            nErrId     => fErrid);

bInsert := TRUE;

fbDataBaseInsert(
            sNetID:='',
            hDBID  := 3,
            pCmdAddr:= ADR(strInsert),
            cbCmdSize:=SIZEOF(strInsert),
            bExecute  := bInsert,
            tTimeout  := T#3S,
            bBusy => bBusy,
            bError => bError,
            nErrID => ErrID,
            sSQLState =>sSQLState
            );

IF NOT bError THEN
    insertData := TRUE;
    bInsert := FALSE;
ELSE
    insertData := FALSE;
    bInsert := FALSE;
END_IF
```

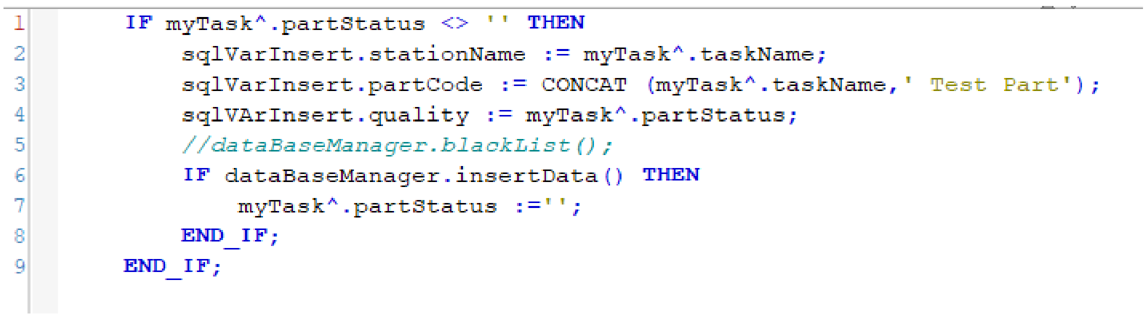
SQL INJECTION

We give the value of quality column inside the machine auto method which is explained later. Firstly we insert some data, then we will manipulate the value of quality and we see what will be happened because all variables can be come from outside of PLC and this means that our inputs can be manipulated.

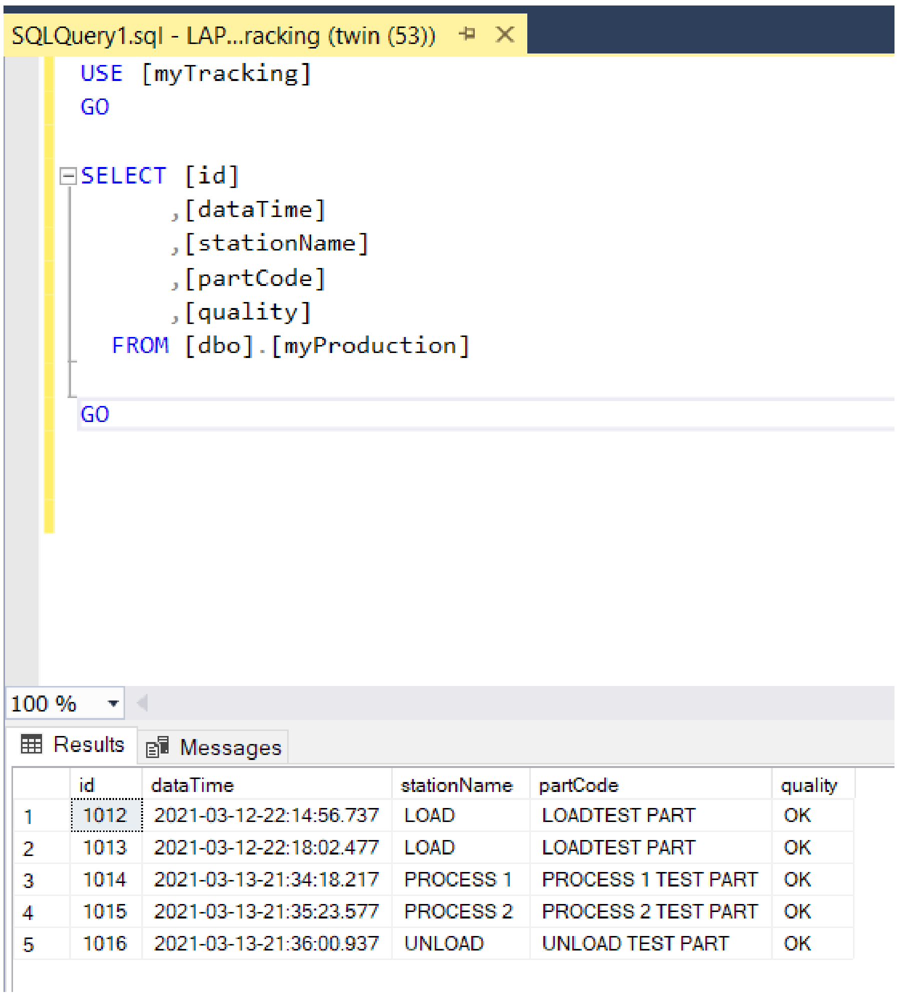


```
        80:
            myTask^.partStatus := WSTRING_TO_STRING("OK") ;
            myTask^.stepA:=90;
```

We can see some inserted data below by calling insertData method.



```
IF myTask^.partStatus <> '' THEN
    sqlVarInsert.stationName := myTask^.taskName;
    sqlVarInsert.partCode := CONCAT (myTask^.taskName,' Test Part');
    sqlVArInsert.quality := myTask^.partStatus;
    //dataBaseManager.blackList();
    IF dataBaseManager.insertData() THEN
        myTask^.partStatus :='';
    END_IF;
END_IF;
```



SQLQuery1.sql - LAP...racking (twin (53))

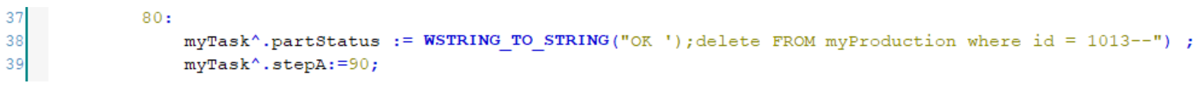
```
USE [myTracking]
GO

SELECT [id]
      ,[dataTime]
      ,[stationName]
      ,[partCode]
      ,[quality]
  FROM [dbo].[myProduction]

GO
```

100 %

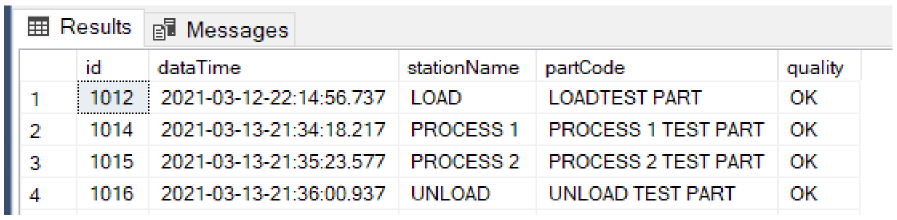
Results | Messages

	id	dataTime	stationName	partCode	quality
1	1012	2021-03-12-22:14:56.737	LOAD	LOADTEST PART	OK
2	1013	2021-03-12-22:18:02.477	LOAD	LOADTEST PART	OK
3	1014	2021-03-13-21:34:18.217	PROCESS 1	PROCESS 1 TEST PART	OK
4	1015	2021-03-13-21:35:23.577	PROCESS 2	PROCESS 2 TEST PART	OK
5	1016	2021-03-13-21:36:00.937	UNLOAD	UNLOAD TEST PART	OK

Now we manipulate string value of quality and delete second row which is id is equal to 1013.



```
        80:
            myTask^.partStatus := WSTRING_TO_STRING("OK ');delete FROM myProduction where id = 1013--") ;
            myTask^.stepA:=90;
```



Results | Messages

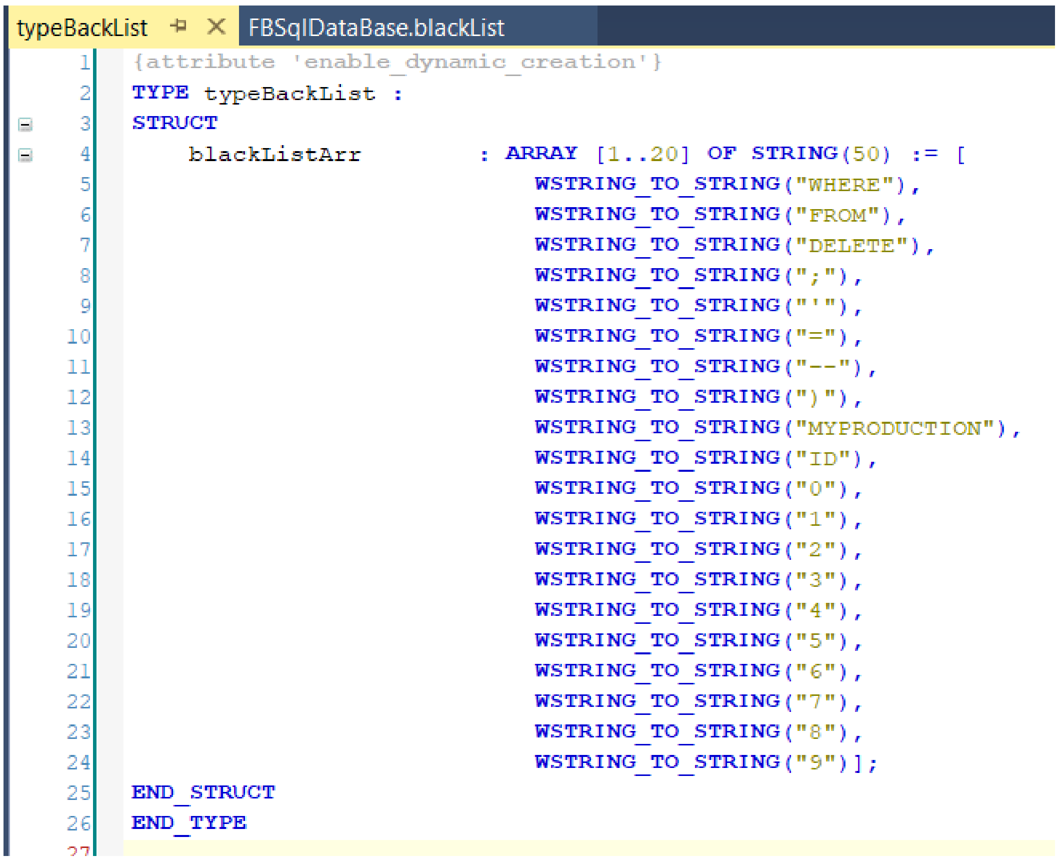
	id	dataTime	stationName	partCode	quality
1	1012	2021-03-12-22:14:56.737	LOAD	LOADTEST PART	OK
2	1014	2021-03-13-21:34:18.217	PROCESS 1	PROCESS 1 TEST PART	OK
3	1015	2021-03-13-21:35:23.577	PROCESS 2	PROCESS 2 TEST PART	OK
4	1016	2021-03-13-21:36:00.937	UNLOAD	UNLOAD TEST PART	OK

Now suppose that quality data is taken from a computer which tests part, then sends data and opens to the internet, so someone can penetrate that computer and destroy our database. Therefore, we have to prevent this and there are some techniques to do this. The most used technique is parameterized queries but this technique is provided with IDE. For this reason, I did a research and I couldn't find anything about it for TwinCAT but SQL Expert Mode in TwinCAT we might use SQL query editor to prevent SQL injection but in this example, we are going to use sanitization method to eliminate exact character.

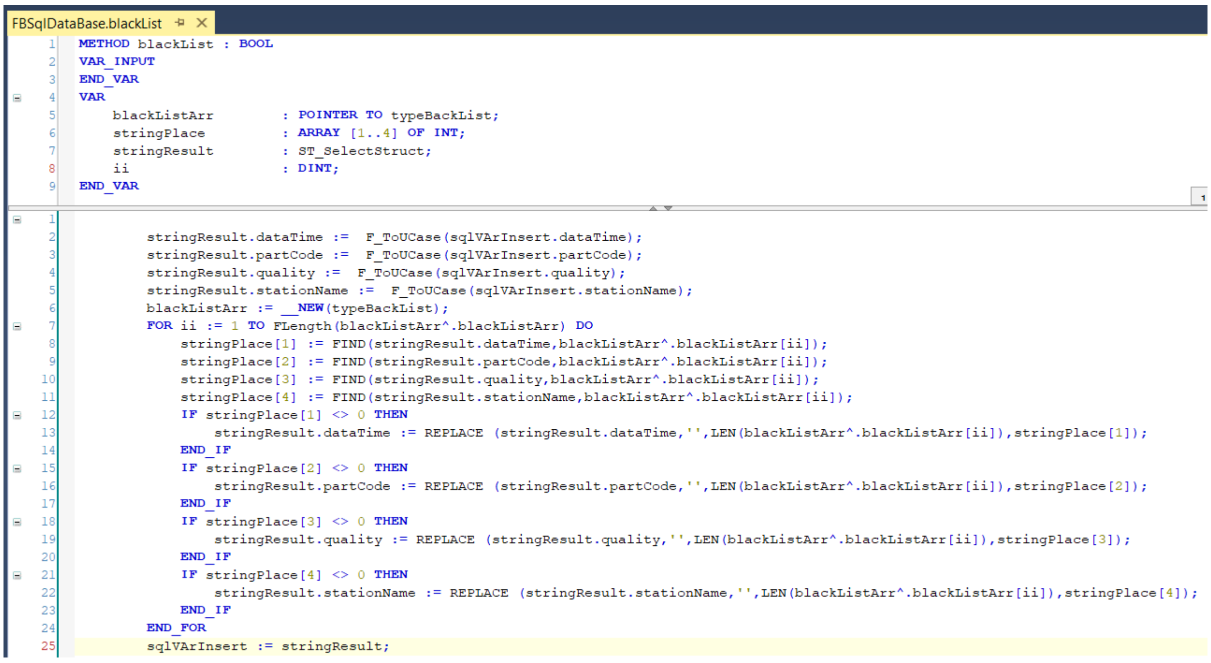
Sanitizing Input

We call this method as a blackList and we will just delete some characters, so this method can not prevent perfectly and we need to extend our list to prevent properly.

Firstly All inputs are in uppercase, then searching characters inside inputs and if it is inside, it is removed from inputs. After calling blackList method, we can do an insertion.



```
{attribute 'enable_dynamic_creation'}
TYPE typeBackList :
STRUCT
    blackListArr        : ARRAY [1..20] OF STRING(50) := [
                            WSTRING_TO_STRING("WHERE"),
                            WSTRING_TO_STRING("FROM"),
                            WSTRING_TO_STRING("DELETE"),
                            WSTRING_TO_STRING(";"),
                            WSTRING_TO_STRING("'"),
                            WSTRING_TO_STRING("="),
                            WSTRING_TO_STRING("--"),
                            WSTRING_TO_STRING(")"),
                            WSTRING_TO_STRING("MYPRODUCTION"),
                            WSTRING_TO_STRING("ID"),
                            WSTRING_TO_STRING("0"),
                            WSTRING_TO_STRING("1"),
                            WSTRING_TO_STRING("2"),
                            WSTRING_TO_STRING("3"),
                            WSTRING_TO_STRING("4"),
                            WSTRING_TO_STRING("5"),
                            WSTRING_TO_STRING("6"),
                            WSTRING_TO_STRING("7"),
                            WSTRING_TO_STRING("8"),
                            WSTRING_TO_STRING("9")];
END_STRUCT
END_TYPE
```

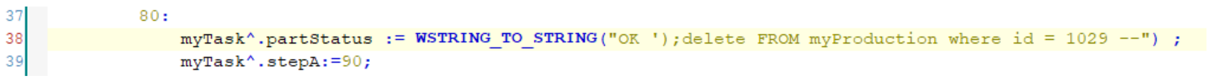


FBSqlDataBase.blackList

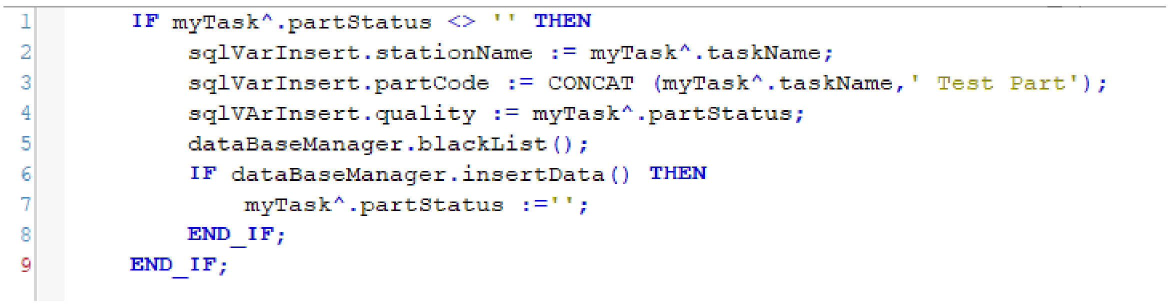
```
METHOD blackList : BOOL
VAR_INPUT
END_VAR
VAR
    blackListArr        : POINTER TO typeBackList;
    stringPlace         : ARRAY [1..4] OF INT;
    stringResult        : ST_SelectStruct;
    ii                  : DINT;
END_VAR
```

```

        stringResult.dataTime :=  F_ToUCase(sqlVArInsert.dataTime);
        stringResult.partCode :=  F_ToUCase(sqlVArInsert.partCode);
        stringResult.quality :=  F_ToUCase(sqlVArInsert.quality);
        stringResult.stationName :=  F_ToUCase(sqlVArInsert.stationName);
        blackListArr := __NEW(typeBackList);
        FOR ii := 1 TO FLength(blackListArr^.blackListArr) DO
            stringPlace[1] := FIND(stringResult.dataTime,blackListArr^.blackListArr[ii]);
            stringPlace[2] := FIND(stringResult.partCode,blackListArr^.blackListArr[ii]);
            stringPlace[3] := FIND(stringResult.quality,blackListArr^.blackListArr[ii]);
            stringPlace[4] := FIND(stringResult.stationName,blackListArr^.blackListArr[ii]);
            IF stringPlace[1] <> 0 THEN
                stringResult.dataTime := REPLACE (stringResult.dataTime,'',LEN(blackListArr^.blackListArr[ii]),stringPlace[1]);
            END_IF
            IF stringPlace[2] <> 0 THEN
                stringResult.partCode := REPLACE (stringResult.partCode,'',LEN(blackListArr^.blackListArr[ii]),stringPlace[2]);
            END_IF
            IF stringPlace[3] <> 0 THEN
                stringResult.quality := REPLACE (stringResult.quality,'',LEN(blackListArr^.blackListArr[ii]),stringPlace[3]);
            END_IF
            IF stringPlace[4] <> 0 THEN
                stringResult.stationName := REPLACE (stringResult.stationName,'',LEN(blackListArr^.blackListArr[ii]),stringPlace[4]);
            END_IF
        END_FOR
        sqlVArInsert := stringResult;
```

Now we will use similar input which was manipulated with different id because there is no 1013 id in the database but this time we use the blackList method before inserting data. Therefore, we expect that we will insert data properly.

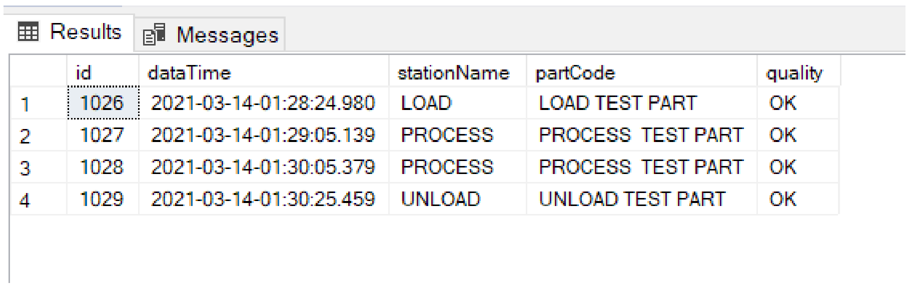


```
        80:
            myTask^.partStatus := WSTRING_TO_STRING("OK ');delete FROM myProduction where id = 1029 --") ;
            myTask^.stepA:=90;
```



```
        IF myTask^.partStatus <> '' THEN
            sqlVarInsert.stationName := myTask^.taskName;
            sqlVarInsert.partCode := CONCAT (myTask^.taskName,' Test Part');
            sqlVArInsert.quality := myTask^.partStatus;
            dataBaseManager.blackList();
            IF dataBaseManager.insertData() THEN
                myTask^.partStatus :='';
            END_IF;
        END_IF;
```

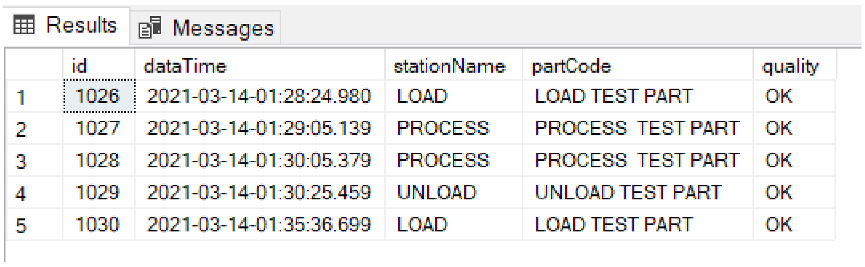
Before executing program, Database looked like;



Results | Messages

	id	dataTime	stationName	partCode	quality
1	1026	2021-03-14-01:28:24.980	LOAD	LOAD TEST PART	OK
2	1027	2021-03-14-01:29:05.139	PROCESS	PROCESS TEST PART	OK
3	1028	2021-03-14-01:30:05.379	PROCESS	PROCESS TEST PART	OK
4	1029	2021-03-14-01:30:25.459	UNLOAD	UNLOAD TEST PART	OK

After executing program, Database looks like;



	id	dataTime	stationName	partCode	quality
1	1026	2021-03-14-01:28:24.980	LOAD	LOAD TEST PART	OK
2	1027	2021-03-14-01:29:05.139	PROCESS	PROCESS TEST PART	OK
3	1028	2021-03-14-01:30:05.379	PROCESS	PROCESS TEST PART	OK
4	1029	2021-03-14-01:30:25.459	UNLOAD	UNLOAD TEST PART	OK
5	1030	2021-03-14-01:35:36.699	LOAD	LOAD TEST PART	OK

So we could insert data properly even if data was manipulated but if you want to use this method to prevent SQL injection, you need to use more proper list because I just created list for my own case, or if you know exactly which characters will be in use, then your program accepts only those characters but name of method should be white list, not black list.

I wish you all the best

Cheers

Murat TOPRAK

TwinCAT SQL Expert mode:

https://infosys.beckhoff.com/english.php?content=../content/1033/tf6420_tc3_database_server/2674388619.html&id