**DataBase Operation Manual**





**<DataBase V1.0>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Author** | Yan Rongbao | | | PC based iComputer Comes SINSEGYE  West Tower, Baidu International Building, Yuehai Subdistrict, Nanshan District, Shenzhen Website: [www.sinsegye.com.cn](http://www.sinsegye.com.cn)  E-mail: Sales@sinsegye.com.cn  Hotline: 400 - 013 - 2158 |
| **Date** | | Nov. 13, 2024 |  |
| **Version** | | V1.0 |  |
| Email | | yanrongbao@ sinsegye.com.cn |  |

Update Description

**2024/11/14 / V1.0**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Change Type** | **Description** | **Related Documents** |
| * Create a table * Read data in table * Add data * Update data in table * Retain the last few rows of the table | Initial release | / | / |

|  |  |
| --- | --- |
|  | **PC based iComputer Comes SINSEGYE** |

**DataBase Usage Introduction**

### Preface

#### I. Document Description

This description is specifically formulated for control and automation technology experts who are familiar with relevant national standards and have received professional training.  
When installing and debugging components, it is essential to carefully review all relevant documents and the following instructions.  
Qualified personnel should always apply the latest valid documents.  
Responsible personnel must ensure that the application or use of the products described fully complies with all safety requirements, covering all relevant laws, regulations, guidelines, and standards.

1. **Disclaimer**This document has been carefully compiled. However, given that the products described are in a continuous process of development and upgrading, Sinsegye (Shenzhen) Computer System Co., Ltd. reserves the right to modify and update the document at any time without prior notice. Please note that any modification to the delivered products based on the data diagrams and the description in this document is prohibited.  
   Sinsegye (Shenzhen) Computer System Co., Ltd. shall not be liable for any loss or damage caused by the use or reliance on the information stated or not explicitly stated in this manual.
2. **Copyright**The ownership of this manual belongs to Sinsegye (Shenzhen) Computer System Co., Ltd. Without written permission, no person shall reproduce, distribute, translate, or otherwise use all or part of this manual in any form.  
   This manual is protected by copyright law. Any reproduction, distribution, translation, display, performance, adaptation, or use of the content of this manual, for whatever purpose, requires the explicit permission of Sinsegye (Shenzhen) Computer System Co., Ltd. Without permission, any such act shall be regarded as an infringement of the copyright of Sinsegye (Shenzhen) Computer System Co., Ltd.

### II. Safety Statement

1. **Regulations on work safety**For your safety, please read the following instructions. You shall always abide by the specific regulations on work safety of the product, which you can find in the appropriate sections of this document.
2. **Liability exemption**All components are provided with hardware and software configurations. Modifications to hardware or software configurations other than those described in the document are not permitted. SINSEGYE shall not be responsible for hardware or software other than that described in the document.
3. **Personnel qualifications**This description is only applicable to trained experts in control, automation, and drive technologies who are familiar with the applicable national standards.
4. **Signal words**The signal words used in the document are classified as follows. To prevent harm to people and damage to properties, please read and comply with the safety and warning notices.
5. **Personal injury warnings**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | □ Warning | | **Type of Danger**  Describe the consequences of not avoiding the danger.  Describe how to prevent the occurrence of the danger. | | Warning indicates a potential hazardous situation which, if not avoided, could result in serious injury or death. |
| |  |  | | --- | --- | |  | □ Caution | | **Type of Danger**  Describe the consequences of not avoiding the danger.  Describe how to prevent the occurrence of the danger. | | Caution indicates a potential hazardous situation which, if not avoided, could result in minor or moderate injury, or damage to equipment. |
| |  | | --- | | **Remind** | | **Type of Danger**  Describe the consequences of not avoiding the danger.  Describe how to prevent the occurrence of the danger. | | Caution indicates a potential hazardous situation. If not avoided, it may only lead to damage to the equipment. |

1. **Warnings of damage to property or environment**

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | Caution | | **Type of Danger**  Describe the consequences of not avoiding the danger.  Describe how to prevent the occurrence of the danger. | | The environment, equipment, or data may be damaged. |

1. **Product disposal information**

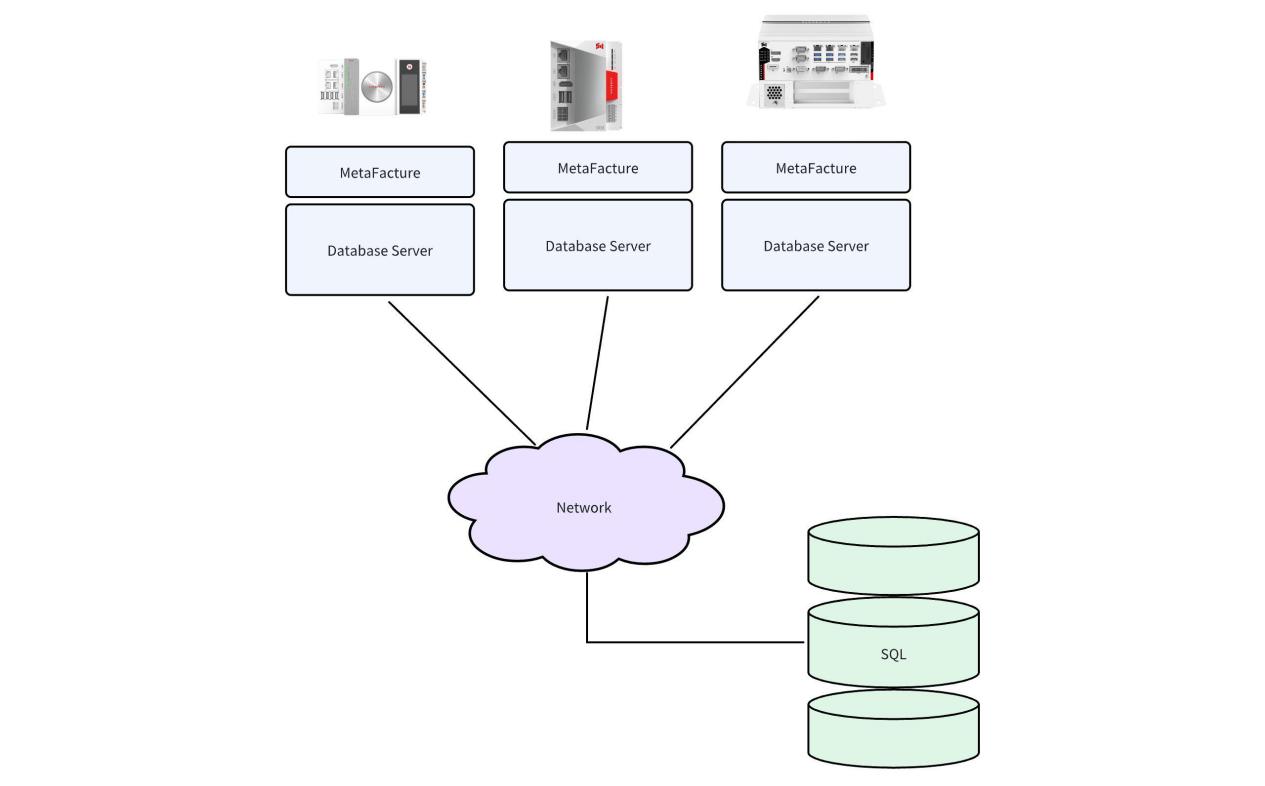
For example, this information includes: recommended actions, assistance, or further information about the product.

**Overview**

**Application Scenarios**

* Industrial automation: In scenarios such as production line monitoring, equipment condition monitoring, and fault diagnosis, PLCs (Programmable Logic Controllers) can effectively cooperate with databases.
* Predictive maintenance: Analyze historical data, carry out predictive maintenance of equipment and reduce downtime.

**Overall Architecture**



**Main Functions and Product Components**

* DataBase enables the interaction between PLC variables and the database through PLC programming. (Currently, it supports interaction with MySQL databases, and support for other types of databases will be added later.)
* Interaction functions supported: creating tables, reading data from tables, adding data, updating data in tables, and retaining the last few rows in a table. Support for other common database functions will be gradually added later,such as deleting specific rows of data from a table, inserting data, querying table names, querying table structures, and deleting tables.
* The DataBase product consists of three components:

(Note: You can download components to the corresponding modules from website of the company. The versions may be different from those shown below, but this will not affect the usage.)

|  |  |
| --- | --- |
| **Product components** | **Description** |
| sf4401db\_0.0.1-rc.4\_amd64.deb | sf4401db RTE component |
| dbservice.service | Database backend service |
| SF4401\_DB. library | The library file used by the host computer program |

**Installation and Uninstallation**

## Installation Requirements

* The built-in system of the iComputer manufactured by SINSEGYE.
* The iComputer has access to the Internet.

1. **Installation Process**

The DataBase product consists of three components. Therefore, to use DataBase, you need to install these three components firstly. Details are shown as follows:

### 1. Install DataBase RTE component on iComputer

* Upload the deb package to the /home/sinsegye directory in the Linux environment of iComputer,Refer to the appendix for the uploading method.
* Then, execute the command on iComputer to install deb package (refer to the screenshot below. If the module file name changes, make corresponding changes to the file name in the command line).

|  |
| --- |
| Shell cd $HOME sudo dpkg -i sf4401db\_0.0.1-rc.4\_amd64.deb |



* Modify the configuration file of RTE and add "sf4401db" under ComponentManger module, and at the same time, remove SE4100.

|  |
| --- |
| Shell sudo nano /usr/local/etc/SinsegyeRTE/SinsegyeRTE.cfg |

|  |
| --- |
| Shell [ComponentManager]  Component.0=retainDeamon  Component.1=CmpCanBusUtils  Component.2=CmpSinsegyeLibs  Component.3=SinsegyeCmp  Component.4=sf4401db |

* Restart RTE service to call sf4401db.

|  |
| --- |
| Shell sudo systemctl restart sinsegyerte.service |

**2. Deploy the database background service component on iComputer**

* Firstly, deploy the .Net SDK 6.0.0 and execute the following commands while your device is connected to the Internet.

|  |
| --- |
| Shell wget https://packages.microsoft.com/config/ubuntu/22.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb # If the operating system version of iComputer is not 22.04, please replace 22.04 in the first command with the corresponding version. sudo dpkg -i packages-microsoft-prod.deb sudo rm -rf packages-microsoft-prod.deb sudo apt-get update sudo apt-get install -y dotnet-sdk-6.0 sudo apt-get install -y aspnetcore-runtime-6.0 sudo apt-get install -y dotnet-runtime-6.0 sudo cp -r /usr/lib/dotnet/host/ /usr/share/dotnet/ # This may not be required for versions other than 22.04.  sudo cp -r /usr/lib/dotnet/shared/ /usr/share/dotnet/ # This may not be required for versions other than 22.04. dotnet --info # Check the net service information. No error message indicates a successful installation. |

* Upload the database background service package to the /home/sinsegye directory of iComputer (refer to the appendix for the uploading method), and then extract it.

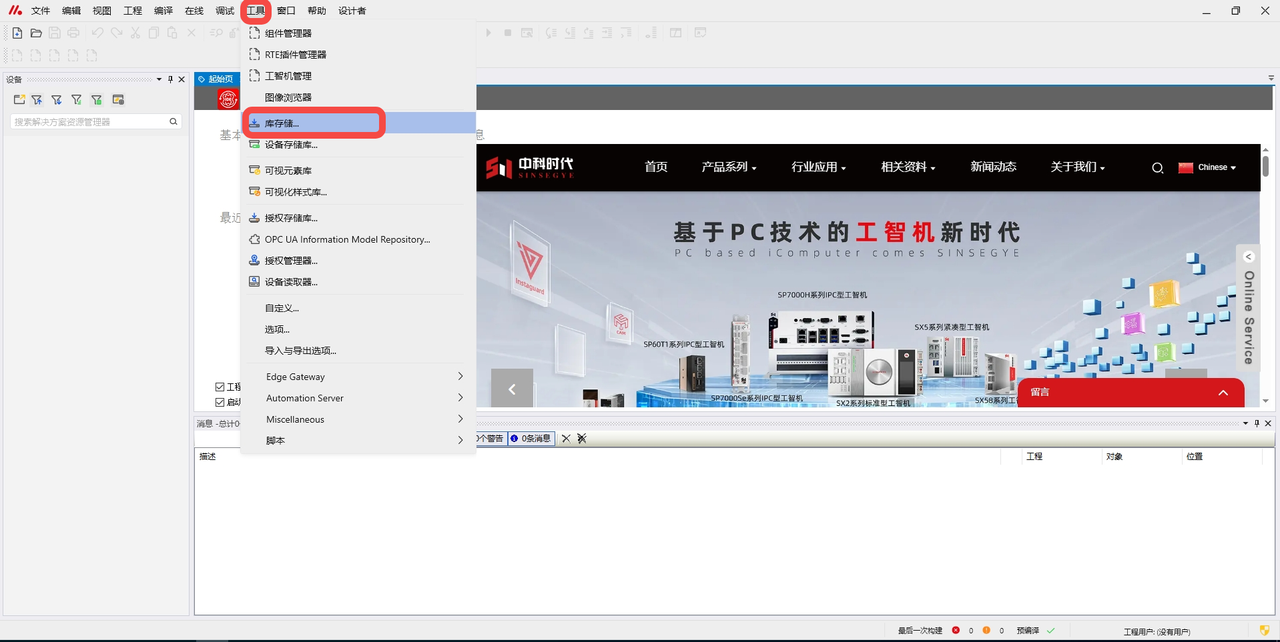
|  |
| --- |
| Shell sudo unzip dbservice.zip |

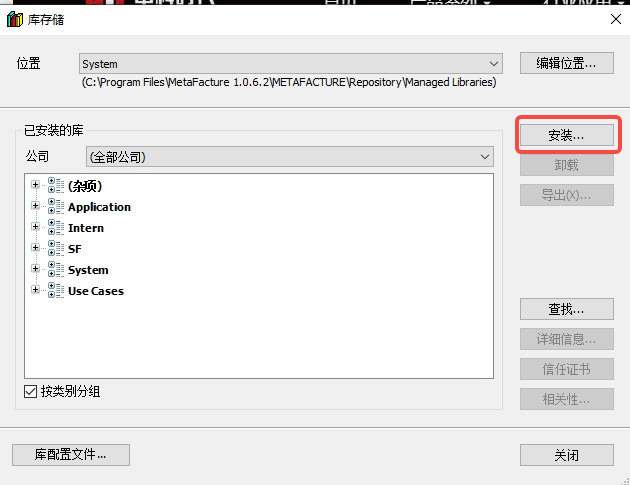
* Upload the dbservice.service file to the /home/sinsegye directory (refer to the appendix for the uploading method), and then execute the following commands:

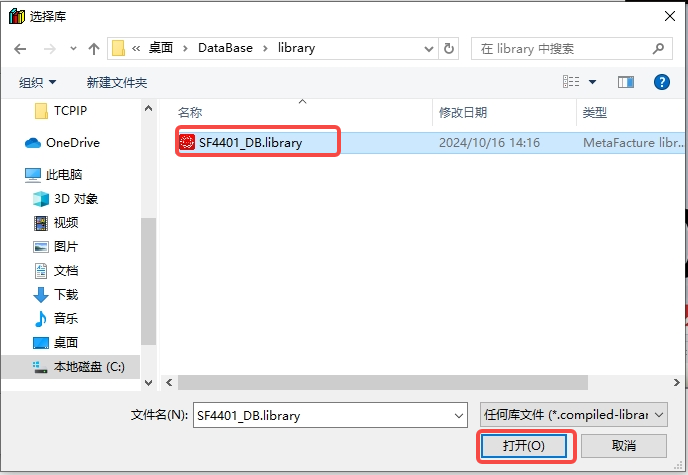
|  |
| --- |
| Shell cd $HOME sudo mv dbservice.service /lib/systemd/system/ sudo systemctl daemon-reload sudo systemctl status dbservice.service sudo systemctl start dbservice.service sudo systemctl enable dbservice.service |

**3.Deploy the library of DataBase on IDE side**

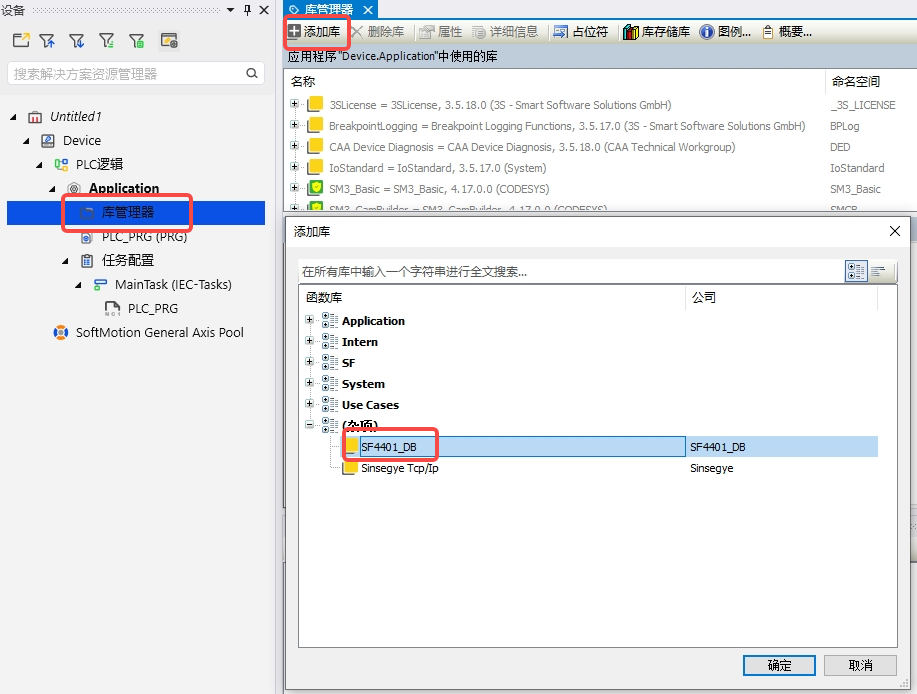
* In MetaFacture, click “Tools" -- "Library Storage" in the menu bar on the top;



* In the dialog box, click "Install", then select "SF4401\_DB. library ", and finally click "Open".



* In the project, double click "Library Manager" -- "Add Library" -- double click "SF4401\_DB" to load the library.



## Update Installation

1. **Upgrade database RTE Component on iComputer**

* Upload the upgraded deb package to the /home/sinsegye directory in the Linux environment of iComputer. Refer to the appendix for the upload method.
* Then, execute the installation command on iComputer (refer to the screenshot below. If the module file name changes, make corresponding changes to the file name in the command line).

|  |
| --- |
| Shell cd $HOME sudo dpkg -i sf4401db\_0.0.1-rc.4\_amd64.deb |

* Restart RTE service to call sf4401db.

|  |
| --- |
| Shell sudo systemctl restart sinsegyerte.service |

1. **Upgrade the database background service of iComputer**

* Upload the upgrade package of the database background service to the /home/sinsegye directory of iComputer (refer to the appendix for the uploading method), and then extract it.

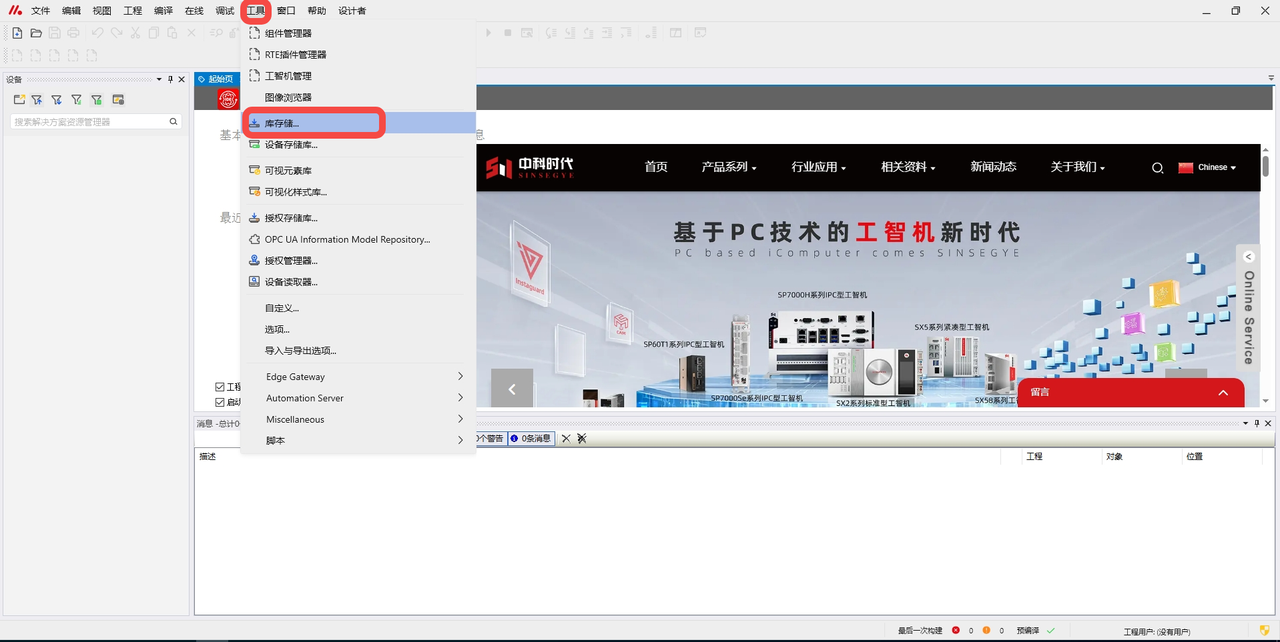
|  |
| --- |
| Shell sudo unzip dbservice.zip |

* Upload the new dbservice.service file to the /lib/systemd/system/ directory (refer to the appendix for the uploading method), and then execute the following commands.

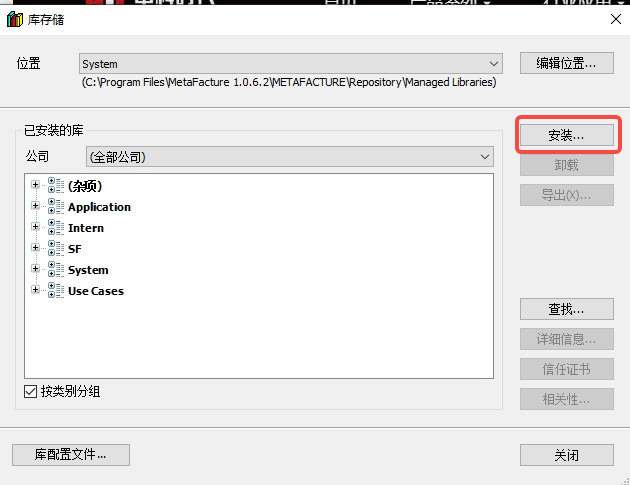
|  |
| --- |
| Shell cd $HOME sudo mv dbservice.service /lib/systemd/system/ sudo systemctl daemon-reload sudo systemctl status dbservice.service sudo systemctl start dbservice.service sudo systemctl enable dbservice.service |

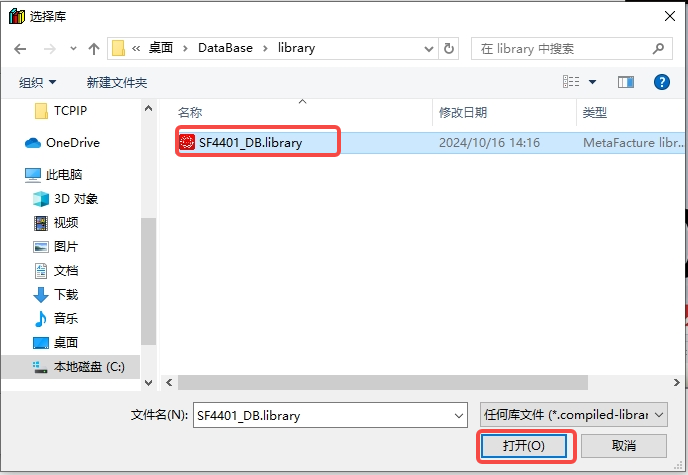
1. **Upgrade the library of DataBase on IDE side**

* In MetaFacture, click "Tools" -- "Library Storage" in the menu bar on the top.

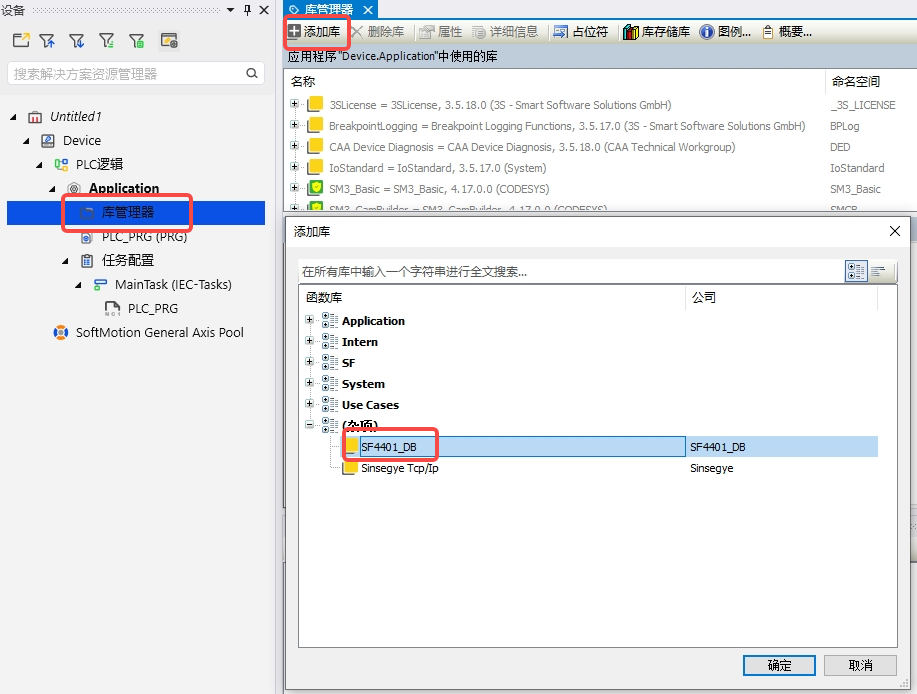


* In the dialog box, click "Install", select "SF4401\_DB. library" file, and then click "Open".





* In the project, double click the "Library Manager" -- "Add Library" and then double click "SF4401\_DB" to load the library.



## Uninstallation Process

### Uninstall database RTE Component on iComputer

* Execute the uninstallation command on iComputer.

|  |
| --- |
| Shell sudo dpkg -r sf4401db |

* Modify RTE configuration file and remove sf4401\_db from ComponentManger module.

|  |
| --- |
| Shell sudo vim /usr/local/etc/SinsegyeRTE/SinsegyeRTE.cfg |

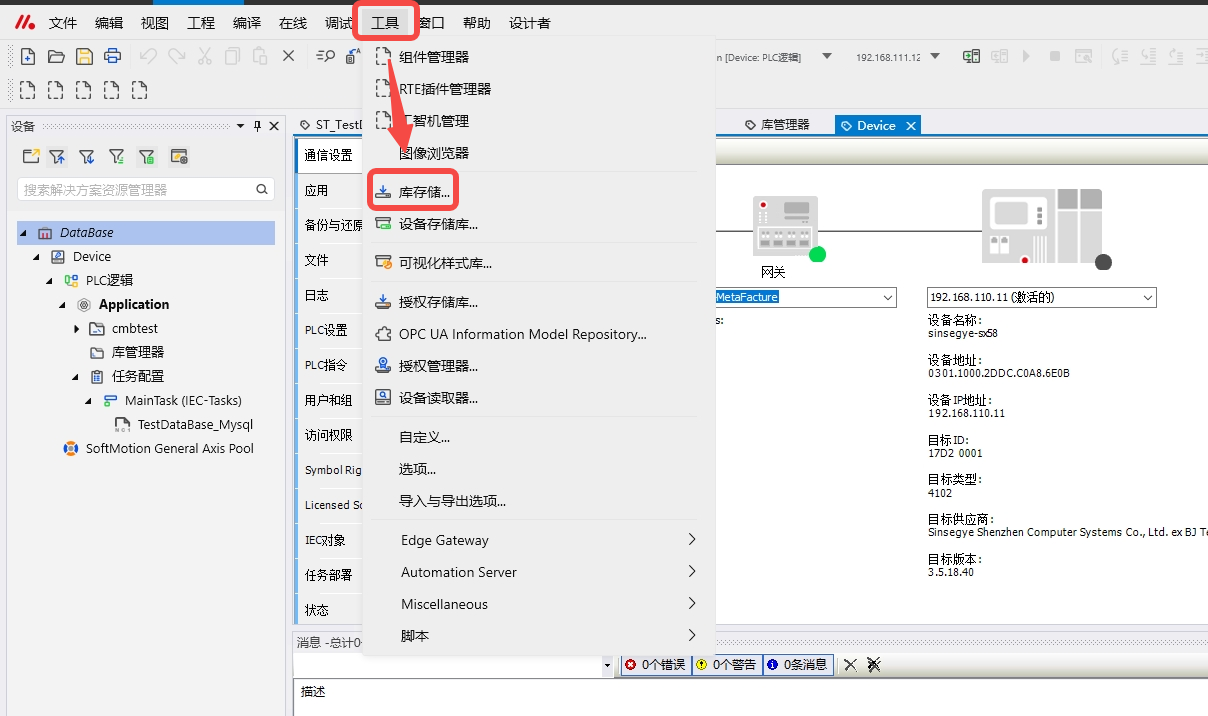
1. **Uninstall the database background service on iComputer**

* Execute the following commands on iComputer.

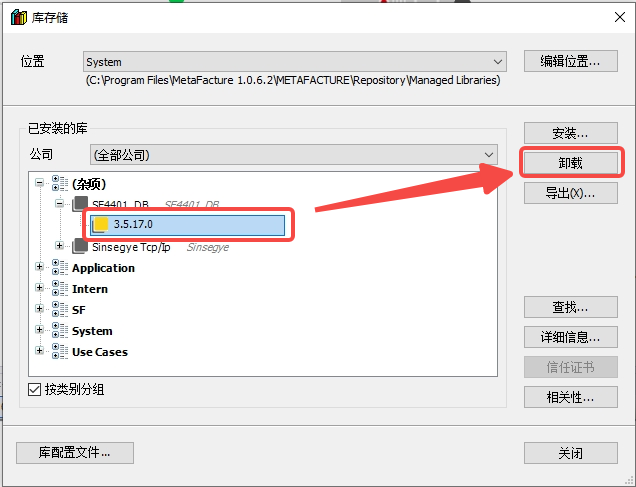
|  |
| --- |
| Shell sudo systemctl stop dbservice.service sudo systemctl disable dbservice.service sudo rm -rf /lib/systemd/system/dbservice.service sudo systemctl daemon-reload sudo rm -rf /home/sinsegye/dbservice/ |

1. **Uninstall DataBase library on IDE side**

* In the MetaFacture interface, click "Tools" -- "Library Storage".



* In the dialog box, select the installed sf4401 library and click "Uninstall"



### 

### Technical Instructions

#### Quick Start

**(1) Software and Hardware Configuration for This Example**

|  |  |
| --- | --- |
| **Hardware**   1. SX58 iComputer 2. win10 PC | **Software**   1. MetaFacutre V1.0.6.2 |

1. **Steps of experimental operations in this example**

Experimental requirement: Configure the DataBase environment according to the "Installation Process" in the "Installation and Uninstallation" section, and there is a "test1" database in the mysql database.

1. **The experimental steps for establishing connection to the database are as follows:**

**1.1 Experimental steps**

* Call the function block FB\_PLCDBAddConnection of SF4401\_DB library in the declaration area of POU.

|  |
| --- |
| SQL  fbCreateCon : SF4401.FB\_PLCDBAddConnection;  sdbs : STRING := 'test1'; //Define the name of the database test1  addcon\_execute：bool; |

* In the program area of POU, call fbCreateCon to configure the IP address, port, name of the database, as well as the username and password for accessing the database.

|  |
| --- |
| SQL fbCreateCon(sNetID:= '192.168.111.152:3306', //mysql The IP address and port of the database.  eDBType:= 0, //0 It represents the database type. mysql  sDBServer:= sdbs, //The name of the database.  sDBUserId:= 'remoteuser',   sDBPassword:= 'password',   bExecute:= addcon\_execute,   tTimeout:= T#5S, bBusy=> ,   bError=> ,   sDBID=> DBID,   sMessage=> ); |

* Trigger the rising edge of bExecute to establish connection with the database.

1. **The experimental steps for obtaining the database ID are as follows:**

Experimental requirement: conduct experiment 1 first to establish connection with mysql database.

* In the declaration area of POU, call the function block FB\_PLCDBGetConnection of the SF4401\_DB library.

|  |
| --- |
| SQL fbGetCon: SF4401.FB\_PLCDBGetConnection; sdbs: STRING: = 'test1’; //Define the name of the database test1 get\_execute: Bool; DBID: STRING: = '‘; //Used to store the acquired database ID |

* In the program area of POU, call fbGetCon

|  |
| --- |
| SQL fbGetCon(sNetID:= '192.168.111.152:3306',   name:= sdbs,   eDBType:= 0,   bExecute:= get\_execute,   tTimeout:= T#5S,   bBusy=> ,   bError=> ,   sDBID=> DBID,  sMessage=>); |

* Trigger the rising edge of bExecute to obtain the ID of the data

1. **The test steps for creating a table are as follows:**

Experimental requirement: You need to conduct Experiment 2 first to obtain the database ID.

* In the declaration area of POU, call the function block FB\_PLCDBDeleteConnection from the SF4401\_DB library, and use ST\_ColumnInfo to define the column information of the table.

|  |
| --- |
| SQL fbCreateTable : SF4401.FB\_PLCDBCreateTable; ColumnInfo : ARRAY[0..255] OF SF4401.ST\_ColumnInfo; table\_name : STRING := 'test1.dbtest\_table1'; //Name of the table to be created |

* In the program area of POU, define the name and attributes of each column of the table, and call fbCreateTable.

|  |
| --- |
| SQL ColumnInfo[0].sName := 'Long\_test';ColumnInfo[0].eType := SF4401.E\_ColumnType.Long; ColumnInfo[0].nLength := 8;  ColumnInfo[1].sName := 'Int\_test';ColumnInfo[1].eType := SF4401.E\_ColumnType.Integer; ColumnInfo[1].nLength := 4; ColumnInfo[2].sName := 'Bool\_test';ColumnInfo[2].eType := SF4401.E\_ColumnType.Bool\_; ColumnInfo[2].nLength := 1; ColumnInfo[3].sName := 'Long\_test2';ColumnInfo[3].eType := SF4401.E\_ColumnType.Long; ColumnInfo[3].nLength := 8; ColumnInfo[4].sName := 'String\_Test';ColumnInfo[4].eType := SF4401.E\_ColumnType.STRING\_; ColumnInfo[4].nLength := 81; ColumnInfo[5].sName := 'Double\_test';ColumnInfo[5].eType := SF4401.E\_ColumnType.Double; ColumnInfo[5].nLength := 8; ColumnInfo[6].sName := 'Double\_test2';ColumnInfo[6].eType := SF4401.E\_ColumnType.Double; ColumnInfo[6].nLength := 8; ColumnInfo[7].sName := 'Time\_test';ColumnInfo[7].eType := SF4401.E\_ColumnType.DateTime; ColumnInfo[7].nLength := 8; //Create the defined table fbCreateTable(sDBID:= DBID, //The ID of the data acquired in Experiment 2  sTableName:= table\_name, //Name of the table  aTableCfg:= ColumnInfo,   cbTableCfg:= 8,   bExecute:= table\_execute,   tTimeout:= T#5S,   bBusy=> ,   bError=> ,  sMessage=>); |

* Trigger the rising edge of bExecute, to create the table dbtest\_table1 in the database test1.

1. **Write data into the table**

Experimental requirement: firstly, conduct Experiment 3 to create the table `dbtest\_table1`.

* In the declaration area of POU, call the function block SF4401.FB\_PLCDBWrite from the SF4401\_DB library, and define the column information of the table to be written to.

|  |
| --- |
| SQL fbWrite : SF4401.FB\_PLCDBWrite; writeColumnNames: ARRAY [0..255] OF STRING(50); writeStructArr : ARRAY [0..255] OF ST\_TestDataBase; |

* ST\_TestDataBase is a predefined structure that defines the column names and type information for each column of the table.

|  |
| --- |
| SQL TYPE ST\_TestDataBase : STRUCT Long\_test : LINT; Int\_test : DINT; Bool\_test : BOOL; Long\_test2 : LINT; String\_Test : STRING; Double\_test : LREAL; Double\_test2 : LREAL; Time\_test : DATE\_AND\_TIME; END\_STRUCT END\_TYPE |

* In the program area of POU, set the column name information of the table, then assign values to each column, and then call fbWrite to trigger the writing action.

|  |
| --- |
| SQL writeColumnNames[0] := 'Long\_test'; writeColumnNames[1] := 'Int\_test'; writeColumnNames[2] := 'Bool\_test'; writeColumnNames[3] := 'Long\_test2'; writeColumnNames[4] := 'String\_Test'; writeColumnNames[5] := 'Double\_test'; writeColumnNames[6] := 'Double\_test2'; writeColumnNames[7] := 'Time\_test';  writeStructArr[0].Long\_test := 110; writeStructArr[0].Int\_test := 111; writeStructArr[0].Bool\_test := FALSE; writeStructArr[0].Long\_test2 := 115; writeStructArr[0].String\_Test := 'asdsadsada'; writeStructArr[0].Double\_test := 101.0; writeStructArr[0].Double\_test2 := 100.0; writeStructArr[0].Time\_Test := DT#2024-10-22-13:10:00;  writeStructArr[1].Long\_test := 216; writeStructArr[1].Int\_test := 215; writeStructArr[1].Bool\_test := TRUE; writeStructArr[1].Long\_test2 := 200; writeStructArr[1].String\_Test := 'gggsdfsfdsf'; writeStructArr[1].Double\_test := 200.0; writeStructArr[1].Double\_test2 := 200.0; writeStructArr[1].Time\_Test := DT#2024-10-22-13:13:00;  writeStructArr[2].Long\_test := 214; writeStructArr[2].Int\_test := 215; writeStructArr[2].Bool\_test := TRUE; writeStructArr[2].Long\_test2 := 200; writeStructArr[2].String\_Test := 'gggsdfsfdsf'; writeStructArr[2].Double\_test := 200.0; writeStructArr[2].Double\_test2 := 200.0; writeStructArr[2].Time\_Test := DT#2024-10-21-12:12:12;  fbWrite(sDBID:= DBID,   sTableName:= table\_name,   sColumnNames:= writeColumnNames,   sOrderByColumn:=writeColumnNames[0],   pRecord:= ADR(writeStructArr),   cbRecord:= SIZEOF(ST\_TestDataBase),   dataCount:= 3,   eDBWriteMode:= SF4401.E\_WriteMode.DB\_Append, //Usage of Append Mode  //eDBWriteMode: = SF4401.E\_WriteMode.DB\_RingBuff\_count // Usage of RingBuff\_count Mode  nRingBuffParameter: = 10,   bExecute:= write\_append\_execute,   tTimeout:= T#5S,   bBusy=> ,   bError=> ,  sMessage=>); |

• Trigger the rising edge of bExecute to write data in dbtest\_table1.

1. **Read the data in the table**

Experimental requirement: You need to conduct Experiment 3 and Experiment 4 first to create dbtest\_table1 and write data.

* In the declaration area of POU, call the function block FB\_PLCDBReadStruct from the SF4401\_DB library. Define the column information ColumnNames of the table to be read, and define the variable myCustomStructArr for storing the data.

|  |
| --- |
| SQL fbReadStruct : SF4401.FB\_PLCDBReadStruct; ColumnNames: ARRAY [0..255] OF STRING(50); myCustomStructArr : ARRAY [0..3] OF ST\_TestDataBase; |

* In the program area of POU, specify the column name information of the table to be read, and then call fbReadStruct.

|  |
| --- |
| SQL ColumnNames[0] := 'Long\_test'; ColumnNames[1] := 'Int\_test'; ColumnNames[2] := 'Bool\_test'; ColumnNames[3] := 'Long\_test2'; ColumnNames[4] := 'String\_Test'; ColumnNames[5] := 'Double\_test'; ColumnNames[6] := 'Double\_test2'; ColumnNames[7] := 'Time\_Test';  fbReadStruct(   sDBID:= DBID,   sTableName:= table\_name,   sColumnNames:= ColumnNames,   sOrderByColumn:= 'id',   eOrderType:= SF4401.E\_OrderType.ASC,   nStartIndex:= 1,   nRecordCount:= 2,   pData:= ADR(myCustomStructArr),   cbData:= SIZEOF(ST\_TestDataBase),   bExecute:= read\_execute,   tTimeout:= T#5S,   bBusy=> ,   bError=> ,  sMessage=>); |

* Trigger the rising edge of bExecute to read data in dbtest\_table1 which is saved in myCustomStructArr.

1. **Disconnect from MySQL database**

Experimental requirement: first conduct Experiment 1 to establish connection with database and obtain the database ID information.

* In the declaration area of POU, call the function block FB\_PLCDBDeleteConnection from SF4401\_DB library.

|  |
| --- |
| SQL fbDelCon: SF4401.FB\_PLCDBDeleteConnection; DBID : STRING :=''; delecon\_execute : Bool; |

* In the program area of POU, call fbDelCon and input the IP address, port number, and ID of the database.

|  |
| --- |
| SQL fbDelCon(  sNetID:= '192.168.111.152:3306',   sDBID:= DBID,   bExecute:= delcon\_execute,   tTimeout:= T#5S,   bBusy=> ,   bError=> ,  sMessage=>); |

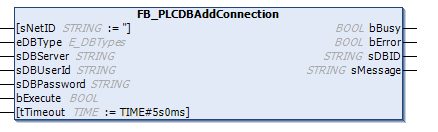
**(III) Experimental precautions**

* Some experiments have sequential requirements. It is recommended to conduct the experiments in the order specified in the documentation.

**Function introduction**

1. **Establish connection with database**

**(I) Introduction to Function Block FB\_PLCDBAddConnection**



**(II) Parameter Introduction**

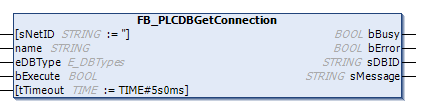
* Input parameters

|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| sNetID | String | IP address and port of the database |
| eDBType | Int | Database type, 0 represents MySQL, 1 represents SQL, 2 represents PostgreSQL, 3 represents Oracle, 4 represents MongoDB, and 5 represents DB2 |
| sDBServer | String | Database name |
| sDBUserId | String | Database username |
| sDBPassword | String | Database password |
| bExecute | Bool | The rising edge triggers the execution |
| tTimeout | Time | Timeout settings |

* Output parameters

|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| bBusy | Bool | Busy status, True is busy, False is not busy |
| bError | Bool | Whether to report an error |
| sDBID | String | Database ID |
| sMessage | String | Error message |

1. **Obtain the Database ID**
2. **Introduction to the Function Block FB\_PLCDBGetConnection**



**(II) Parameter Introduction**

* Input parameters

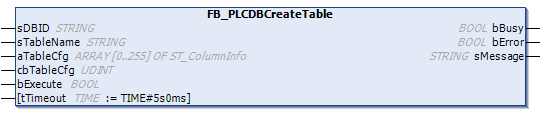
|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| sNetID | String | IP address and port of the database |
| name | String | The name of the database |
| eDBType | Int | Database type, 0 represents MySQL, 1 represents SQL, 2 represents PostgreSQL, 3 represents Oracle, 4 represents MongoDB, and 5 represents DB2 |
| bExecute | Bool | The rising edge triggers the execution |
| tTimeout | Time | Timeout settings |

* Output parameters

|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| bBusy | Bool | Busy status, True is busy, False is not busy |
| bError | Bool | Whether to report an error |
| sDBID | String | Database ID |
| sMessage | String | Error message |

1. **Create Table**

**(I) Introduction to Function Block FB\_PLCDBCreateTable**



**(II) Parameter Introduction**

* Input parameters

|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| sDBID | String | Database connection id |
| sTableName | String | The name of table created |
| aTableCfg | ARRAY [0..255] OF SF4401.ST\_ColumnInfo | Column information of table |
| cbTableCfg | Udint | Number of columns in the table |
| bExecute | Bool | The rising edge triggers the execution |
| tTimeout | Time | Timeout settings |

* Output parameters

|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| bBusy | Bool | Busy status, True is busy, False is not busy |
| bError | Bool | Whether to report an error |
| sMessage | String | Error message |

1. **Writing data in the database**

**(I) Introduction to Function Block FB\_PLCDBWrite**



**(II) Parameter Introduction**

* Input parameters

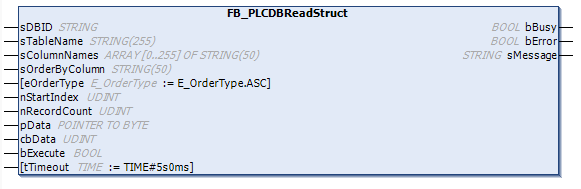
|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| sDBID | String | Database connection id |
| sTableName | String (255) | Name of the table created |
| sColumnNames | String (50) | Table column names |
| sOrderByColumn | String (50) | Sort by column |
| pRecord | Pointer to byte | Data source to write to |
| cbRecord | UDint | Data source length |
| datacount | UDint | Number of rows to write |
| eDBWriteMode | Int | Three types of write mode settings: DB\_Append means appending data, DB\_Update means updating existing data, and DB\_RingBuff\_count means retaining a specific number of the last rows of data in the table. |
| nRingBuffParameter | UDint | When used in DB\_RingBuff\_count mode, it represents the number of rows to be retained. |
| tTimeout | Time | Timeout |
| bExecute | Bool | The rising edge triggers the execution |

* Output parameters

|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| bBusy | Bool | Busy status, True is busy, False is not busy |
| bError | Bool | Whether to report an error |
| sMessage | String | Error message |

1. **Read data from database**

**(I) Introduction to Function Block FB\_PLCDBReadStruct**



**(II) Parameter Introduction**

* Input parameters

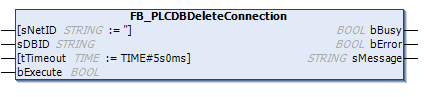
|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| sDBIDs | String | Database connection id |
| sTableName | String (255) | Table name |
| sColumnNames | String (50) | Table column names |
| sOrderByColumn | String (50) | Sort by column |
| eOrderType | Int | Sorting methods include ASC for ordering and DESC for reverse ordering |
| nStartIndex | UDint | The number of starting rows, starting at 0 |
| nRecordCount | UDint | Number of rows read |
| pData | Pointer to byte | Data destination position |
| cbData | UDint | The length of the data destination |
| bExecute | Bool | The rising edge triggers the execution |
| tTimeout | Time | Timeout |

* Output parameters

|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| bBusy | Bool | Busy status, True is busy, False is not busy |
| bError | Bool | Whether to report an error |
| sMessage | String | Error message |

1. **Disconnect from database**

**(I) Introduction to Function Block FB\_PLCDBDeleteConnection**



**(II) Parameter Introduction**

* Input parameters

|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| sNetID | String | IP address and port of the database |
| sDBID | String | Database connection id |
| tTimeout | Time | Timeout |
| bExecute | Bool | The rising edge triggers the execution |

* Output parameters

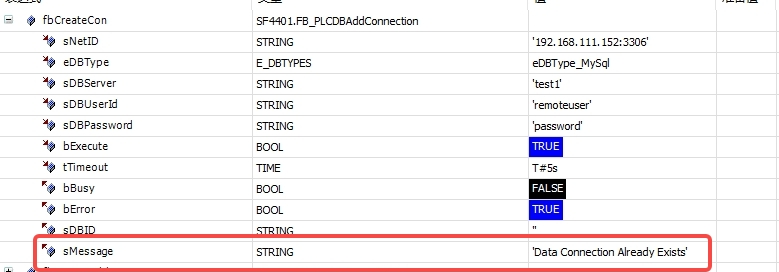
|  |  |  |
| --- | --- | --- |
| Parameter name | Parameter type | Description |
| bBusy | Bool | Busy status, True is busy, False is not busy |
| bError | Bool | Whether to report an error |
| sMessage | String | Error message |

**Appendix**

**I. Error Diagnosis**

**(I) Connection Abnormality**

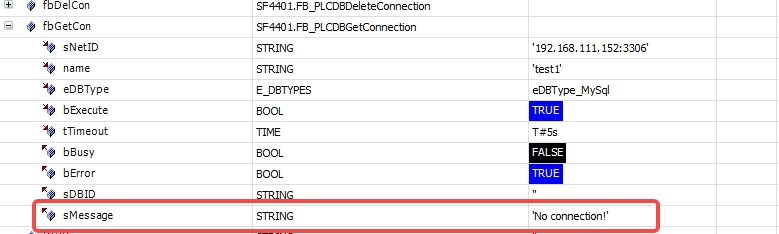
* The error message is as follows



* Analysis 1: If the connection to the same database was not closed previously, this error message will be prompted when attempting to reconnect. You need to first obtain the Database ID, then close the connection, and finally you can obtain a new connection.
* Analysis 2: Previously, a database of a different type with the same name was connected and the connection was not closed. You need to first connect to the previously connected database with the same name and close the connection, and then connect to the current database.

1. **Abnormal to get ID**

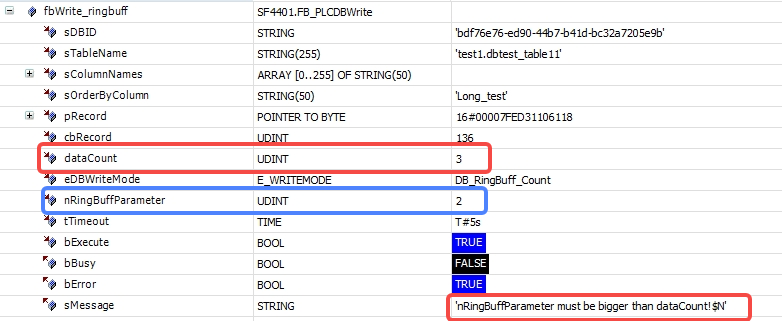
* The error message is as follows



* Handling method: There is no connection established with the database. You need to establish a connection first and then obtain the database ID.

1. **DB\_Ringbuff\_count mode exception**

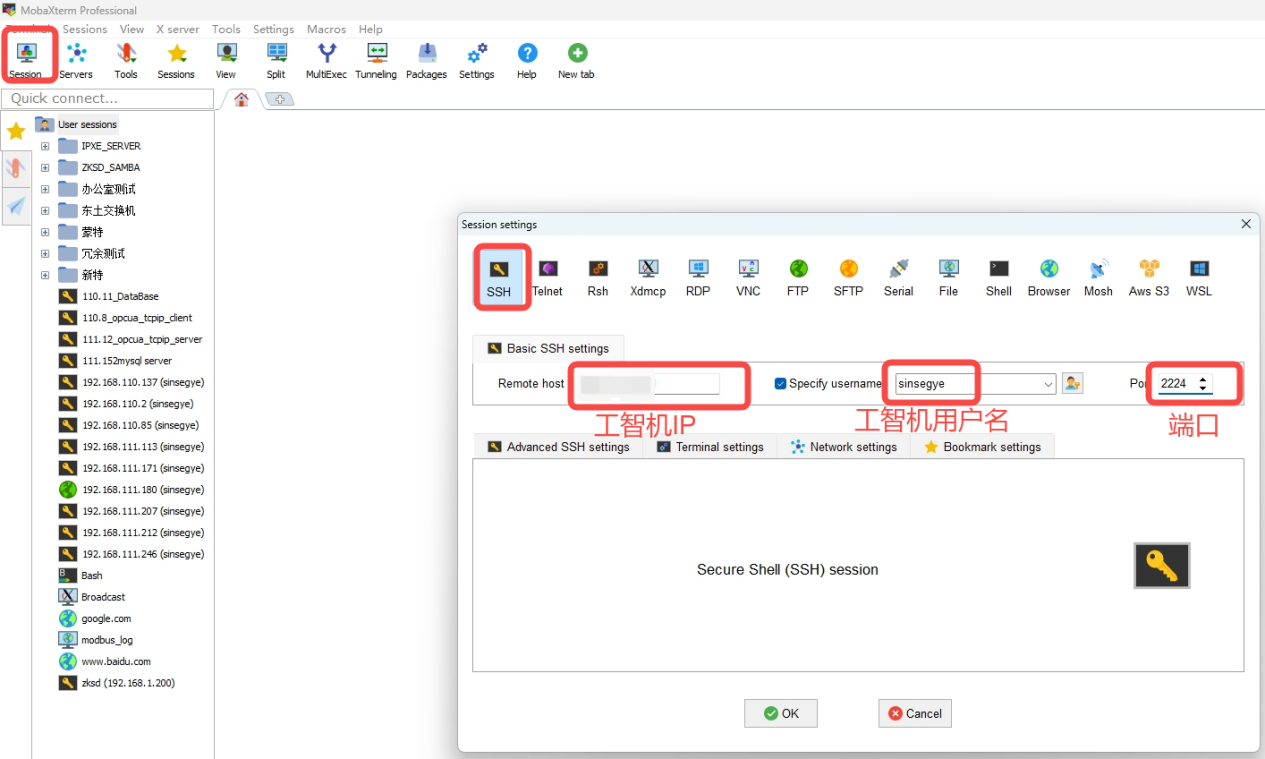
* The error message is as follows



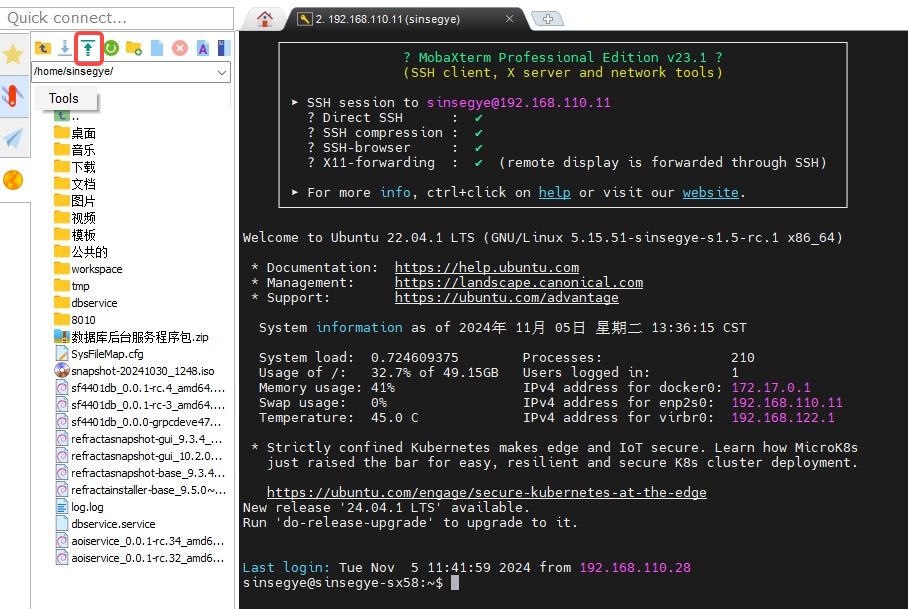
* Analysis: In RingBuff\_count mode, the number of rows datacount set for writing shall be less than the number of rows nRingBuffParameter to be retained.

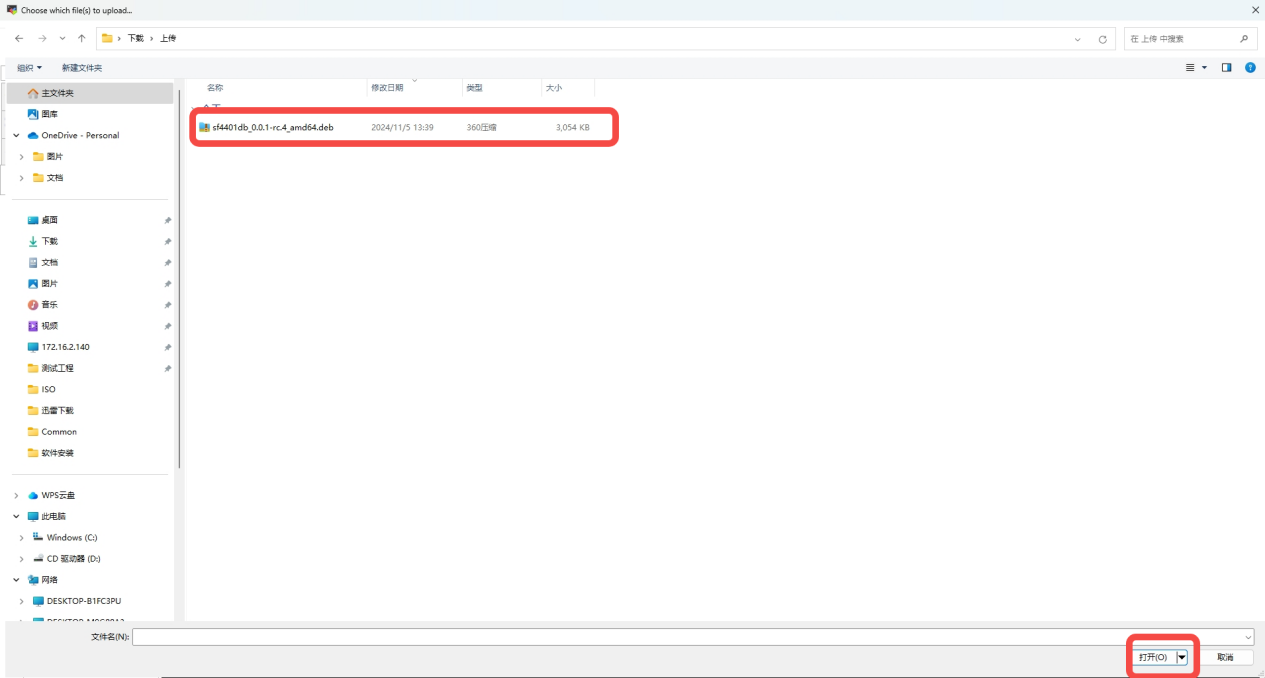
**II. Methods for uploading files to iComputer**

* First, use the Mobaxterm tool to remotely connect to iComputer: Click "Session" -- "SSH", then enter the IP address, username, and port of iComputer, and click "OK".

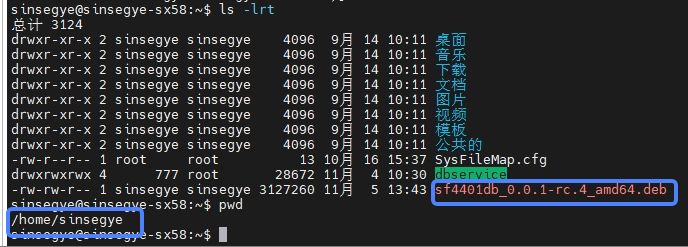


* After successful connection, click the upload button, select the file to be uploaded, and then click "Open";





* The uploaded file will be in the `/home/sinsegye` directory of iComputer by default.



## Support and Services

Sinsegye provides comprehensive support and services for its products and solutions, ensuring a rapid and professional response to relevant issues.

### Download Data

Our data download section offers varieties of document resources, including application cases, technical documents, product introductions, etc., to meet your diverse needs.  
Data download address: <https://help.sinsegye.com.cn/>

### Get Support

To get local support and services for Sinsegye products, you can visit our website [www.sinsegye.com.cn](https://www.sinsegye.com.cn/) to find the addresses of Sinsegye's branches and obtain more information about Sinsegye.  
Other contact information is shown as follows:

* Hotline: 400 - 013 - 2158
* Email address: support@sinsegye.com.cn