DEX-100

Software User Manual for Modules (DDS, REST, Modbus) and Edge Server

Version 1.1

Revision History

Revision	Date	Changes	Prepared by
0.1	08/23/2017	Create draft document	Steven Tu
0.2	09/19/2017	New add installation, DDS, Modbus and Edge Server chapter	Steven Tu
0.3	10/23/2017	New add Rest new API and Fix some picture error	Steven Tu
0.4	11/15/2017	1.Support JSON format on Rest API 2.Add Java and Python post function sample	Steven Tu
0.5	06/12/2018	New REST API to get the DEX-Pro status	Steven Tu
0.6	11/20/2018	Fix the post function example	Steven Tu
0.7	03/28/2018	Fix the duplicated section 3.1.4	Steven Tu
1.0	10/01/2019	Add the access OCR table, execute/stop script and keep the message data rest API	Steven Tu
1.1	09/07/2020	Add some limitation note for OCR restful API	

List of Contents

1. INTRODUCTION 5

2. INSTALLATION 5

2.1 DEX-100 VORTEX OPENSPLICE 5
2.2 EDGE SERVER VORTEX OPENSPLICE (THE DEFAULT IMAGE IS ALREADY INCLUDED) 9

3. REST MODULE 11

3.1 REST API 11
3.1.1 LIGHTCOLOR 11
3.1.1.1 LIGHTCOLOR XML FORMAT 11
3.1.1.2 LIGHTCOLOR JSON FORMAT 12
3.1.2 WARNINGMSG 13
3.1.2.1 WARNINGMSG XML FORMAT 13
3.1.2.1 WARNINGMSG JSON FORMAT 15
3.1.3 ALARMMSG 17
3.1.3.1 ALARMMSG XML FORMAT 17
3.1.3.2 ALARMMSG JSON FORMAT 19
3.1.4 MACHINESTATUS 21
3.1.4.1 MACHINESTATUS XML FORMAT 21
3.1.4.2 MACHINESTATUS JSON FORMAT 22
3.1.5 SPTSTATUS 23
3.1.5.1 SPTSTATUS XML FORMAT 23
3.1.5.2 SPTSTATUS JSON FORMAT 24
3.1.6 SYSMODE 25
3.1.6.1 SYSMODE XML FORMAT 25
3.1.6.2 SYSMODE JSON FORMAT 26
3.1.7 MESSAGE 27
3.1.7.1 MESSAGE JSON FORMAT 27
3.1.8 SCRIPT 28
3.1.8.1 EXECUTESCRIPT JSON FORMAT 28
3.1.8.2 STOPSCRIPT JSON FORMAT 29
3.1.9 OCR TABLE VALUE 29
3.1.9.1 SETOCRDATA JSON FORMAT 29
3.1.9.2 GETOCRDATA JSON FORMAT 30
3.2 WRITING WEB SERVICES CLIENT APPLICATIONS 31
3.2.1 C# 31
3.2.2 JAVA 32
3.2.3 PYTHON 32

4. DDS MODULE 34

4.1 DDS IDL 34

5. MODBUS MODULE 37

5.1 MODBUS FUNCTION 37
5.1.1 READ COIL STATUS 37
5.1.2 READ INPUT STATUS 39
5.1.3 READ HOLDING REGISTERS 41
5.1.4 READ INPUT REGISTERS 42
5.1.5 FORCE SINGLE COIL 44
5.1.6 PRESET SINGLE REGISTER 46
5.1.7 FORCE MULTIPLE COILS 48
5.1.8 FORCE MULTIPLE COILS 50
5.2 MODBUS REST API ERROR MESSAGE REFERENCE 52

6. EDGE SERVER 53

6.1 STARTING EDGE SERVER PROGRAM 53

6.2 DATABASE 53

1. Introduction

There are three modules as services in DEX-100. They are DDS, REST and Modbus modules. We simply go through the three modules main purposes as follows. Every modules are introduced in detail in the later chapters.

- DDS Module: It accepts the DEX-100 main program data and publish the four topics data on DDS.
- REST Module: The same, it accepts the DEX-100 main program data and provides the rest api to let user program to get the lastest data from DEX-100.
- Modbus Module: It is the modbus master, provides the modbus tcp and rtu for FC1,2,3,4,5,6,15,16 and also supports swap word and registers functions to get the modbus slave data and reply to DEX-100 main program.

For the Edge Server, it is a composite of main program and MySQL database. The main purposes are as follows.

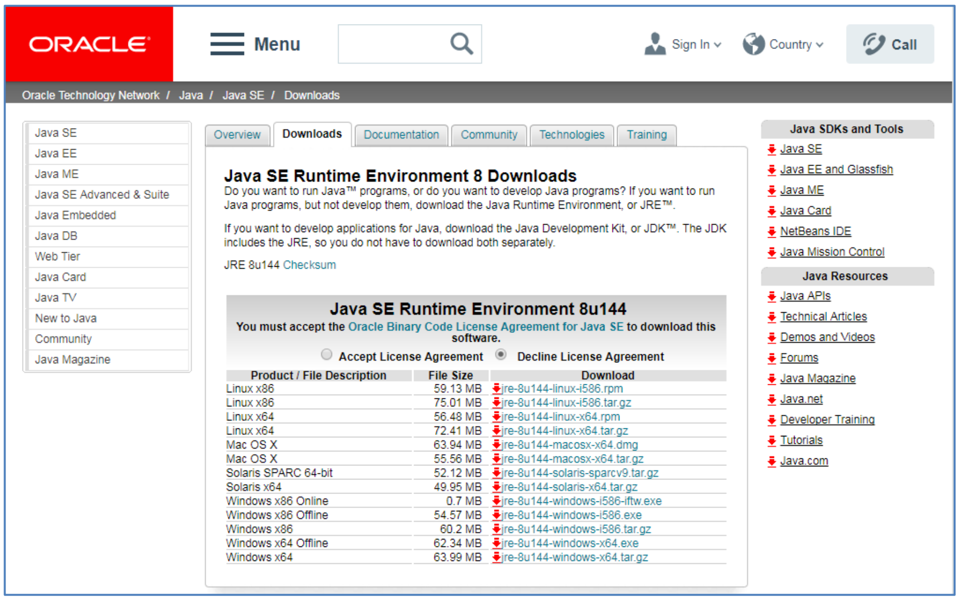
- Main Program: It subscribes DDS data, parses the dynamic parameters of machine status and translate to database data.
- MySQL database: It provides the simple way to make user access every DEX-100's data on four topics. Users only use the SQL instructions to filter data, get data, and etc. Easy to use to integrate with other IT system, e.g. MES, ALM and etc.

2. Installation

2.1 DEX-100 Vortex OpenSplice

If you want to use the DDS Module, it can start by the DEX-100 main program. Please refer to the DEX-100 main program manual. Before you start the DDS Module, you have to check the Vortex OpenSplice already existing.

1. Install the JRE x64 version



Java SE Runtime Environment 8 Downloads

Do you want to run Java™ programs, or do you want to develop Java programs? If you want to run Java programs, but not develop them, download the Java Runtime Environment, or JRE™.

If you want to develop applications for Java, download the Java Development Kit, or JDK™. The JDK includes the JRE, so you do not have to download both separately.

JRE 8u144 Checksum

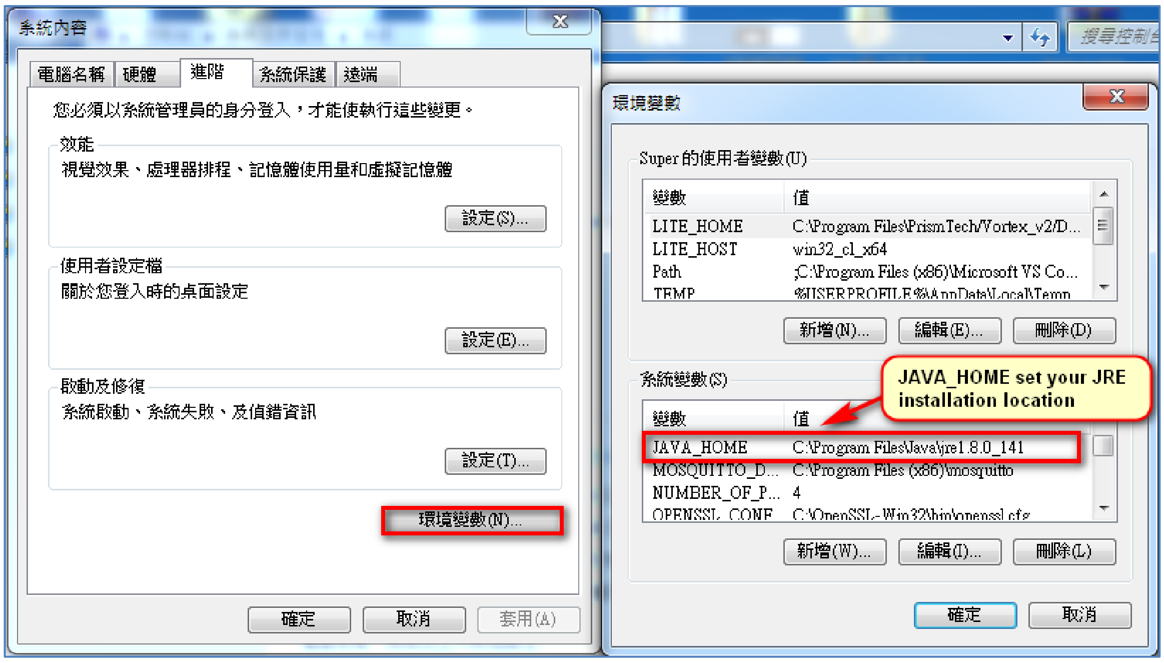
Java SE Runtime Environment 8u144

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux x86	59.13 MB	jre-8u144-linux-i586.rpm
Linux x86	75.01 MB	jre-8u144-linux-i586.tar.gz
Linux x64	56.48 MB	jre-8u144-linux-x64.rpm
Linux x64	72.41 MB	jre-8u144-linux-x64.tar.gz
Mac OS X	63.94 MB	jre-8u144-macosx-x64.dmg
Mac OS X	55.56 MB	jre-8u144-macosx-x64.tar.gz
Solaris SPARC 64-bit	52.12 MB	jre-8u144-solaris-sparcv9.tar.gz
Solaris x64	49.95 MB	jre-8u144-solaris-x64.tar.gz
Windows x86 Online	0.7 MB	jre-8u144-windows-i586-iftw.exe
Windows x86 Offline	54.57 MB	jre-8u144-windows-i586.exe
Windows x86	60.2 MB	jre-8u144-windows-i586.tar.gz
Windows x64 Offline	62.34 MB	jre-8u144-windows-x64.exe
Windows x64	63.99 MB	jre-8u144-windows-x64.tar.gz

2. Set the environment PATH


系統內容

電腦名稱 | 硬體 | 進階 | 系統保護 | 遠端

您必須以系統管理員的身分登入，才能使執行這些變更。

效能
視覺效果、處理器排程、記憶體使用量和虛擬記憶體

使用者設定檔
關於您登入時的桌面設定

啟動及修復
系統啟動、系統失敗，及偵錯資訊

環境變數

Super 的使用者變數(U)

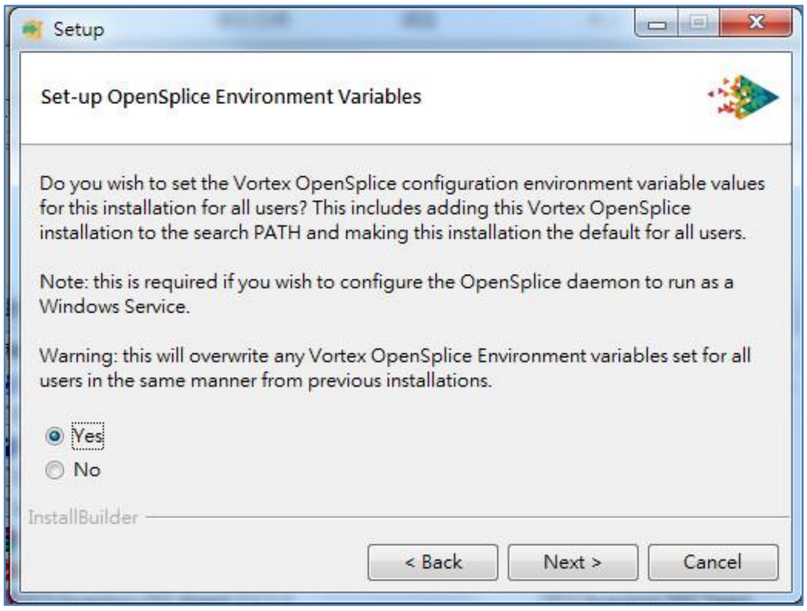
變數	值
LITE_HOME	C:\Program Files\PrismTech\Vortex_v2/D...
LITE_HOST	win32_cl_x64
Path	;C:\Program Files (x86)\Microsoft VS Co...
TEMP	%USERPROFILE%\AppData\Local\Temp

系統變數(S)

變數	值
JAVA_HOME	C:\Program Files\Java\jre1.8.0_141
MOSQUITTO_D...	C:\Program Files (x86)\mosquitto
NUMBER_OF_P...	4
OPENSSL_CONF	C:\OpenSSL-Win32\bin\openssl.cfg

3. Install the Vortex OpenSplice

Remember to choose “YES” in the installation process “Set-up OpenSplice Environment Variables”


Setup

Set-up OpenSplice Environment Variables

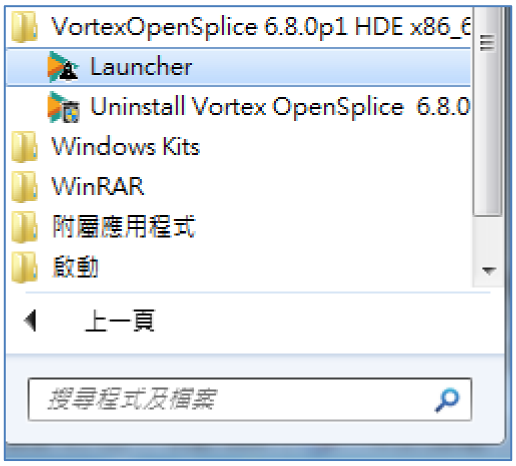
Do you wish to set the Vortex OpenSplice configuration environment variable values for this installation for all users? This includes adding this Vortex OpenSplice installation to the search PATH and making this installation the default for all users.

Note: this is required if you wish to configure the OpenSplice daemon to run as a Windows Service.

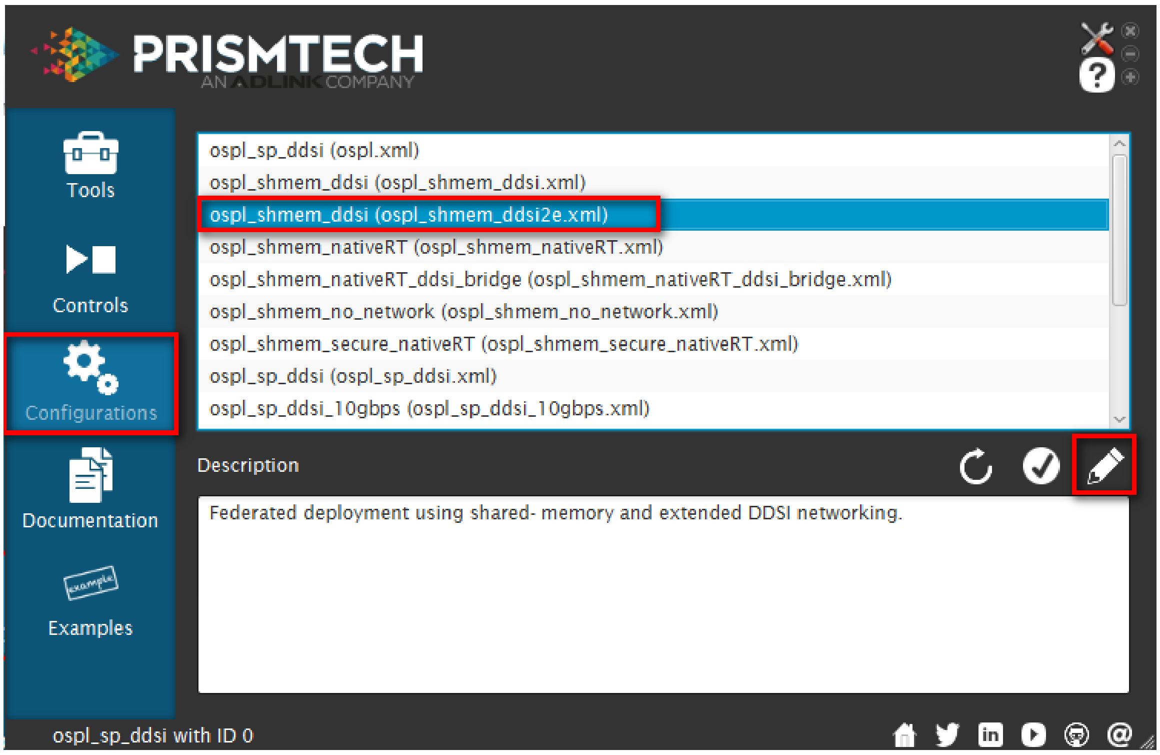
Warning: this will overwrite any Vortex OpenSplice Environment variables set for all users in the same manner from previous installations.

[✓ Yes] No

4. Open the OpenSplice Launcher


VortexOpenSplice 6.8.0p1 HDE x86_6
Launcher
Uninstall Vortex OpenSplice 6.8.0
Windows Kits
WinRAR
附屬應用程式
啟動
上一頁
搜尋程式及檔案

5. Click the “Configurations” and set up the ospl_shmem_ddsi (ospl_shmem_ddsi2e.xml)

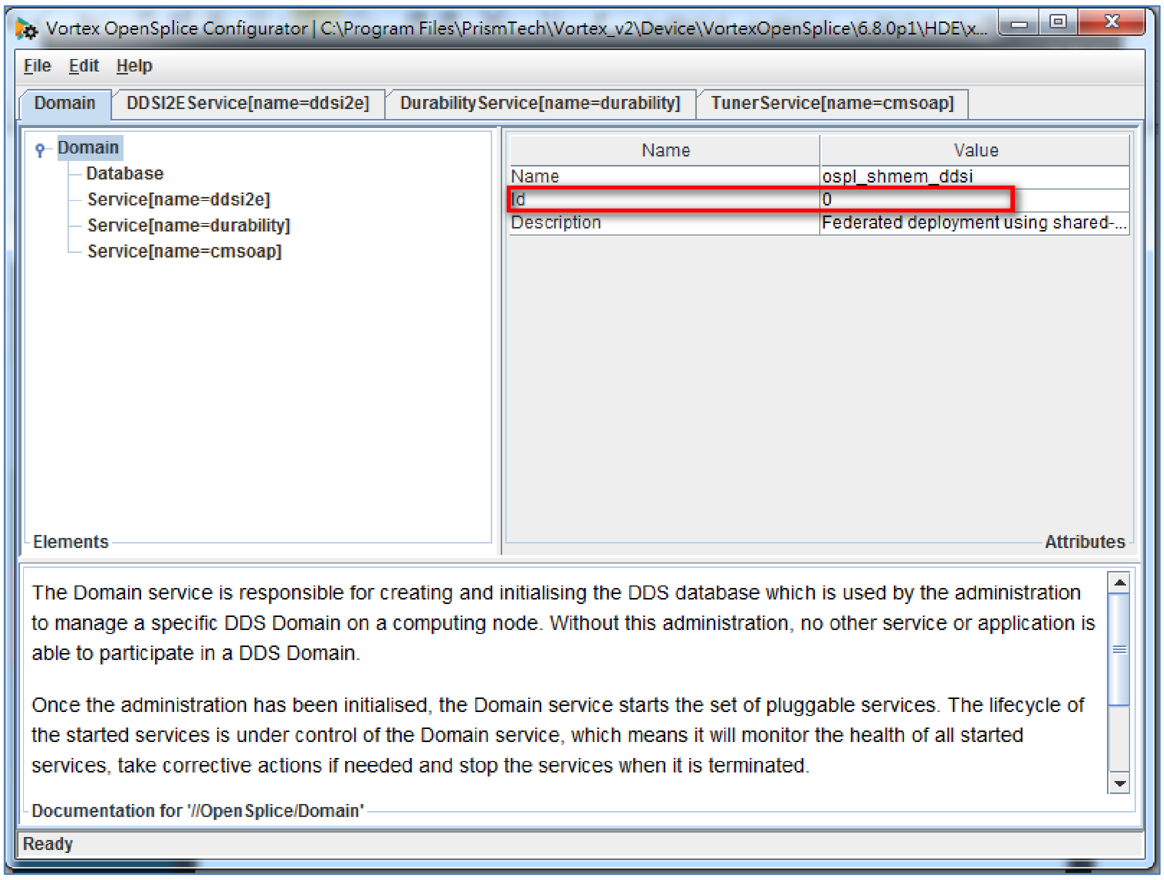


ospl_sp_ddsi (ospl.xml)
ospl_shmem_ddsi (ospl_shmem_ddsi.xml)
ospl_shmem_ddsi (ospl_shmem_ddsi2e.xml)
ospl_shmem_nativeRT (ospl_shmem_nativeRT.xml)
ospl_shmem_nativeRT_ddsi_bridge (ospl_shmem_nativeRT_ddsi_bridge.xml)
ospl_shmem_no_network (ospl_shmem_no_network.xml)
ospl_shmem_secure_nativeRT (ospl_shmem_secure_nativeRT.xml)
ospl_sp_ddsi (ospl_sp_ddsi.xml)
ospl_sp_ddsi_10gbps (ospl_sp_ddsi_10gbps.xml)

Federated deployment using shared- memory and extended DDSI networking.

ospl_sp_ddsi with ID 0

6. You need to set the domain id, if you want to make the dds program in the same scope (e.g. DDS module and Edge Server DDS program. If you let DDS and Edge Server DDS program, you must set the same domain id.)


Vortex OpenSplice Configurator | C:\Program Files\PrismTech\Vortex_v2\Device\VortexOpenSplice\6.8.0p1\HDE\x...
File Edit Help
Domain
DDSI2EService[name=ddsi2e]
DurabilityService[name=durability]
TunerService[name=cmsoap]
Domain
Database
Service[name=ddsi2e]
Service[name=durability]
Service[name=cmsoap]
Elements
Attributes
Ready

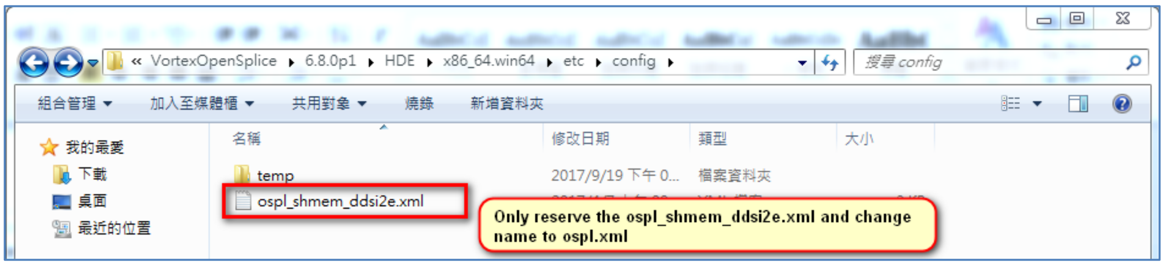
Name	Value
Name	ospl_shmem_ddsi
Id	0
Description	Federated deployment using shared-...

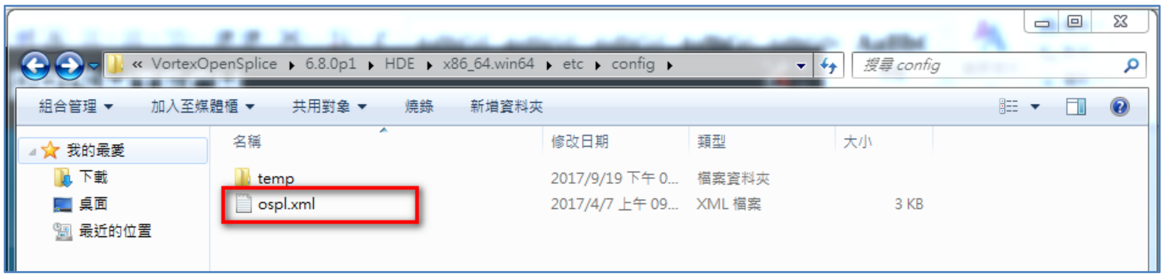
The Domain service is responsible for creating and initialising the DDS database which is used by the administration to manage a specific DDS Domain on a computing node. Without this administration, no other service or application is able to participate in a DDS Domain.

Once the administration has been initialised, the Domain service starts the set of pluggable services. The lifecycle of the started services is under control of the Domain service, which means it will monitor the health of all started services, take corrective actions if needed and stop the services when it is terminated.

Documentation for '//OpenSplice/Domain'

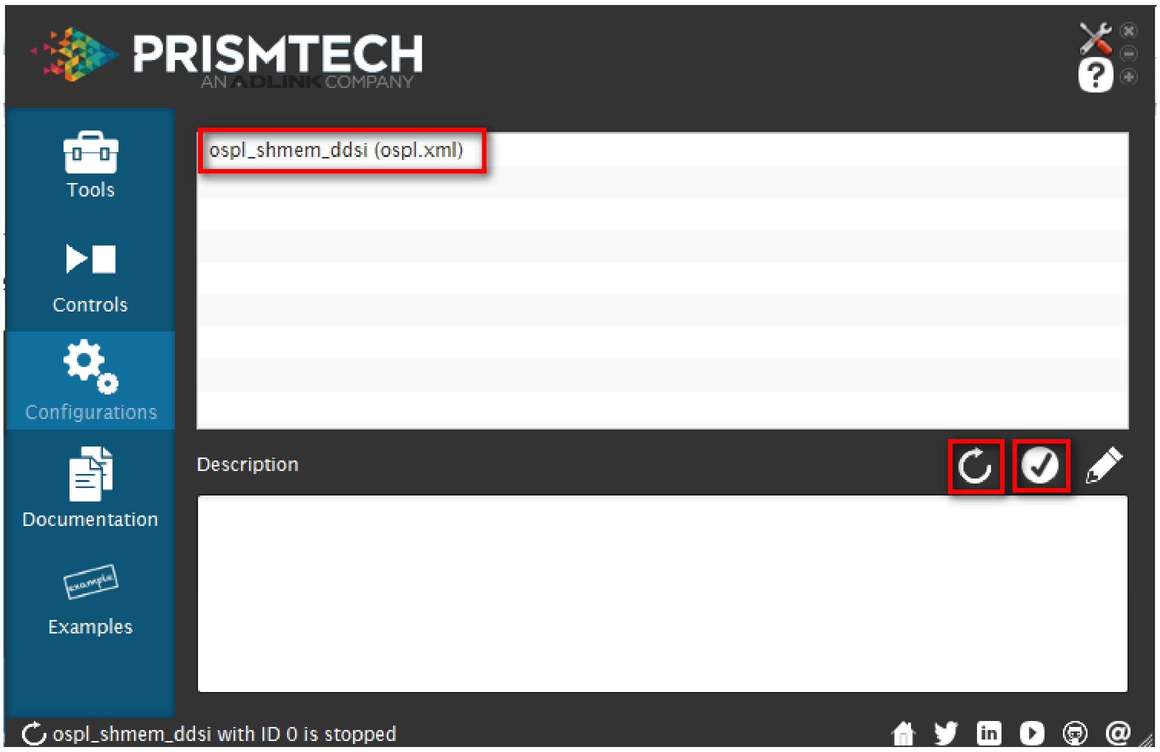
7. Copy others config files to other folder and change the ospl_shmem_ddsi2e.xml to ospl.xml


« VortexOpenSplice ▸ 6.8.0p1 ▸ HDE ▸ x86_64.win64 ▸ etc ▸ config ▸
搜尋 config
組合管理 ▾ 加入至媒體櫃 ▾ 共用對象 ▾ 燒錄 新增資料夾
我的最愛
下載
桌面
最近的位置
名稱 修改日期 類型 大小
temp 2017/9/19 下午 0... 檔案資料夾
ospl_shmem_ddsi2e.xml
Only reserve the ospl_shmem_ddsi2e.xml and change name to ospl.xml

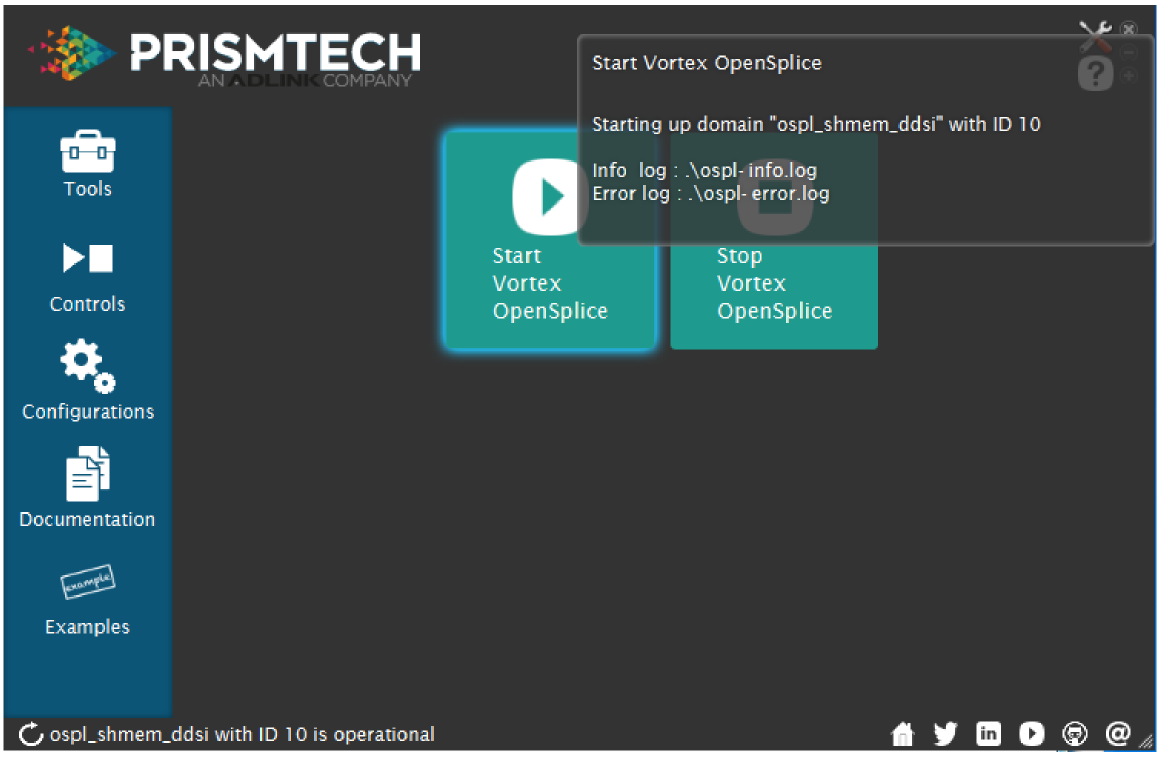

« VortexOpenSplice ▸ 6.8.0p1 ▸ HDE ▸ x86_64.win64 ▸ etc ▸ config ▸
搜尋 config
組合管理 ▾ 加入至媒體櫃 ▾ 共用對象 ▾ 燒錄 新增資料夾
我的最愛
下載
桌面
最近的位置

名稱	修改日期	類型	大小
temp	2017/9/19 下午 0...	檔案資料夾	
ospl.xml	2017/4/7 上午 09...	XML 檔案	3 KB

8. Refresh the configuration and set ospl.xml to default configuration



9. Click to “Controls” and start the Vortex OpenSplice



Start Vortex OpenSplice

Starting up domain "ospl_shmem_ddsi" with ID 10

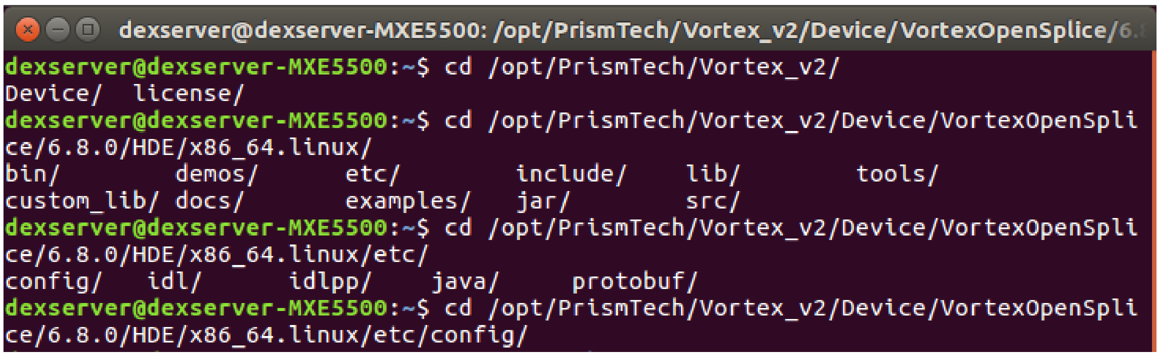
```
Info  log : .\ospl-info.log
Error log : .\ospl-error.log
```

ospl_shmem_ddsi with ID 10 is operational

2.2 Edge Server Vortex OpenSplice (The default image is already included)

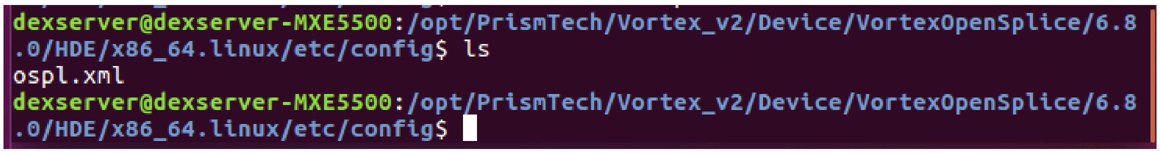
In the previous section, we mention that if we want to DEX-100 DDS module and Edge Server program working normally, we shall set the Edge Server Vortex OpenSplice in the same domain id and the same configuration. We will set the confiuration and start the Vortex OpenSplice step by step as follow.

1. Open the terminal and change dictionary to configuration location
   “cd /opt/Prismtech/Vortex_v2/Device/VortexOpenSplice/6.8.0/HDE/x86_64.linux/etc/config”



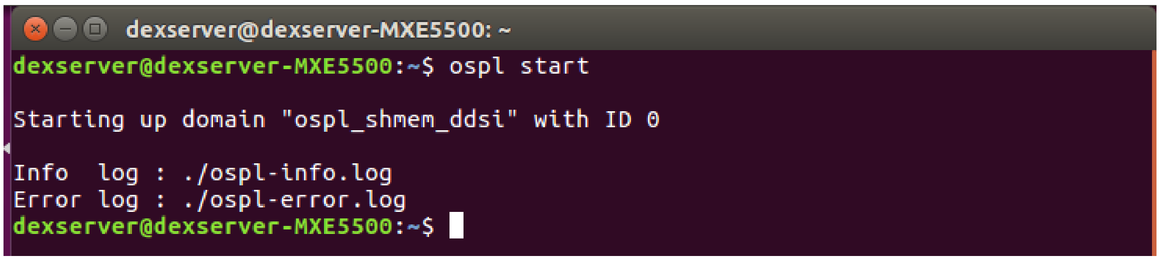
```
dexserver@dexserver-MXE5500: /opt/PrismTech/Vortex_v2/Device/VortexOpenSplice/6.
dexserver@dexserver-MXE5500:~$ cd /opt/PrismTech/Vortex_v2/
Device/  license/
dexserver@dexserver-MXE5500:~$ cd /opt/PrismTech/Vortex_v2/Device/VortexOpenSpli
ce/6.8.0/HDE/x86_64.linux/
bin/        demos/      etc/        include/    lib/        tools/
custom_lib/ docs/       examples/   jar/        src/
dexserver@dexserver-MXE5500:~$ cd /opt/PrismTech/Vortex_v2/Device/VortexOpenSpli
ce/6.8.0/HDE/x86_64.linux/etc/
config/   idl/      idlpp/    java/     protobuf/
dexserver@dexserver-MXE5500:~$ cd /opt/PrismTech/Vortex_v2/Device/VortexOpenSpli
ce/6.8.0/HDE/x86_64.linux/etc/config/
```

2. Edit the ospl.xml domain id
   “sudo gedit ospl.xml” and save



```
dexserver@dexserver-MXE5500:/opt/PrismTech/Vortex_v2/Device/VortexOpenSplice/6.8
.0/HDE/x86_64.linux/etc/config$ ls
ospl.xml
dexserver@dexserver-MXE5500:/opt/PrismTech/Vortex_v2/Device/VortexOpenSplice/6.8
.0/HDE/x86_64.linux/etc/config$ 
```

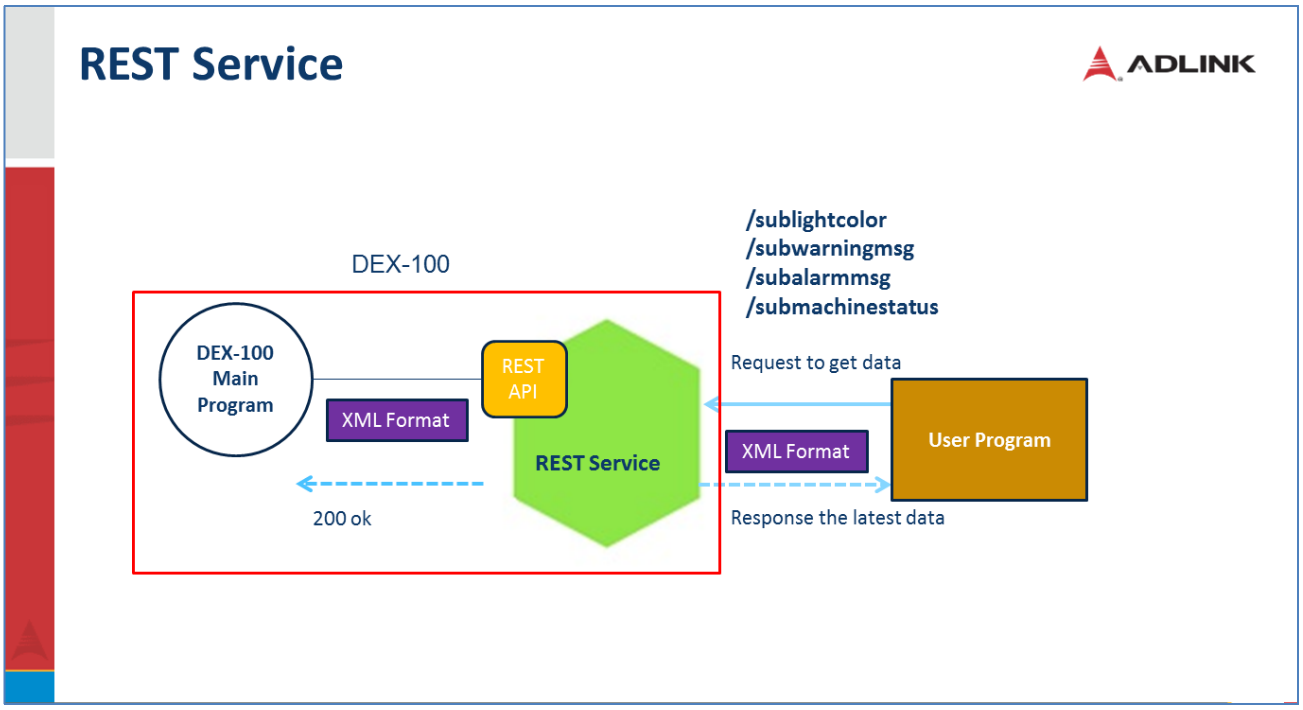
3. Start the Vortex OpenSplice
   “ospl start”



```
dexserver@dexserver-MXE5500: ~
dexserver@dexserver-MXE5500:~$ ospl start

Starting up domain "ospl_shmem_ddsi" with ID 0

Info  log : ./ospl-info.log
Error log : ./ospl-error.log
dexserver@dexserver-MXE5500:~$ 
```

3. REST Module

We defined the four API categories to support DEX-100 data extraction; they are “LightColor”, “WarningMsg”, “AlarmMsg”, “MachineStatus”, as one of the service in DES (Data Extraction Service). When you start the REST Module in the DEX-100 main program, you can use the REST API to get latest data. The details are as follow section:



3.1 REST API

When DEX-100 main program extracts data, it will send data to the REST Module and the data will keep the latest data in the REST module. In order to make user retrieve data easily, REST module provides many REST APIs as follows:

3.1.1 LightColor

3.1.1.1 LightColor XML format

[URI]
http://hostname:port/sublightcolor

[Content-Type]
text/xml

[Request]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<DES>
      <LightColor>
            <SubDataType>RAW_DATA</SubDataType>
      </LightColor>
</DES>
```

[Explanation]

<SubDataType>RAW_DATA</SubDataType>
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<DES>
   <LightColor>
            <machine_id>20</machine_id>
            <date>20170502</date>
            <time>12:00:00</time>
            <color>YELLOW</color>
            <response>Success</response>
   </LightColor>
</DES>
```

3.1.1.2 LightColor JSON format

[URI]
http://hostname:port/sublightcolor

[Content-Type]
application/json

[Request]

```
{
   "DES":  {
      "LightColor":  {  "SubDataType":  "RAW_DATA"  }
   }
}
```

[Explanation]

"SubDataType": "RAW_DATA"
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now

RAW_DATA only)

[Response]

```
{
   "DES": {
      "LightColor": {
         "machine_id": "20",
         "date": "20170502",
         "time": "12:00:00",
         "color": "YELLOW",
         "response": "Success"
      }
   }
}
```

3.1.2 WarningMsg

[Description]
The subwarningmsg is to get data by rest and it doesn’t clean the warning message data when call this API. In another way, we provide the takewarningmsg to get data and clean the data to NaN immediately when getting warning message successfully.

3.1.2.1 WarningMsg XML format

[URI]
http://hostname:port/subwarningmsg

[Content-Type]
text/xml

[Request]

```
<?xml version="1.0" encoding="UTF-8"?>
<DES>
      <WarningMsg>
            <SubDataType>RAW_DATA</SubDataType>
      </WarningMsg>
</DES>
```

[Explanation]

<SubDataType>RAW_DATA</SubDataType>

--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
<?xml  version="1.0"  encoding="UTF8"?>
<DES>
    <WarningMsg>
        <machine_id>30</machine_id>
        <date>20170502</date>
        <time>12:00:00</time>
        <msg_num>TD1000</msg_num>
        <msg>Not  Enough  Energy</msg>
        <response>Success</response>
    </WarningMsg>
</DES>
```

[URI]
http://hostname:port/takewarningmsg

[Content-Type]
text/xml

[Request]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<DES>
    <WarningMsg>
        <SubDataType>RAW_DATA</SubDataType>
    </WarningMsg>
</DES>
```

[Explanation]

<SubDataType>RAW_DATA</SubDataType>
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
<?xml  version="1.0"  encoding="UTF8"?>
<DES>
        <WarningMsg>
                <machine_id>30</machine_id>
                <date>20170502</date>
                <time>12:00:00</time>
                <msg_num>TD1000</msg_num>
                <msg>Not  Enough  Energy</msg>
                <response>Success</response>
        </WarningMsg>
</DES>
```

3.1.2.1 WarningMsg JSON format

[URI]
http://hostname:port/subwarningmsg

[Content-Type]
application/json

[Request]

```
{
   "DES":  {
      "WarningMsg":  {  "SubDataType":  "RAW_DATA"  }
   }
}
```

[Explanation]

"SubDataType": "RAW_DATA"
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
{
   "DES":  {
        "WarningMsg": {
            "machine_id": "30",
            "date": "20170502",
            "time": "12:00:00",
            "msg_num": "TD1000",
            "msg": "Not Enough Energy",
            "response": "Success"
        }
    }
}
```

[URI]
http://hostname:port/takewarningmsg

[Content-Type]
application/json

[Request]

```
{
    "DES": {
        "WarningMsg": { "SubDataType": "RAW_DATA" }
    }
}
```

[Explanation]

"SubDataType": "RAW_DATA"
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
{
    "DES": {
        "WarningMsg": {
            "machine_id": "30",
            "date":  "20170502",

            "time":  "12:00:00",

            "msg_num":  "TD1000",

            "msg":  "Not  Enough  Energy",

            "response":  "Success"

        }

    }

}
```

3.1.3 AlarmMsg

[Description]
The subalarmmsg is to get data by rest and it doesn’t clean the warning message data when call this API. In another way, we provide the takealarmmsg to get data and clean the data to NaN immediately when getting alarm message successfully. [URI]

3.1.3.1 AlarmMsg XML format

[URI]
http://hostname:port/subalarmmsg

[Content-Type]
text/xml

[Request]

```
<?xml  version="1.0"  encoding="UTF-8"?>

<DES>

     <AlarmMsg>

          <SubDataType>RAW_DATA</SubDataType>

     </AlarmMsg>

</DES>
```

[Explanation]

<SubDataType>RAW_DATA</SubDataType>
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
<?xml  version="1.0"  encoding="UTF8"?>
```

```
<DES>
    <AlarmMsg>
        <machine_id>15</machine_id>
        <date>20170502</date>
        <time>12:00:00</time>
        <major>X199</major>
        <minor>Y20</minor>
        <msg>Error  Arm</msg>
        <response>Success</response>
    </AlarmMsg>
</DES>
```

[URI]
http://hostname:port/takealarmmsg

[Content-Type]
text/xml

[Request]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<DES>
    <AlarmMsg>
        <SubDataType>RAW_DATA</SubDataType>
    </AlarmMsg>
</DES>
```

[Explanation]

<SubDataType>RAW_DATA</SubDataType>
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
<?xml  version="1.0"  encoding="UTF8"?>
<DES>
    <AlarmMsg>
        <machine_id>15</machine_id>
```

```
            <date>20170502</date>

            <time>12:00:00</time>

            <major>X199</major>

            <minor>Y20</minor>

            <msg>Error  Arm</msg>

            <response>Success</response>

        </AlarmMsg>

</DES>
```

3.1.3.2 AlarmMsg JSON format

[URI]
http://hostname:port/subalarmmsg

[Content-Type]
application/json

[Request]

```
{

   "DES": {

      "AlarmMsg": { "SubDataType": "RAW_DATA" }

   }

}
```

[Explanation]

"SubDataType": "RAW_DATA"
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
{

   "DES": {

      "AlarmMsg": {

         "machine_id": "15",

         "date": "20170502",

         "time": "12:00:00",

         "major": "X199",
        "minor": "Y20",
        "msg": "Error Arm",
        "response": "Success"
      }
   }
}
```

[URI]
http://hostname:port/takealarmmsg

[Content-Type]
application/json

[Request]

```
{
   "DES": {
      "AlarmMsg": { "SubDataType": "RAW_DATA" }
   }
}
```

[Explanation]

"SubDataType": "RAW_DATA"
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
{
   "DES": {
      "AlarmMsg": {
        "machine_id": "15",
        "date": "20170502",
        "time": "12:00:00",
        "major": "X199",
        "minor": "Y20",
        "msg": "Error Arm",
        "response": "Success"
        }
    }
}
```

3.1.4 MachineStatus

3.1.4.1 MachineStatus XML format

[URI]
http://hostname:port/submachinestatus

[Content-Type]
text/xml

[Request]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<DES>
        <MachineStatus>
                <SubDataType>RAW_DATA</SubDataType>
        </MachineStatus>
</DES>
```

[Explanation]

<SubDataType> </SubDataType>
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
<?xml  version="1.0"  encoding="UTF8"?>
<DES>
        <MachineStatus>
                <machine_id>11</machine_id>
                <date>20170502</date>
                <time>12:00:00</time>
                <descriptors>Complete_PWB,Attempted_Pickup,Missed_Pickup_Errors,Abnormal_Pickup_Errors,Recogition_Errors,Abnormal_Pickup_Errors,Recogition_Errors,Successful_Pickup_Rate,Machine_Troubles,Operation_Time,Mounting_Time,Stopped_Time</descriptors>
```

```
            <datatypes>u8,u8,u8,u16,u16,f64,u16,str,str,str</datatypes>

            <values>11,50,50,0,0,100.0,0,15H33M16S,15H33M16S,15H33M16S</values>

            <response>Success</response>

        </MachineStatus>

</DES>
```

3.1.4.2 MachineStatus JSON format

[URI]
http://hostname:port/submachinestatus

[Content-Type]
application/json

[Request]

```
{

   "DES": {

      "MachineStatus": { "SubDataType": "RAW_DATA" }

   }

}
```

[Explanation]

"SubDataType": "RAW_DATA"
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
{

   "DES": {

      "MachineStatus": {

         "machine_id": "11",

         "date": "20170502",

         "time": "12:00:00",

         "descriptors": "Complete_PWB,Attempted_Pickup,Missed_Pickup_Errors,Abnormal_Pickup_Errors,Recogition_Errors,Abnormal_Pickup_Errors,Recogition_Errors,Successful_Pickup_Rate,Machine_Troubles,Operation_Time,Mounting_Time,Stopped_Time",

         "datatypes": "u8,u8,u8,u16,u16,f64,u16,str,str,str",
        "values":  "11,50,50,0,0,100.0,0,15H33M16S,15H33M16S,15H33M16S",
        "response":  "Success"
      }
    }
}
```

3.1.5 SptStatus

3.1.5.1 SptStatus XML format

[URI]
http://hostname:port/subsptstatus

[Content-Type]
text/xml

[Request]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<DES>
    <SptStatus>
        <SubDataType>RAW_DATA</SubDataType>
    </SptStatus>
</DES>
```

[Explanation]

<SubDataType> </SubDataType>
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
<?xml  version="1.0"  encoding="UTF8"?>
<DES>
    <SptStatus>
        <date>20180612</date>
        <time>12:00:00</time>
        <spt_cmd_step>1</spt_cmd_step>
        <spt_cmd_name>Connect</spt_cmd_name>
```

```
            <response>Success</response>
        </SptStatus>
</DES>
```

3.1.5.2 SptStatus JSON format

[URI]
http://hostname:port/subsptstatus

[Content-Type]
application/json

[Request]

```
{
    "DES":  {
        "SptStatus":  {  "SubDataType":  "RAW_DATA"  }
    }
}
```

[Explanation]

"SubDataType": "RAW_DATA"
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
{
    "DES":  {
        "SptStatus":  {
            "date":  "20180612",
            "time":  "12:00:00",
            "spt_cmd_step":  "1",
            "spt_cmd_name":  "Connect",
            "response":  "Success"
        }
    }
}
```

3.1.6 SysMode

3.1.6.1 SysMode XML format

[URI]

http://hostname:port/subsysmode

[Content-Type]

text/xml

[Request]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<DES>
    <SysMode>
        <SubDataType>RAW_DATA</SubDataType>
    </SysMode>
</DES>
```

[Explanation]

<SubDataType> </SubDataType>
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
<?xml  version="1.0"  encoding="UTF8"?>
<DES>
    <SysMode>
        <date>20180612</date>
        <time>12:00:00</time>
        <mode>1</mode>
        <response>Success</response>
    </SysMode>
</DES>
```

[Explanation]

<Mode> 1 </Mode>
1: Data Extraction Mode
2: Full Control Mode

3.1.6.2 SysMode JSON format

[URI]
http://hostname:port/subsysmode

[Content-Type]
application/json

[Request]

```
{
   "DES": {
      "SysMode": { "SubDataType": "RAW_DATA" }
   }
}
```

[Explanation]

"SubDataType": "RAW_DATA"
--- the subscribe data type of DEX-series, it may be RAW_DATA, ANA_DATA and etc.(Now RAW_DATA only)

[Response]

```
{
   "DES": {
      "SysMode": {
         "date": "20180612",
         "time": "12:00:00",
         "mode": "1",
         "response": "Success"
      }
   }
}
```

[Explanation]

"SubDataType": "1"
1: Data Extraction Mode
2: Full Control Mode

3.1.7 message

3.1.7.1 message JSON format

[URI]
http://hostname:port/message

[Method]
POST

[Content-Type]
application/json

[Request]

```
{
   "message":  "test",
   "messageTitle":  "test",
   "enterFullOperationWhenClosed":  1
}
```

[Response]

```
{
   "DES":  {
      "response":"Send  [Message]  topic  ok"
   }
}
```

[URI]
http://hostname:port/message

[Method]
GET

[Content-Type]
application/json

[Request]
No body

[Response]

```
{
        {
            "message":  "test",
            "messageTitle":  "test",
            "enterFullOperationWhenClosed":  1
        },
        {
            "message":  "test",
            "messageTitle":  "test",
            "enterFullOperationWhenClosed":  2
        },
}
```

[URI]
http://hostname:port/message

[Method]
DELETE

[Content-Type]
application/json

[Request]
No body

[Response]

```
{
   "DES":  {
       "response":"Delete  [Message]  topic  ok"
   }
}
```

3.1.8 Script

3.1.8.1 executescript JSON format

[URI]
http://hostname:port/exectuescript/#

[Method]
POST

[Content-Type]
application/json

[Request]
No body

[Response]

```
{
  "ErrorCode":  0
}
```

3.1.8.2 stopscript JSON format

[URI]
http://hostname:port/stopscript

[Method]
POST

[Content-Type]
application/json

[Request]
No body

[Response]

```
{
  "ErrorCode":  0
}
```

3.1.9 OCR table value

3.1.9.1 setocrdata JSON format

[URI]
http://hostname:port/setocrdata

[Method]
POST

[Content-Type]
application/json

[Request]
```
{
   "OCRID":  "2",
   "OCRValue":  "abcd"
}
```

[Response]
```
{
   "ErrorCode":  0
}
```

[Note]
1. The index 0 is not accessible.
2. The response time is about 200ms once

3.1.9.2 getocrdata JSON format
[URI]
http://hostname:port/getocrdata/#

[Method]
GET

[Content-Type]
application/json

[Request]
No body

[Response]
```
{
   "ErrorCode":  0,
   "ECItem":  [
          {
            "ECID":  2,
              "ECValue":  "abcd"
```

```
        }
    ]
}
```

[Note]
The index 0 is not accessible.

3.2 Writing Web Services Client Applications

This section briefly describes how to write a web service client to get data. It is divided by programming language.

3.2.1 C#

[POST Procedure]

You have to pass two parameters in postXMLData,
The first parameter is URL, it likes "http://127.0.0.1:8888/sublightcolor"
and second is xml data, it likes

```
"<?xml version="1.0" encoding="UTF-8"?>
<DES>
        <LightColor>
                <SubDataType>RAW_DATA</SubDataType>
        </LightColor>
</DES>"
```

```
public string postXMLData(string destinationUrl, string requestXml)
{
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(destinationUrl);
        byte[] bytes;
        bytes = System.Text.Encoding.ASCII.GetBytes(requestXml);
        request.ContentType = "text/xml";
        request.ContentLength = bytes.Length;
        request.Method = "POST";
        Stream requestStream = request.GetRequestStream();
        requestStream.Write(bytes, 0, bytes.Length);
        requestStream.Close();
        HttpWebResponse response;
        response = (HttpWebResponse)request.GetResponse();
        if (response.StatusCode == HttpStatusCode.OK)
        {
        Stream responseStream = response.GetResponseStream();
        string responseStr = new StreamReader(responseStream).ReadToEnd();
        return responseStr;
        }
        return null;
}
```

3.2.2 JAVA

[POST Procedure]

Apache HttpClient Package

You have to pass two parameters in postXMLData,

The first parameter is URL, it likes "http://127.0.0.1:8888/sublightcolor"

and second is xml data, it likes

```
"<?xml version="1.0" encoding="UTF-8"?>
<DES>
        <LightColor>
                <SubDataType>RAW_DATA</SubDataType>
        </LightColor>
</DES>"
```

```
public string postXMLData(string destinationUrl, string requestXml)
{
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpPost postRequest = new HttpPost(destinationUrl);
StringEntity input = new StringEntity(requestXml);
input.setContentType("text/xml");
postRequest.setEntity(input);
HttpResponse response = httpClient.execute(postRequest);
int code = response.getStatusLine().getStatusCode();
String body = EntityUtils.toString(response.getEntity());
return body;
} //for 200 ok only
```

3.2.3 Python

[POST Procedure]

urllib package

You have to pass two parameters in postXMLData,

The first parameter is URL, it likes "http://127.0.0.1:8888/sublightcolor"

and second is xml data, it likes

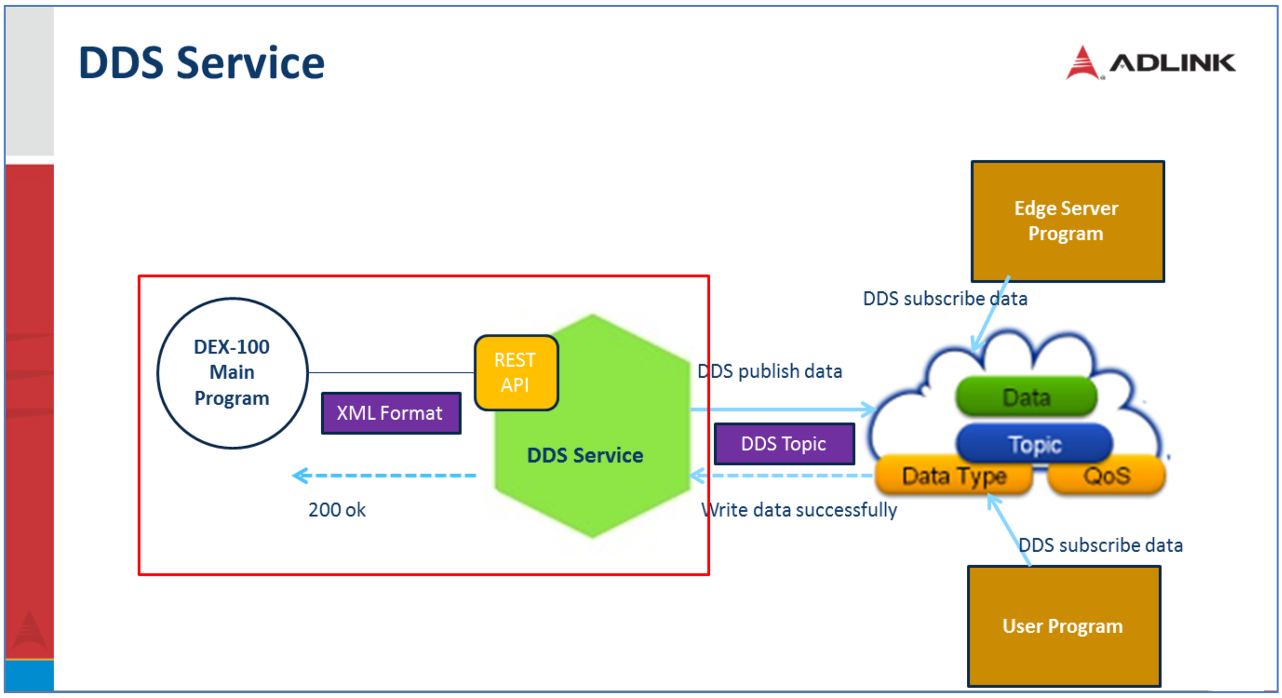
```
"<?xml version="1.0" encoding="UTF-8"?>
<DES>
        <LightColor>
                <SubDataType>RAW_DATA</SubDataType>
        </LightColor>
</DES>"
```

```
method = "POST"
handler = urllib2.HTTPHandler()
```

```
opener = urllib2.build_opener(handler)
#xml setting
data = urllib.urlencode(dictionary_of_POST_fields_or_None)
#url setting
request = urllib2.Request(url, data=data)
request.add_header("Content-Type",'text/xml')
request.get_method = lambda: method
try:
        connection = opener.open(request)
except urllib2.HTTPError,e:
        connection = e
if connection.code == 200:
#get response xml data
        data = connection.read()
else:
        # handle the error case. connection.read() will still contain data
```

4. DDS Module

The DDS Module also defined four DDS topic in the idl. They are "LightColor", "WarningMsg", "AlarmMsg", "MachineStatus", as one of the service in DES (Data Extraction Service). When you start the DDS Module in the DEX-100 main program, you can write a DDS subscriber to retrieve data. The details are as follow section:



4.1 DDS IDL

If you want to retrieve the DEX-100 extraction data, you have to need the DES.idl on your DDS subscriber program. The DES.idl is as follows:

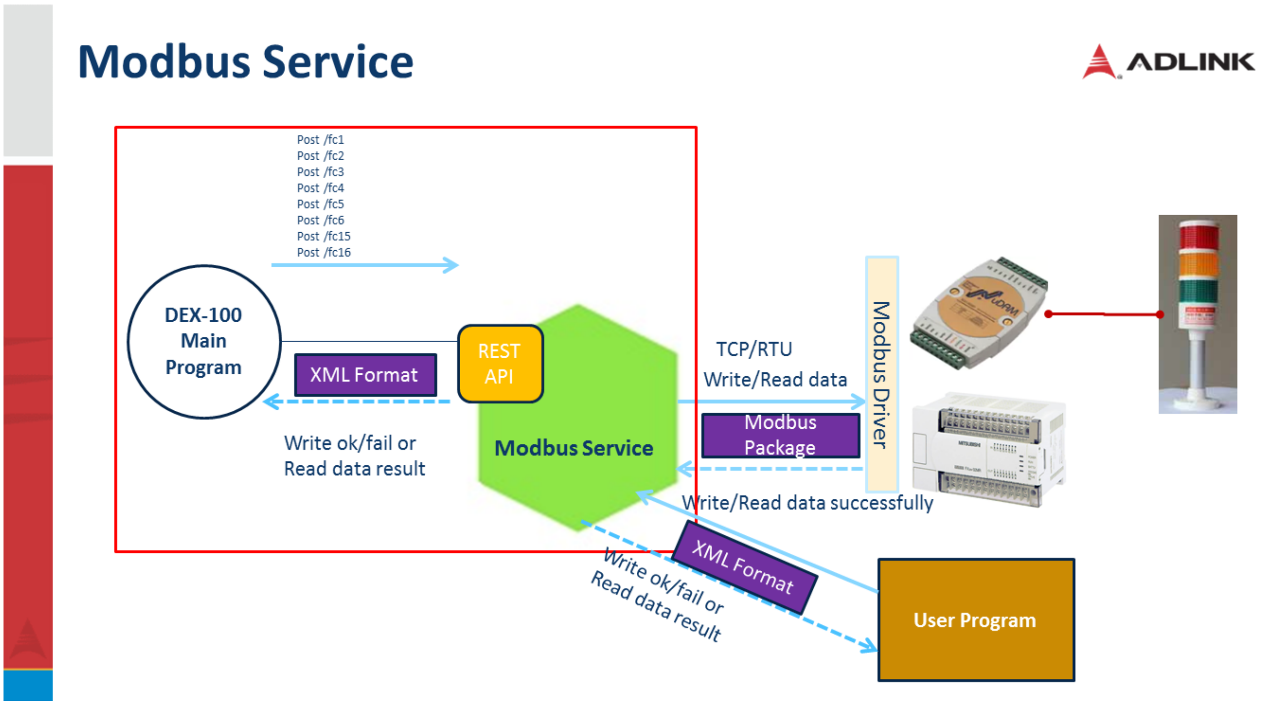
```
module DES

{

        struct TableSchema

        {

                unsigned short machine_id;

                unsigned short para_length;

                string value;

        };

        #pragma keylist TableSchema


        struct LightColor
```

```
   {
            unsigned short machine_id;
            string date;
            string time;
            string color;
   };
   #pragma keylist LightColor machine_id

struct WarningMsg
{
      unsigned short machine_id;
            string date;
            string time;
            string msg_num;
            string msg;
};
#pragma keylist WarningMsg machine_id

struct AlarmMsg
{
      unsigned short machine_id;
            string date;
            string time;
            string major;
            string minor;
            string msg;
};
#pragma keylist AlarmMsg machine_id

   struct MachineStatus
```

```
        {
                unsigned short machine_id;
                string date;
                string time;
                string descriptors;
                string datatypes;
                string values;
        };
        #pragma keylist MachineStatus machine_id
};
```

5. Modbus Module

Modbus Module support most main Modbus functions to access Modbus devices. It can only use the DEX-100 main program to link Modbus module to access data or write a simple web service client program to retrieve Modbus device data which connected on DEX-100 (RTU) or in the same scope network (TCP).



5.1 Modbus Function

This section list the support Modbus function and the exchange XML format

5.1.1 Read Coil Status

[URI]
http://hostname:port/fc1

[Request]
TCP connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
        <connectionType>TCP</connectionType>
        <ip>127.0.0.1</ip>
        <port>502</port>
        <slave>1</slave>
        <dataType>UINT8</dataType>
        <startAddress>1</startAddress>
```

```
            <length>5</length>
</ModbusService>
```

RTU connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
            <connectionType>RTU</connectionType>
            <device>\\.\COM12</device>
            <baud>115200</baud>
            <parity>N</parity>
            <dataBit>8</dataBit>
            <stopBit>1</stopBit>
            <slave>1</slave>
            <dataType>UINT8</dataType>
            <startAddress>1</startAddress>
            <length>1</length>
</ModbusService>
```

[Explanation]

<connectionType>TCP</connectionType>
--- Modbus connection type, it may be TCP and RTU
<ip>127.0.0.1</ip>
--- TCP device IP
<port>502</port>
--- TCP device Port
<slave>1</slave>
--- Modbus Slave ID
<dataType>UINT8</dataType>
--- Data type, fc1 and fc2 only has UINT8
<startAddress>1</startAddress>
--- Start Address on Modbus device
<length>5</length>
--- Length of retrieved data on Modbus device

<device>\\.\COM12</device>
--- RTU device COM port
<baud>115200</baud>
--- RTU device baud rate
<parity>N</parity>
--- RTU device parity (N, E, O)
<dataBit>8</dataBit>

--- RTU device data bit (5,6,7,8)
<stopBit>1</stopBit>
--- RTU device stop bit (1,2)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<readBits>
        <startAddress>1</startAddress>
        <length>5</length>
        <values>0,1,0,1,0</values>
</readBits>
```

5.1.2 Read Input Status

[URI]
http://hostname:port/fc2

[Request]
TCP connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
        <connectionType>TCP</connectionType>
        <ip>127.0.0.1</ip>
        <port>502</port>
        <slave>1</slave>
        <dataType>UINT8</dataType>
        <startAddress>10000</startAddress>
        <length>5</length>
</ModbusService>
```

RTU connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
        <connectionType>RTU</connectionType>
        <device>\\.\COM12</device>
        <baud>115200</baud>
        <parity>N</parity>
```

```
            <dataBit>8</dataBit>

            <stopBit>1</stopBit>

            <slave>1</slave>

            <dataType>UINT8</dataType>

            <startAddress>10000</startAddress>

            <length>1</length>

</ModbusService>
```

[Explanation]

<connectionType>TCP</connectionType>
--- Modbus connection type, it may be TCP and RTU
<ip>127.0.0.1</ip>
--- TCP device IP
<port>502</port>
--- TCP device Port
<slave>1</slave>
--- Modbus Slave ID
<dataType>UINT8</dataType>
--- Data type, fc1 and fc2 only has UINT8
<startAddress>10000</startAddress>
--- Start Address on Modbus device
<length>5</length>
--- Length of retrieved data on Modbus device

<device>\\.\COM12</device>
--- RTU device COM port
<baud>115200</baud>
--- RTU device baud rate
<parity>N</parity>
--- RTU device parity (N, E, O)
<dataBit>8</dataBit>
--- RTU device data bit (5,6,7,8)
<stopBit>1</stopBit>
--- RTU device stop bit (1,2)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>

<readInputBits>

            <startAddress>10000</startAddress>

            <length>5</length>

            <values>1,0,1,0,1</values>
```

```
</readInputBits>
```

5.1.3 Read Holding Registers

[URI]

http://hostname:port/fc3

[Request]

TCP connection

```
<?xml version="1.0" encoding="UTF-8"?>
<ModbusService>
        <connectionType>TCP</connectionType>
        <ip>127.0.0.1</ip>
        <port>502</port>
        <slave>1</slave>
        <dataType>INT32</dataType>
        <dataSwapType>SWAP_BYTE</dataSwapType>
        <startAddress>0</startAddress>
        <length>7</length>
</ModbusService>
```

RTU connection

```
<?xml version="1.0" encoding="UTF-8"?>
<ModbusService>
        <connectionType>RTU</connectionType>
        <device>\\.\COM12</device>
        <baud>115200</baud>
        <parity>N</parity>
        <dataBit>8</dataBit>
        <stopBit>1</stopBit>
        <slave>1</slave>
        <dataType>INT32</dataType>
        <dataSwapType>SWAP_BYTE</dataSwapType>
        <startAddress>0</startAddress>
        <length>7</length>
</ModbusService>
```

[Explanation]

<connectionType>TCP</connectionType>
--- Modbus connection type, it may be TCP and RTU
<ip>127.0.0.1</ip>
--- TCP device IP
<port>502</port>
--- TCP device Port
<slave>1</slave>
--- Modbus Slave ID
<dataType>INT32</dataType>
--- Data type, fc3 and fc4 only has UINT16, UINT32, INT16, INT32 and FLOAT32
<dataSwapType>SWAP_BYTE</dataSwapType>
--- Data swap type, fc3 and fc4
For the data type UINT16 and INT16 are ORIGINAL and SWAP_BYTE
For the data type UINT32, INT32 and FLOAT32 are ORIGINAL, SWAP_BYTE, SWAP_WORD and SWAP_BYTE_WORD
<startAddress>0</startAddress>
--- Start Address on Modbus device
<length>7</length>
--- Length of retrieved data on Modbus device

<device>\\.\COM12</device>
--- RTU device COM port
<baud>115200</baud>
--- RTU device baud rate
<parity>N</parity>
--- RTU device parity (N, E, O)
<dataBit>8</dataBit>
--- RTU device data bit (5,6,7,8)
<stopBit>1</stopBit>
--- RTU device stop bit (1,2)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<readRegisters>
        <startAddress>0</startAddress>
        <length>7</length>
        <values>2560,60671,7680,-30,50,10,1376286</values>
        <rawdata>00000A00,0000ECFF,00001E00,FFFFFFE2,00000032,0000000A,0015001E</rawdata>
</readRegisters>
```

5.1.4 Read Input Registers

[URI]

http://hostname:port/fc4

[Request]
  TCP connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
        <connectionType>TCP</connectionType>
        <ip>127.0.0.1</ip>
        <port>502</port>
        <slave>1</slave>
        <dataType>INT32</dataType>
        <dataSwapType>SWAP_BYTE</dataSwapType>
        <startAddress>10000</startAddress>
        <length>7</length>
</ModbusService>
```

  RTU connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
        <connectionType>RTU</connectionType>
        <device>\\.\COM12</device>
        <baud>115200</baud>
        <parity>N</parity>
        <dataBit>8</dataBit>
        <stopBit>1</stopBit>
        <slave>1</slave>
        <dataType>INT32</dataType>
        <dataSwapType>SWAP_BYTE</dataSwapType>
        <startAddress>10000</startAddress>
        <length>7</length>
</ModbusService>
```

[Explanation]

<connectionType>TCP</connectionType>
--- Modbus connection type, it may be TCP and RTU

<ip>127.0.0.1</ip>
--- TCP device IP
<port>502</port>
--- TCP device Port
<slave>1</slave>
--- Modbus Slave ID
<dataType>INT32</dataType>
--- Data type, fc3 and fc4 only has UINT16, UINT32, INT16, INT32 and FLOAT32
<dataSwapType>SWAP_BYTE</dataSwapType>
--- Data swap type, fc3 and fc4
For the data type UINT16 and INT16 are ORIGINAL and SWAP_BYTE
For the data type UINT32, INT32 and FLOAT32 are ORIGINAL, SWAP_BYTE, SWAP_WORD and SWAP_BYTE_WORD
<startAddress>10000</startAddress>
--- Start Address on Modbus device
<length>7</length>
--- Length of retrieved data on Modbus device

<device>\\.\COM12</device>
--- RTU device COM port
<baud>115200</baud>
--- RTU device baud rate
<parity>N</parity>
--- RTU device parity (N, E, O)
<dataBit>8</dataBit>
--- RTU device data bit (5,6,7,8)
<stopBit>1</stopBit>
--- RTU device stop bit (1,2)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<readInputRegisters>
        <startAddress>10000</startAddress>
        <length>7</length>
        <values>2560,60671,7680,-30,50,10,1376286</values>
        <rawdata>00000A00,0000ECFF,00001E00,FFFFFFE2,00000032,0000000A,0015001E</rawdata>
</readInputRegisters>
```

5.1.5 Force Single Coil

[URI]
http://hostname:port/fc5

[Request]
TCP connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
        <connectionType>TCP</connectionType>
        <ip>127.0.0.1</ip>
        <port>502</port>
        <slave>1</slave>
        <dataType>UINT8</dataType>
        <startAddress>0</startAddress>
        <value>1</value>
</ModbusService>
```

RTU connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
        <connectionType>RTU</connectionType>
        <device>\\.\COM12</device>
        <baud>115200</baud>
        <parity>N</parity>
        <dataBit>8</dataBit>
        <stopBit>1</stopBit>
        <slave>1</slave>
        <dataType>UINT8</dataType>
        <startAddress>0</startAddress>
        <value>1</value>
</ModbusService>
```

[Explanation]

<connectionType>TCP</connectionType>
--- Modbus connection type, it may be TCP and RTU
<ip>127.0.0.1</ip>
--- TCP device IP
<port>502</port>
--- TCP device Port
<slave>1</slave>
--- Modbus Slave ID
<dataType>INT32</dataType>
--- Data type, fc5 only has UINT8
<startAddress>0</startAddress>

--- Start Address on Modbus device
<value>1</value>
--- Write bit value to Modbus device, it may be 1, true , True, TRUE, 0, false, False and FALSE

<device>\\.\COM12</device>
--- RTU device COM port
<baud>115200</baud>
--- RTU device baud rate
<parity>N</parity>
--- RTU device parity (N, E, O)
<dataBit>8</dataBit>
--- RTU device data bit (5,6,7,8)
<stopBit>1</stopBit>
--- RTU device stop bit (1,2)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>

<writeBit>

            <startAddress>0</startAddress>

            <values>1</values>

            <response>Success</response>

</writeBit>
```

5.1.6 Preset Single Register

[URI]
http://hostname:port/fc6

[Request]
 TCP connection

```
<?xml  version="1.0"  encoding="UTF-8"?>

<ModbusService>

            <connectionType>TCP</connectionType>

            <ip>127.0.0.1</ip>

            <port>502</port>

            <slave>1</slave>

            <dataType>INT16</dataType>

            <dataSwapType>ORIGINAL</dataSwapType>

            <startAddress>0</startAddress>

            <value>-20</value>
```

```
</ModbusService>
```

RTU connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
                <connectionType>RTU</connectionType>
                <device>\\.\COM12</device>
                <baud>115200</baud>
                <parity>N</parity>
                <dataBit>8</dataBit>
                <stopBit>1</stopBit>
                <slave>1</slave>
                <dataType>INT16</dataType>
                <dataSwapType>ORIGINAL</dataSwapType>
                <startAddress>0</startAddress>
                <value>-20</value>
</ModbusService>
```

[Explanation]

<connectionType>TCP</connectionType>
--- Modbus connection type, it may be TCP and RTU
<ip>127.0.0.1</ip>
--- TCP device IP
<port>502</port>
--- TCP device Port
<slave>1</slave>
--- Modbus Slave ID
<dataType>INT16</dataType>
--- Data type, fc6 only has UINT16 and INT16
<dataSwapType>ORIGINAL</dataSwapType>
--- Data swap type, fc6 only has ORIGINAL and SWAP_BYTE
<startAddress>0</startAddress>
--- Start Address on Modbus device
<value>-20</value>
--- Write register value to Modbus device

<device>\\.\COM12</device>
--- RTU device COM port
<baud>115200</baud>
--- RTU device baud rate
<parity>N</parity>
--- RTU device parity (N, E, O)

<dataBit>8</dataBit>
--- RTU device data bit (5,6,7,8)
<stopBit>1</stopBit>
--- RTU device stop bit (1,2)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>
<writeRegister>
                <startAddress>0</startAddress>
                <values>-20</values>
                <response>Success</response>
</writeRegister>
```

5.1.7 Force Multiple Coils

[URI]
http://hostname:port/fc15

[Request]
TCP connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
                <connectionType>TCP</connectionType>
                <ip>127.0.0.1</ip>
                <port>502</port>
                <slave>4</slave>
                <dataType>UINT8</dataType>
                <startAddress>0</startAddress>
                <length>4</length>
                <values>0,1,0,1</values>
</ModbusService>
```

RTU connection

```
<?xml  version="1.0"  encoding="UTF-8"?>
<ModbusService>
                <connectionType>RTU</connectionType>
                <device>\\.\COM12</device>
```

```
            <baud>115200</baud>
            <parity>N</parity>
            <dataBit>8</dataBit>
            <stopBit>1</stopBit>
            <slave>4</slave>
            <dataType>UINT8</dataType>
            <startAddress>0</startAddress>
            <length>4</length>
            <values>0,1,0,1</values>
</ModbusService>
```

[Explanation]

<connectionType>TCP</connectionType>
--- Modbus connection type, it may be TCP and RTU
<ip>127.0.0.1</ip>
--- TCP device IP
<port>502</port>
--- TCP device Port
<slave>4</slave>
--- Modbus Slave ID
<dataType>UINT8</dataType>
--- Data type, fc15 only has UINT8
<startAddress>0</startAddress>
--- Start Address on Modbus device
<length>4</length>
--- no. of bit value
<value>0,1,0,1</value>
--- Write bits value to Modbus device, it may be 1, true , True, TRUE, 0, false, False and FALSE

<device>\\.\COM12</device>
--- RTU device COM port
<baud>115200</baud>
--- RTU device baud rate
<parity>N</parity>
--- RTU device parity (N, E, O)
<dataBit>8</dataBit>
--- RTU device data bit (5,6,7,8)
<stopBit>1</stopBit>
--- RTU device stop bit (1,2)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>
```

```
<writeBits>
            <startAddress>0</startAddress>
            <length>4</length>
            <values>0,1,0,1</values>
            <response>Success</response>
</writeBits>
```

5.1.8 Force Multiple Coils

[URI]
http://hostname:port/fc16

[Request]
TCP connection

```
<?xml version="1.0" encoding="UTF-8"?>
<ModbusService>
            <connectionType>TCP</connectionType>
            <ip>127.0.0.1</ip>
            <port>502</port>
            <slave>1</slave>
            <dataType>FLOAT32</dataType>
            <dataSwapType>SWAP_BYTE_WORD</dataSwapType>
            <startAddress>0</startAddress>
            <length>3</length>
            <values>10,20,10.1</values>
</ModbusService>
```

RTU connection

```
<?xml version="1.0" encoding="UTF-8"?>
<ModbusService>
            <connectionType>RTU</connectionType>
            <device>\\.\COM12</device>
            <baud>115200</baud>
            <parity>N</parity>
            <dataBit>8</dataBit>
            <stopBit>1</stopBit>
            <slave>1</slave>

            <dataType>FLOAT32</dataType>

            <dataSwapType>SWAP_BYTE_WORD</dataSwapType>

            <startAddress>0</startAddress>

            <length>3</length>

            <values>10,20,10.1</values>

</ModbusService>
```

[Explanation]

<connectionType>TCP</connectionType>
--- Modbus connection type, it may be TCP and RTU
<ip>127.0.0.1</ip>
--- TCP device IP
<port>502</port>
--- TCP device Port
<slave>4</slave>
--- Modbus Slave ID
<dataType>FLOAT32</dataType>
--- Data type, fc16 only has UINT16, UINT32, INT16, INT32 and FLOAT32
<dataSwapType>SWAP_BYTE</dataSwapType>
--- Data swap type, fc16
For the data type UINT16 and INT16 are ORIGINAL and SWAP_BYTE
For the data type UINT32, INT32 and FLOAT32 are ORIGINAL, SWAP_BYTE, SWAP_WORD and SWAP_BYTE_WORD
<startAddress>0</startAddress>
--- Start Address on Modbus device
<length>3</length>
--- no. of register value
<value>10,20,10.1</value>
--- Write registers value to Modbus device

<device>\\.\COM12</device>
--- RTU device COM port
<baud>115200</baud>
--- RTU device baud rate
<parity>N</parity>
--- RTU device parity (N, E, O)
<dataBit>8</dataBit>
--- RTU device data bit (5,6,7,8)
<stopBit>1</stopBit>
--- RTU device stop bit (1,2)

[Response]

```
<?xml  version="1.0"  encoding="UTF-8"?>
```

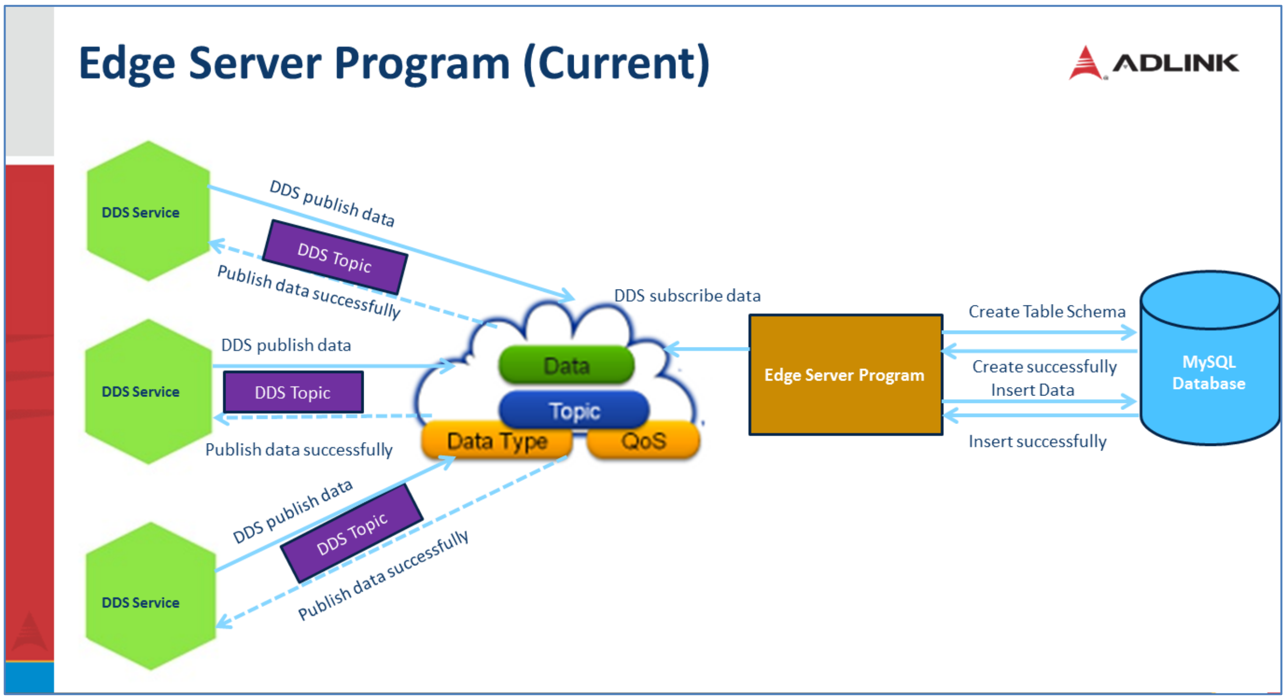
```
<writeRegisters>
            <startAddress>0</startAddress>
            <length>3</length>
            <values>10,20,10.1</values>
            <response>Success</response>
</writeRegisters>
```

5.2 Modbus REST API Error Message Reference

This section describes which situation you will get the error message for the Modbus REST response.
TBD

6. Edge Server

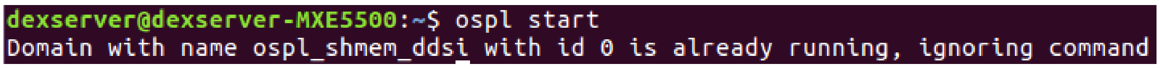
After you start the Vortex OpenSplice, you need to start the edge server program. The edge server program is the first starting program before every DEX-100 DDS module starting because it must subscribe all DEX-100 devices' extraction data and creates table schema automatically.



6.1 Starting Edge Server Program

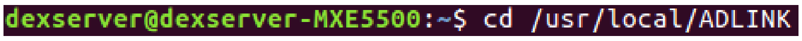
The starting Edge Server Program step by step is as follows:

1. Open the terminal
2. Check your Vortex OpenSplice already started
   "ospl start"



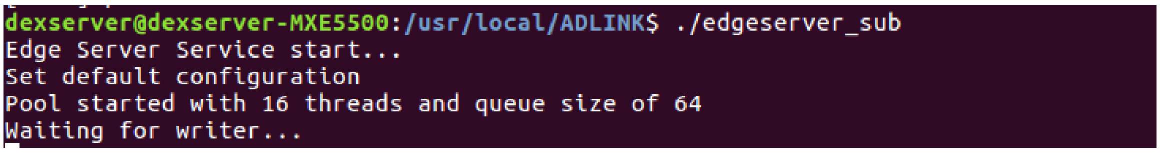
```
dexserver@dexserver-MXE5500:~$ ospl start
Domain with name ospl_shmem_ddsi with id 0 is already running, ignoring command
```

3. If Vortex OpenSplice already started, change path to /usr/local/ADLINK
   "cd /usr/local/ADLINK"



```
dexserver@dexserver-MXE5500:~$ cd /usr/local/ADLINK
```

4. Start program ./edgeserver_sub, then the program will subscribe the DDS data if have any data publishing

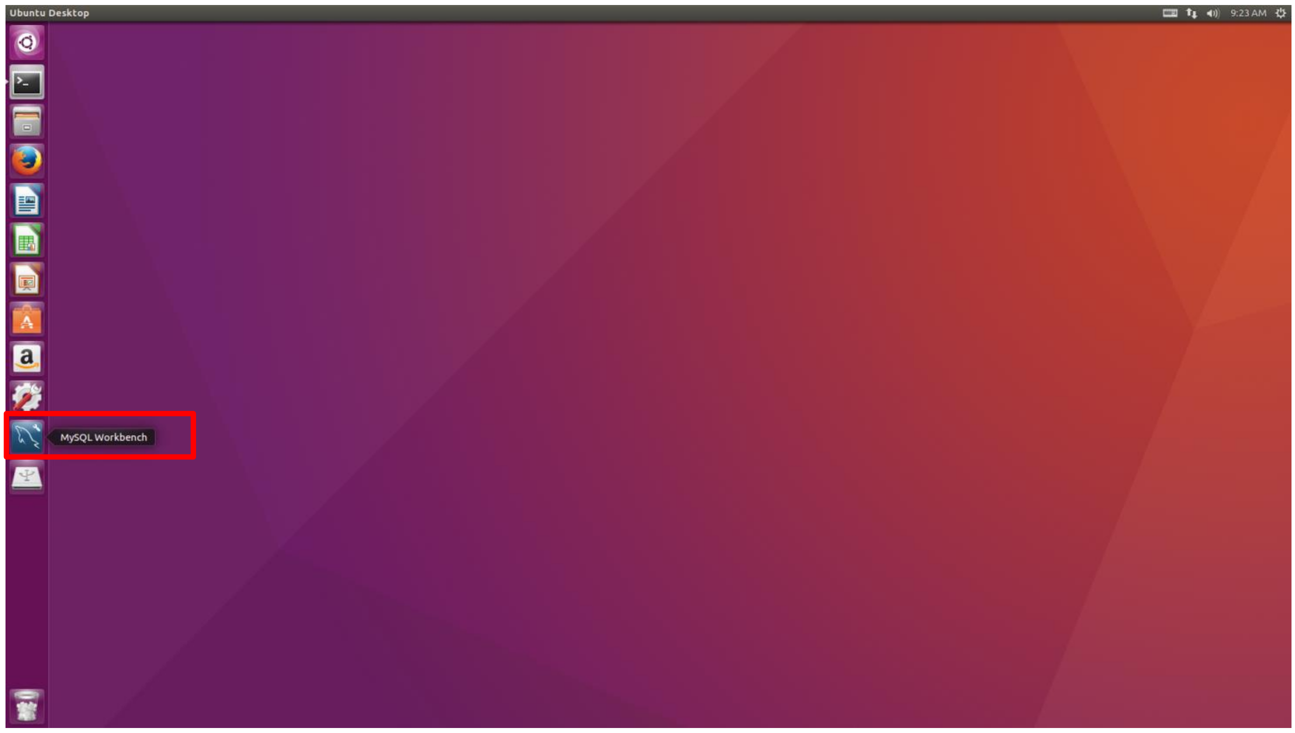


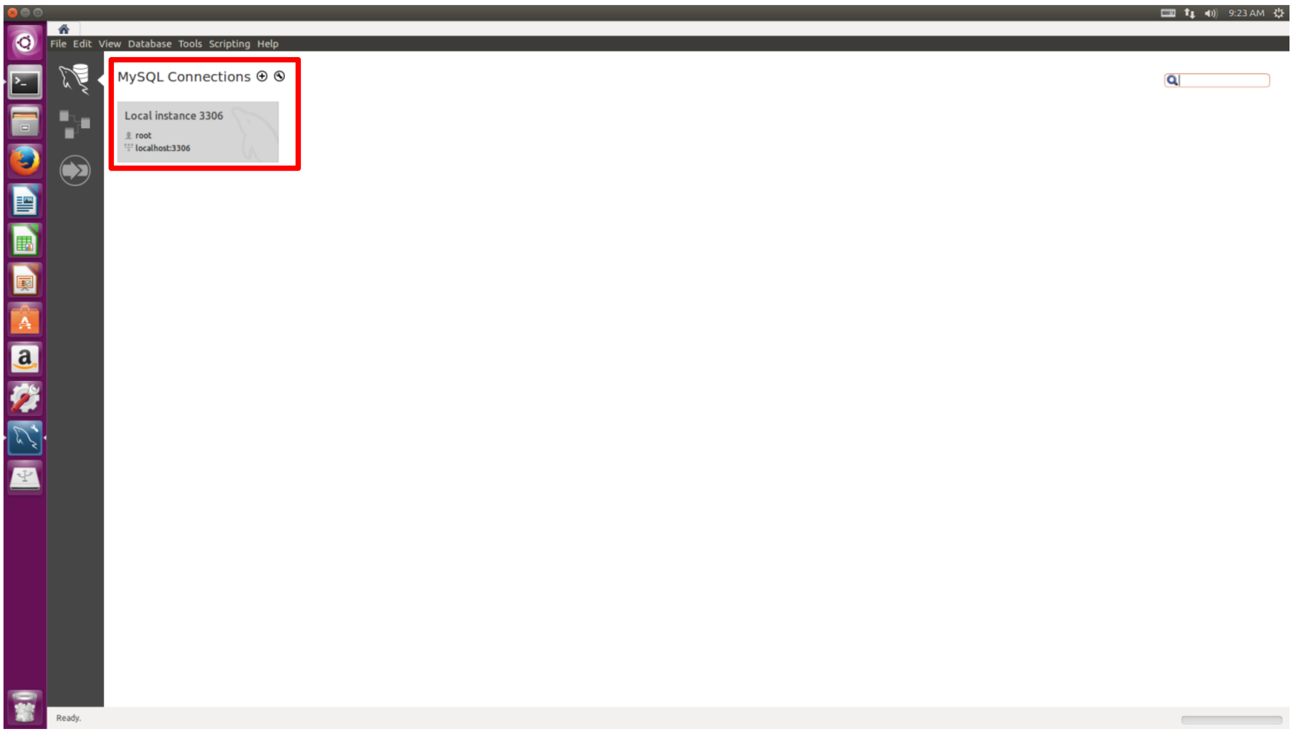
```
dexserver@dexserver-MXE5500:/usr/local/ADLINK$ ./edgeserver_sub
Edge Server Service start...
Set default configuration
Pool started with 16 threads and queue size of 64
Waiting for writer...
```

6.2 Database

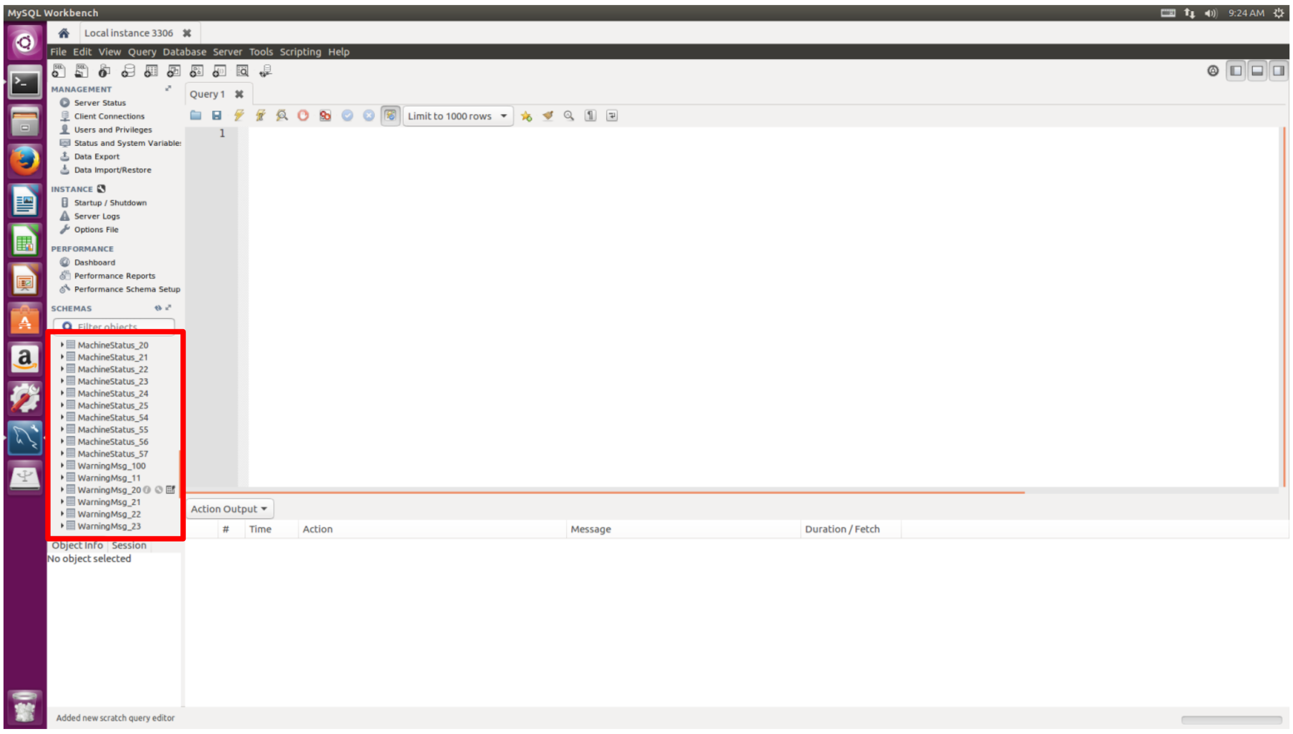
How to check the machine data in the database?

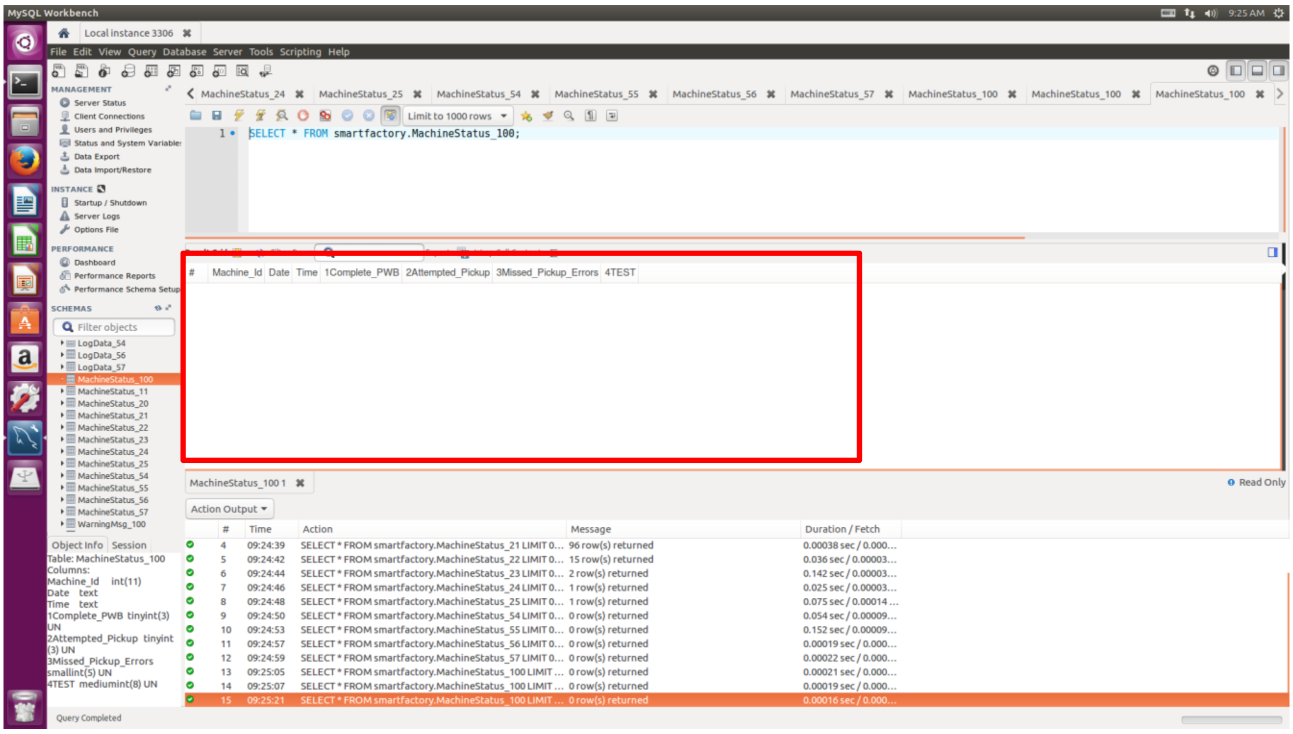
1. Open the MySQL workbench to see the data for all DEX-100





2. One DEX-100 will allocate 4 tables: LightColor_% machineid %, WarningMsg_%machineid, AlarmMsg_%machineid%, MachineStatus_%machineid%





```
SELECT * FROM smartfactory.MachineStatus_100;
```

#	Machine_Id	Date	Time	1Complete_PWB	2Attempted_Pickup	3Missed_Pickup_Errors	4TEST

```
Table: MachineStatus_100
Columns:
Machine_Id    int(11)
Date    text
Time    text
1Complete_PWB  tinyint(3)
UN
2Attempted_Pickup  tinyint
(3) UN
3Missed_Pickup_Errors
smallint(5) UN
4TEST  mediumint(8) UN
```

```
#   Time      Action                                          Message             Duration / Fetch
4   09:24:39  SELECT * FROM smartfactory.MachineStatus_21 LIMIT 0...   96 row(s) returned   0.00038 sec / 0.000...
5   09:24:42  SELECT * FROM smartfactory.MachineStatus_22 LIMIT 0...   15 row(s) returned   0.036 sec / 0.00003...
6   09:24:44  SELECT * FROM smartfactory.MachineStatus_23 LIMIT 0...   2 row(s) returned    0.142 sec / 0.00003...
7   09:24:46  SELECT * FROM smartfactory.MachineStatus_24 LIMIT 0...   1 row(s) returned    0.025 sec / 0.00003...
8   09:24:48  SELECT * FROM smartfactory.MachineStatus_25 LIMIT 0...   1 row(s) returned    0.075 sec / 0.00014 ...
9   09:24:50  SELECT * FROM smartfactory.MachineStatus_54 LIMIT 0...   0 row(s) returned    0.054 sec / 0.00009...
10  09:24:53  SELECT * FROM smartfactory.MachineStatus_55 LIMIT 0...   0 row(s) returned    0.152 sec / 0.00009...
11  09:24:57  SELECT * FROM smartfactory.MachineStatus_56 LIMIT 0...   0 row(s) returned    0.00019 sec / 0.000...
12  09:24:59  SELECT * FROM smartfactory.MachineStatus_57 LIMIT 0...   0 row(s) returned    0.00022 sec / 0.000...
13  09:25:05  SELECT * FROM smartfactory.MachineStatus_100 LIMIT ...   0 row(s) returned    0.00021 sec / 0.000...
14  09:25:07  SELECT * FROM smartfactory.MachineStatus_100 LIMIT ...   0 row(s) returned    0.00019 sec / 0.000...
15  09:25:21  SELECT * FROM smartfactory.MachineStatus_100 LIMIT ...   0 row(s) returned    0.00016 sec / 0.000...
```