


**(ICBMS-3 specification) The IoT Platform for Virtual
Things, Distributed Autonomous Intelligence and Data
Federation/Analysis**

SO Framework Development Guide

Summary: This guide describes how to develop by applying the SO framework.

	SO Framework Development Guide	
Project name:	Step:	Team in Charge:



Document Approval

If approved as an attached document of the main document, it is replaced with the signature of the main document.

Company : PINEONE Communications Co., Ltd.

Wonwook Park
Development Team

Date

Supervisor: PINEONE Communications Co., Ltd.

Taecheol Kim
Development Team

Date

Revised history

[illegible]

INDEX

1.	Development Guide Overview	6
2.	SO Server Internal Configuration Module	7
3.	Source build environment	9
3.1	Downloading and Installing the JDK	9
3.2	Downloading and installing STS	9
3.3	Downloading and decompressing the Oasis SO Framework source	11
3.4	Configuring SO project build environment	12
3.5	Build with Gradle	14
4.	Setting SO server image execution environment	16
4.1	Downloading and Installing the JDK	16
4.2	Kafka / Installation and Execution	16
4.2.1	kafka installation	16
4.2.2	zookeeper / kafka execution	17
4.3	Downloading and Installing MariaDB	18
4.3.1	MariaDB Download and Installation Guide	18
4.3.2	DB Collection and Indexing creation	18
4.4	Setting SO Server execution environment	18
4.5	Running the server in eclipse	20
4.5.1	Setting the server execution environment	20
4.6	Running the server in a distribution environment	24
4.6.1	Setting SO running environment	24
4.6.2	SO execution	25
4.6.3	Check SO Server Operation	25
5.	SO Framework Server TEST	25
5.1	Service Orchestration (SO) operation check by POSTMAN	25
6.	Q&A	28



SO Framework Development Guide

Project name:

Step:

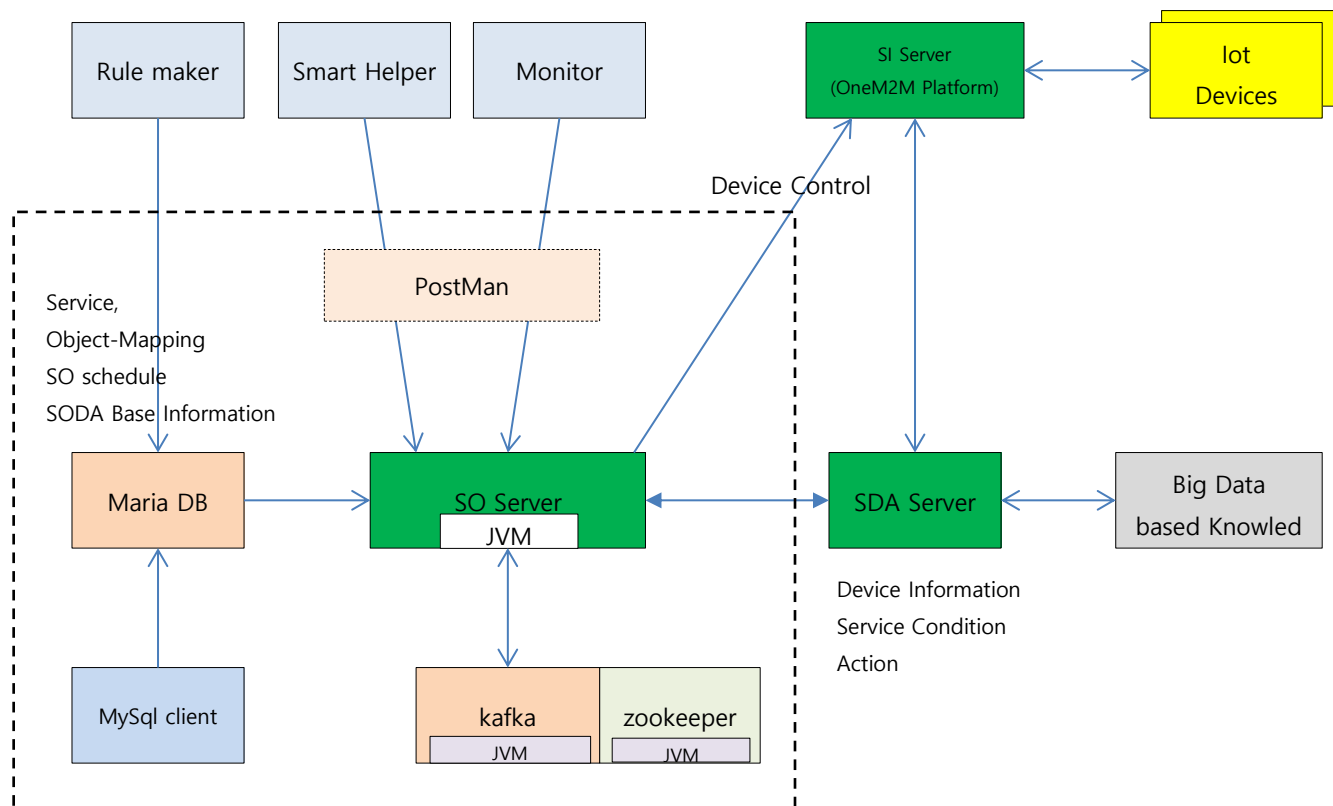
Team in Charge:

1. Development Guide Overview

Oasis (Open-source Architecture Semantic IoT Service-platform) project aims to develop an open source, intelligent IoT(Internet of Things) service platform based on international standards.

The Oasis project will be provided as open source as the outcome of the "(ICBMS-3 specifics) The IoT Platform for Virtual Things, Distributed Autonomous Intelligence and Data Federation/Analysis" as a newly supported by IT & broadcasting technology development project (2015).

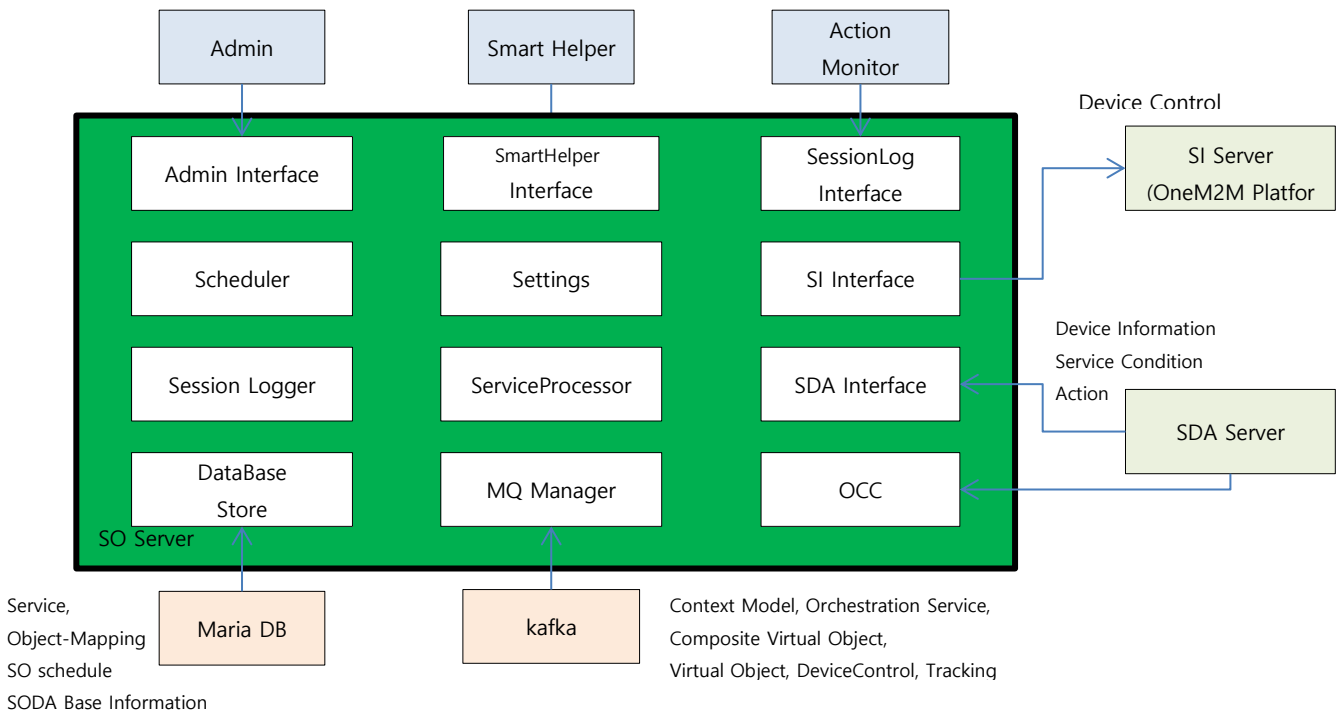
The systems used in the Oasis project consist of a knowledge database based on the data collected from SI Server and SI Server which control IOT devices, receive sensor data from devices, and an providing interface for situation judgment, SO server that provides context-aware services with a knowledge base built on SDA, and a database server, a message queue server, a service creation tool, and a service request application that each server requires.



For new users to approach Oasis SO Framework, this document will guide you through the

installation and execution of the components in the box marked with a dotted line around the SO Server by downloading the source and building the development environment easily.

2. SO Server Internal Configuration Module



- Scheduler : Based on the set schedule data, check whether or not the CM is generated from the SDA server
- Settings : Peripheral server environment settings, server operation environment management
- SI Interface : IOT Device Forward control data to SI Server server
- SDA Interface : Interface that acquires CM generation information and knowledge base device configuration information from SDA server
- OCC : Interface for handling emergency services by SDA Server
- Service Processor : Interprets the authoring data, dynamically creates a virtual object for each layer, and generates corresponding device control information
- MQ Manager : Message is put into Message Queue Server (KAFKA)
- SessionLogger : Store or read log generated by each processing module by Session
- Database Store : Store or read data in the database(MariaDB)
Schedule, OS execution data by CM, Virtual Object control data by OS, CVO configuration

Project name:

Step:

Team in Charge:

data, Authoring data, CM list, Function list, Location list, Profile dependency, Authoring data, management, operation log data read-write management by session

- Admin Interface : Interface for schedule management and internal operation setting change by administrator
- SmartHelper Interface : Handling requested services from Smart Helper
- SessionLog Interface : Interface for obtaining service processing result per session

Project name:

Step:

Team in Charge:

3. Source build environment

To build SO Server source, you need windows, JDK, STS and SO Framework source.

- For linux, refer to the link below to install


[JDK Installation Guide](#)

This document describes how to configure the build environment using the following procedure in Windows 64bit environment.

1. Download and install the JDK
2. Download and install STS
3. Oasis SO Framework Source Download
4. Configure SO project build environment
5. Build

3.1 Downloading and Installing the JDK

1) Download

- Connect to <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
-  Click (recommendation: click Java SE 8uxxx)
- Select the Accept License Agreement and click the link for your OS

2) Installation

- Run downloaded jdk_xxx-windows-x64.exe

3.2 Downloading and installing STS

1) Download

- Connect to <https://spring.io/tools>
- DOWNLOAD STS (3.9.2.RELEASE for Windows) Click to download

2) Installation

- Unzip the downloaded zip file (eg spring-tool-suite-3.9.2.RELEASE-e4.7.2-win32.zip) to a suitable location (eg. C:\W)

Project name:

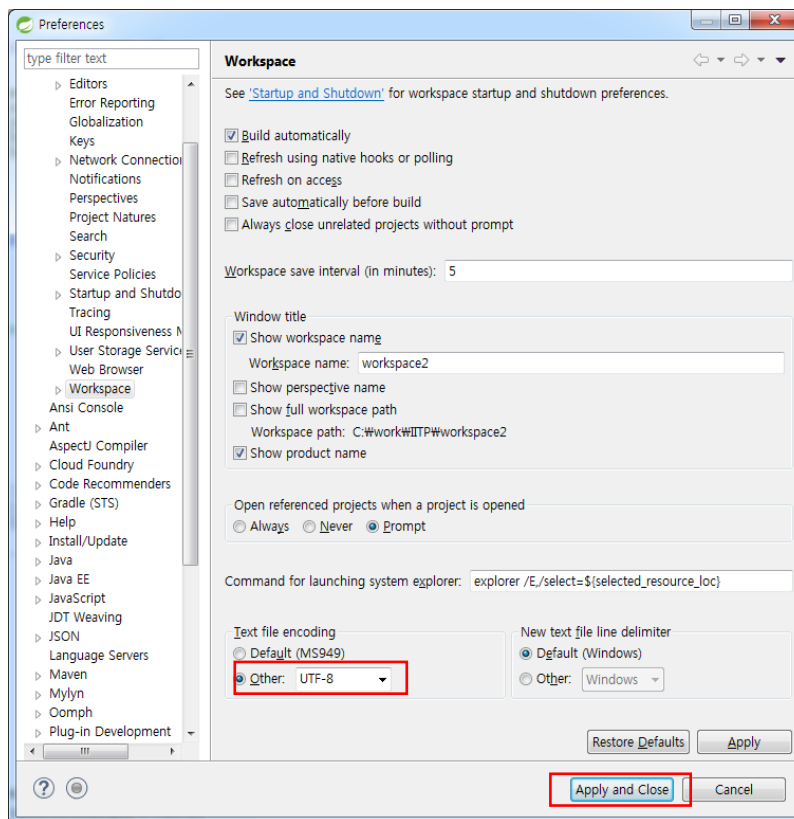
Step:

Team in Charge:

- Run STS.exe in the sts-bundle\sts-3.x.x.RELEASE folder

3) STS Environment Setting

- UTF8 setting
 - Click Preferences on the Windows menu
 - In General> Workspace, select Text file encoding as other, enter UTF-8, and click the Apply and Close button.

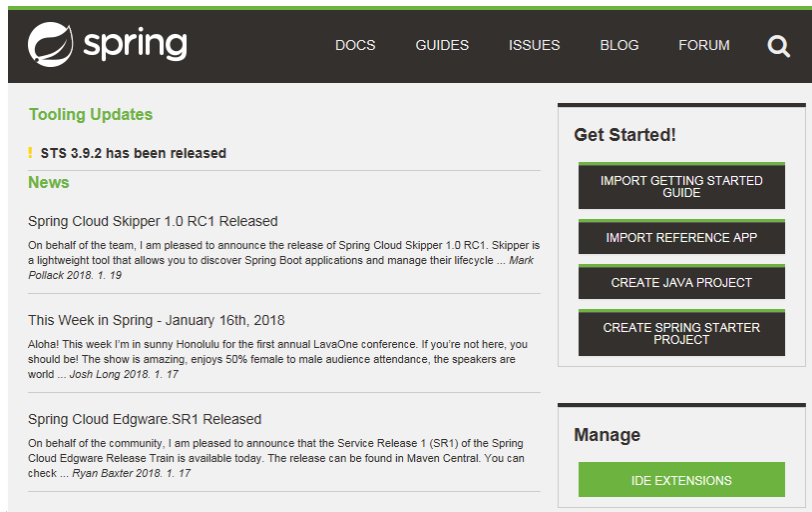


- Gradle installation
 - Click the IDE EXTENSIONS button in the Dashboard window.

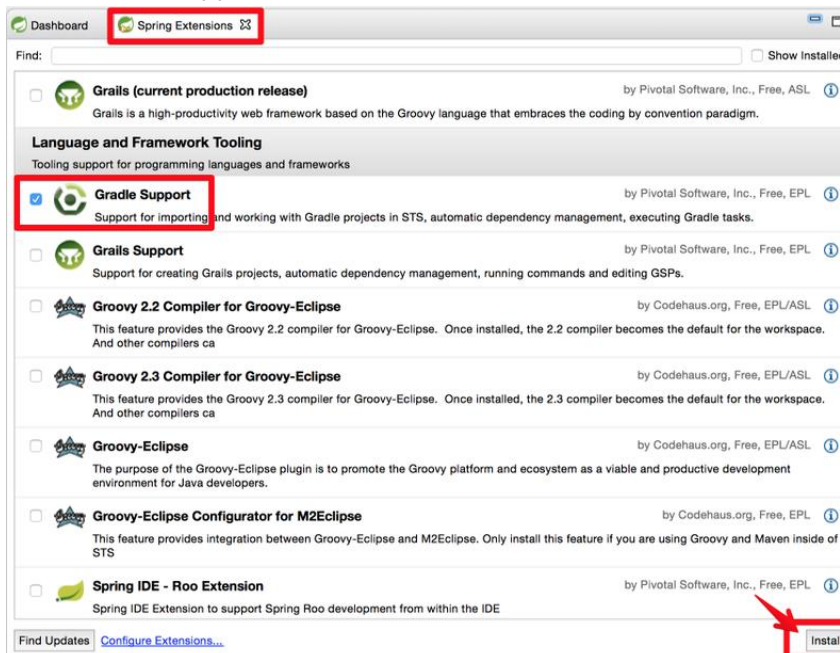
Project name:

Step:

Team in Charge:



Select Gradle Support and click the install button.



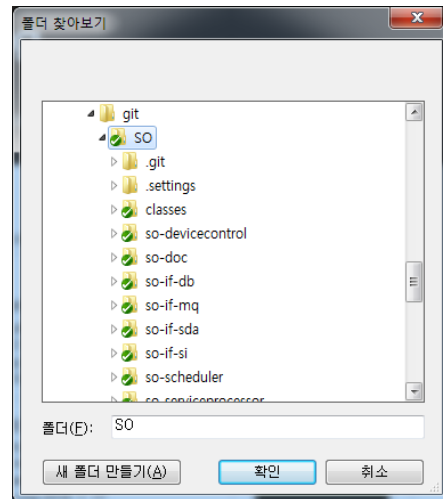
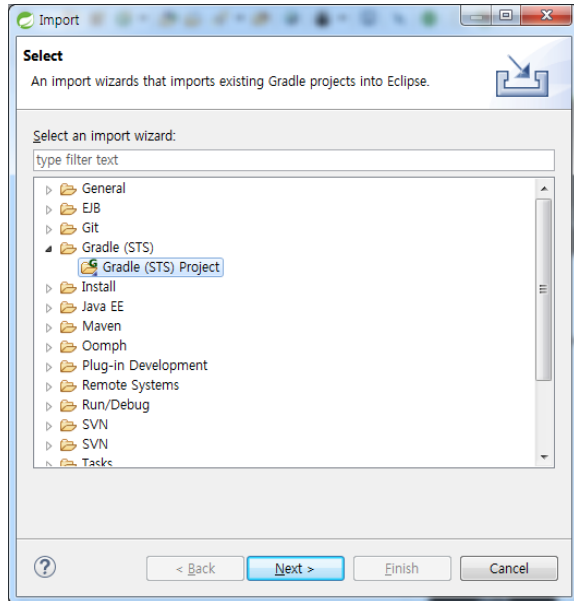
Click next to complete the installation.

3.3 Downloading and decompressing the Oasis SO Framework source

- Download the SO source and installation files from the [release page](#)
- Unzip the downloaded file

3.4 Configuring SO project build environment

- Open the Import window by selecting the File/Import menu from the menu, select Gradle (STS) > Gradle (STS) Project, and select the folder where the SO source is stored.

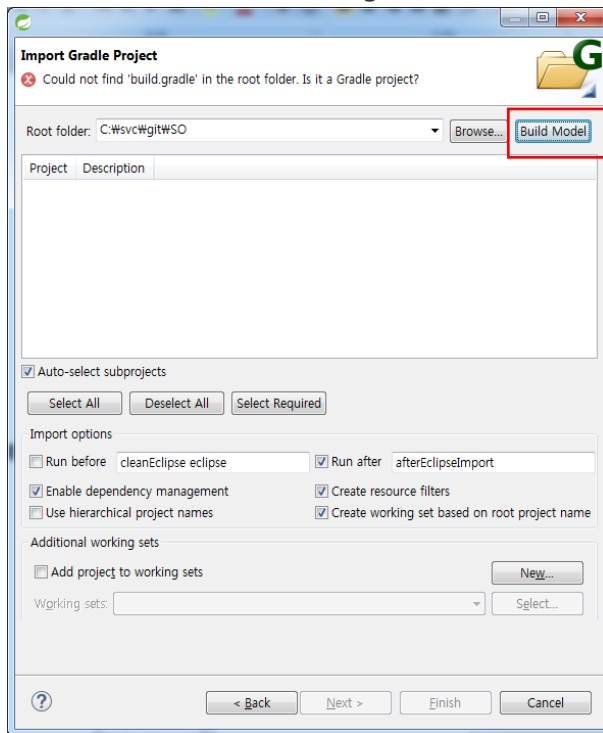


Project name:

Step:

Team in Charge:

- Click the "Build Model" button in the top right corner of the Import Gradle Project window as shown below. Generate build.gradle information.

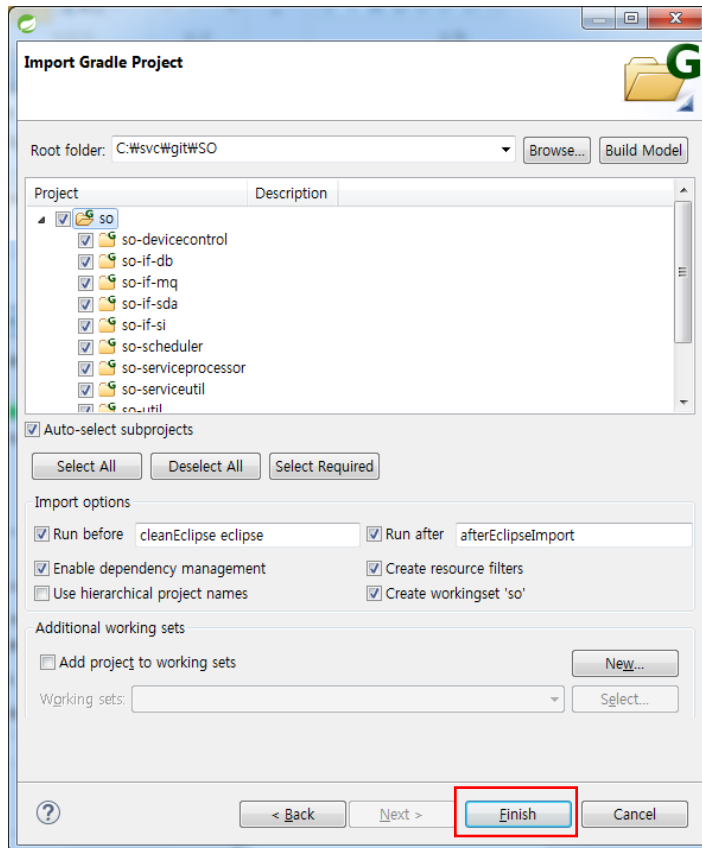


- Select the so directory and press the finish button.

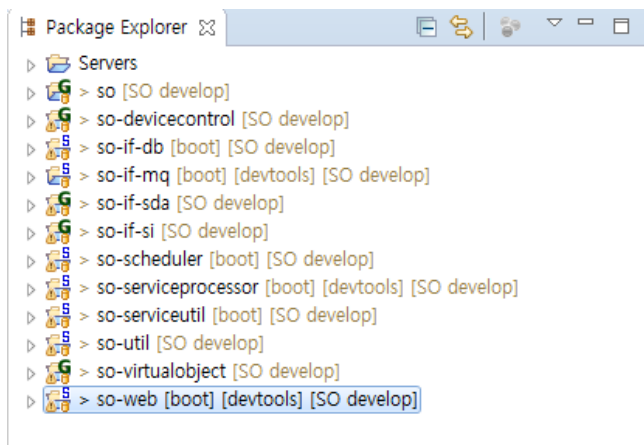
Project name:

Step:

Team in Charge:



- The imported project is displayed as below.



3.5 Build with Gradle

- Run Gradle Build with Ctrl+Alt+Shift+R.

Project name:

Step:

Team in Charge:

- Select project so-web. Type build command in Tasks and press Enter.

Tasks: build

Project: so-web

Type Gradle tasks and press 'Enter'

4. Setting SO server image execution environment

You can proceed to build SO Server execution environment in the following order.

1. Download and install the JDK
2. Install and run Zookeeper/Kafka
3. Install and run the MariaDB server
4. SO Server operating environment setting
5. Copy SO project build image
6. Run SO Server
7. Test with HTTP emulator(PostMan)

Zookeeper, Kafka, and SO Server require the JDK to work. Therefore, if you want to install Zookeeper, Kafka, and SO Server on different servers, you need to install the JDK on each server.

4.1 Downloading and Installing the JDK

- See the JDK installation guide when building 3.Source build environment

4.2 Kafka / Installation and Execution

4.2.1 *kafka installation*

1) kafka download

```
# wget http://apache.mirror.cdnetworks.com/kafka/0.11.0.2/kafka_2.11-0.11.0.2.tgz
```

2) Unzip and link:

```
# tar xvfz kafka_2.11-0.11.0.2.tgz
# ln -s kafka_2.11-0.11.0.2 kafka
```

3) Copy the scripts folder in so-doc of SO-Framework source from the release page to the folder where the downloaded kafka folder is saved.

Project name:

Step:

Team in Charge:



kafka



scripts

4.2.2 zookeeper / kafka execution

1) Run zookeeper

```
./scripts/1-startup-zookeeper.sh
```

2) Run kafka

```
./scripts/2-startup-kafka.sh
```

3) Create kafka topic

```
▪ ./scripts/create-topics-so.sh
```

4) Check topic

```
▪ ./ scripts/topic-list-so.sh | grep "^Topic"
```

Make sure that the following seven topic lists are created

```
...
Topic:compositevirtualobject PartitionCount:1 ReplicationFactor:1 Configs:
Topic:contextmodel PartitionCount:1 ReplicationFactor:1 Configs:
Topic:devicecontrol PartitionCount:1 ReplicationFactor:1 Configs:
Topic:devicecontrol-seq PartitionCount:1 ReplicationFactor:1 Configs:
Topic:orchestrationservice PartitionCount:1 ReplicationFactor:1 Configs:
Topic:tracking PartitionCount:1 ReplicationFactor:1 Configs:
Topic:virtualobject PartitionCount:1 ReplicationFactor:1 Configs:
```

Caution! The zookeeper and kafka must be running while the SO server is running. (Used as queue in SO server)

Reference : [kafka running test](#)

Project name:

Step:

Team in Charge:

4.3 Downloading and Installing MariaDB

4.3.1 *MariaDB Download and Installation Guide*

To download and install MariaDB, refer to the link below.

<https://downloads.mariadb.org/>

4.3.2 *DB Collection and Indexing creation*

Download the [MariaDB Script](#) (db_script.txt) in the so-doc folder from the [release page](#) and use the mysql client tool to create the default collection and index.

```
$ mysql -u root -p
```

Enter password:

```
MariaDB [(none)]> source db_script.txt
```

4.3.2.1 Check the created DB.

```
MariaDB [so.open]> show tables;
```

```
+-----+
| Tables_in_so.open |
+-----+
| CM_LocationTemp   |
| composite_virtual_object |
| context_model     |
| cvo_vo            |
| device1121        |
| device_value      |
| functionality      |
| location          |
| noncvo            |
| orchestration_service |
| physical_device_type |
| profile           |
| profile_dep       |
| rule_body         |
| rule_item         |
| session_data      |
| session_data_device |
| session_data_location |
| session_data_vo   |
| smart_helper      |
| tracking          |
| virtual_object    |
| vo_value_type     |
+-----+
23 rows in set (0.00 sec)
```

4.4 Setting SO Server execution environment

Project name:

Step:

Team in Charge:

The settings are written in the application-xxx.properties file and exist in the conf folder in the so-web folder.

For development : application-dev.properties

For operation : application-product.properties

The meanings of the setting items are as follows.

Item	Description
server.port	Server Port Settings
server.context-path	Server main-path setting
logging.config	SO log file information file
spring.datasource.url	Database connection information
sda.connection.uri	SDA server connection information
spring.datasource.username	DB account information
spring.datasource.password	DB password
spring.jackson.time-zone	Server time setting (Asia/Seoul recommendation)
mq.broker.list	kafka connection information
mq.zookeeper.list	Zookeeper connection information
sda.connection.uri	SDA server connection information
si.connection.uri	OneM2M SI server connection information
lwm2m.connection.uri	LWM2M SI server information
so.connection.uri	SO server information

The below are sample configuration files.

Project name:

Step:

Team in Charge:

```
server.port=10080
server.contextPath=/so

spring.datasource.url =jdbc:mysql://127.0.0.1:33306/so?useUnicode=true&characterEncoding=utf-8&autoReconnect=true
spring.datasource.username=user
spring.datasource.password=pw
spring.datasource.driverClassName=com.mysql.jdbc.Driver
spring.jackson.time-zone=Asia/Seoul

mq.broker.list=localhost:9092
mq.zookeeper.list=localhost:2181

sda.connection.uri=http://127.0.0.1:30080/sda/ctx/
si.connection.uri=http://127.0.0.1:30081
lwm2m.connection.uri=http://127.0.0.1:50081
so.connection.uri=http://127.0.0.1:${server.port}
```

4.5 Running the server in eclipse

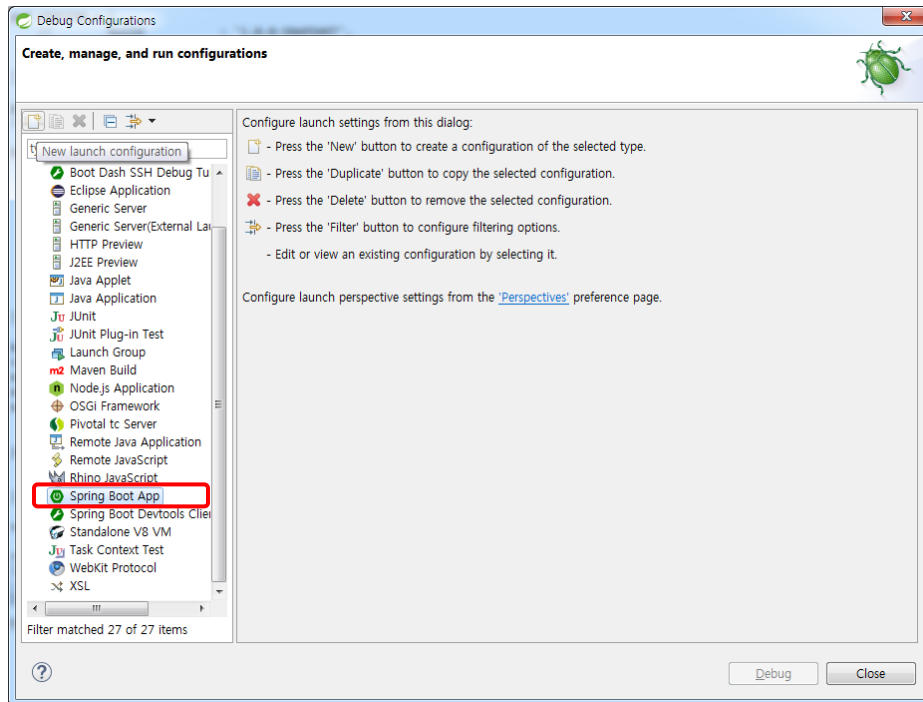
4.5.1 *Setting the server execution environment*

- 1) Select Debug Configuration from the Run submenu of the Main Menu.
- 2) Select the Spring Boot App item and click the New Launch Configuration button (or double-click the Spring Boot App item)

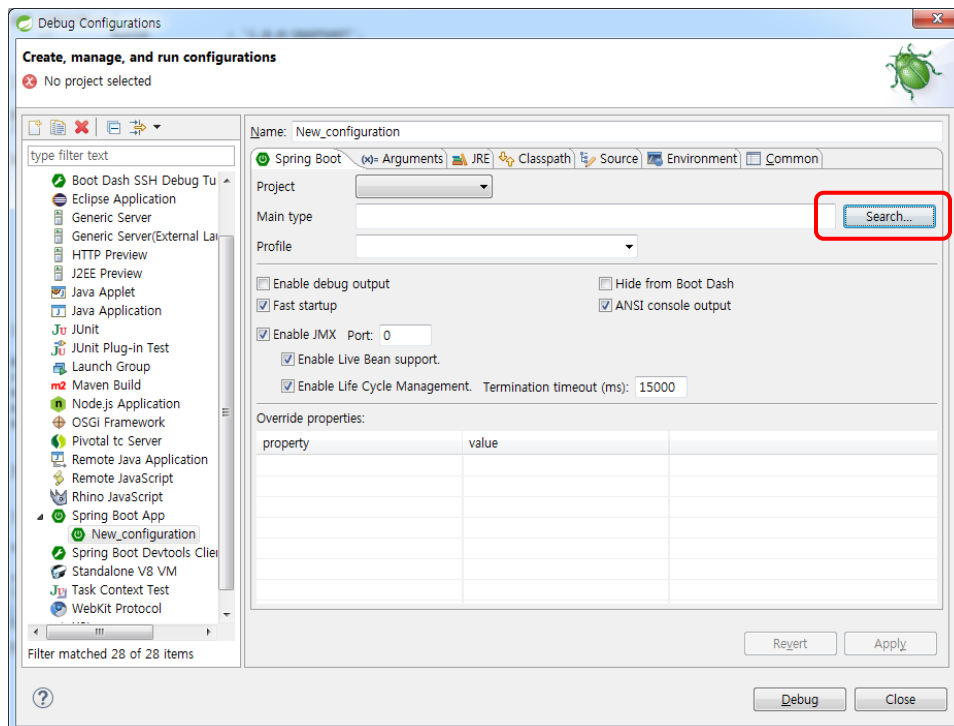
Project name:

Step:

Team in Charge:



3) The Sub menu will appear so that you can specify the New configuration as shown below.

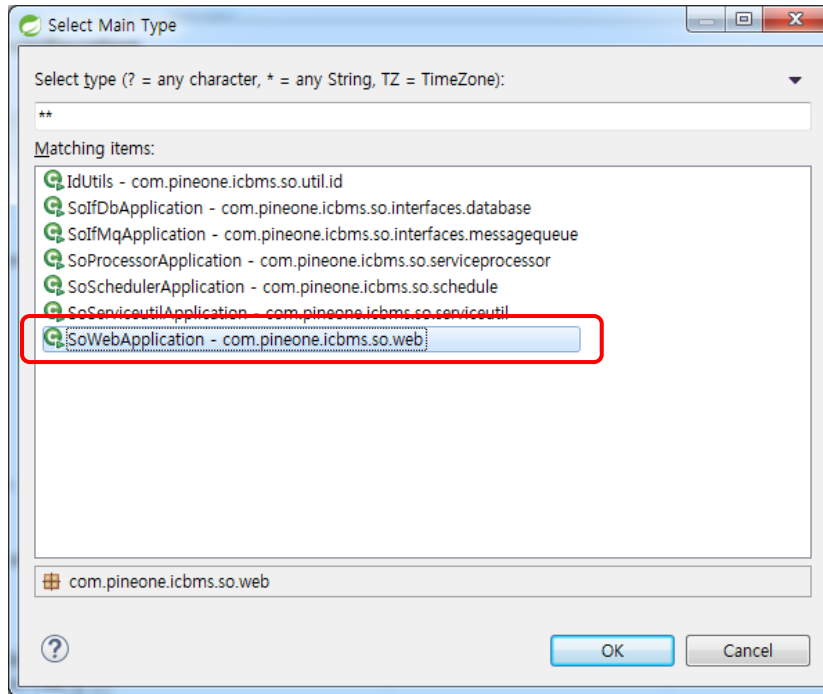


Project name:

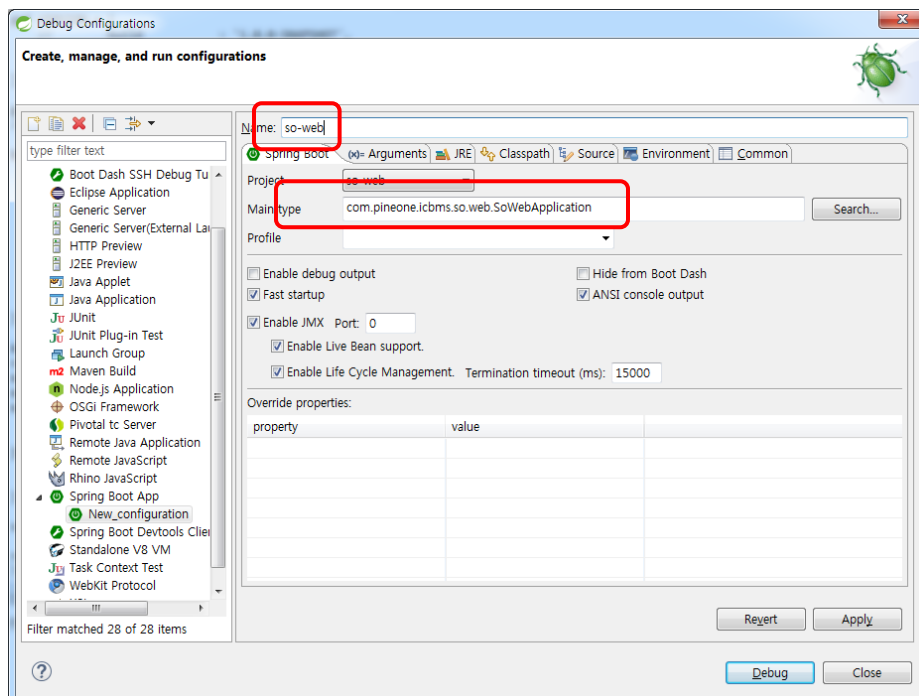
Step:

Team in Charge:

4) Press the search button to enter the Main Type, then the following window appears and select SoWebApplication.



5) In the Name field, type so-web.

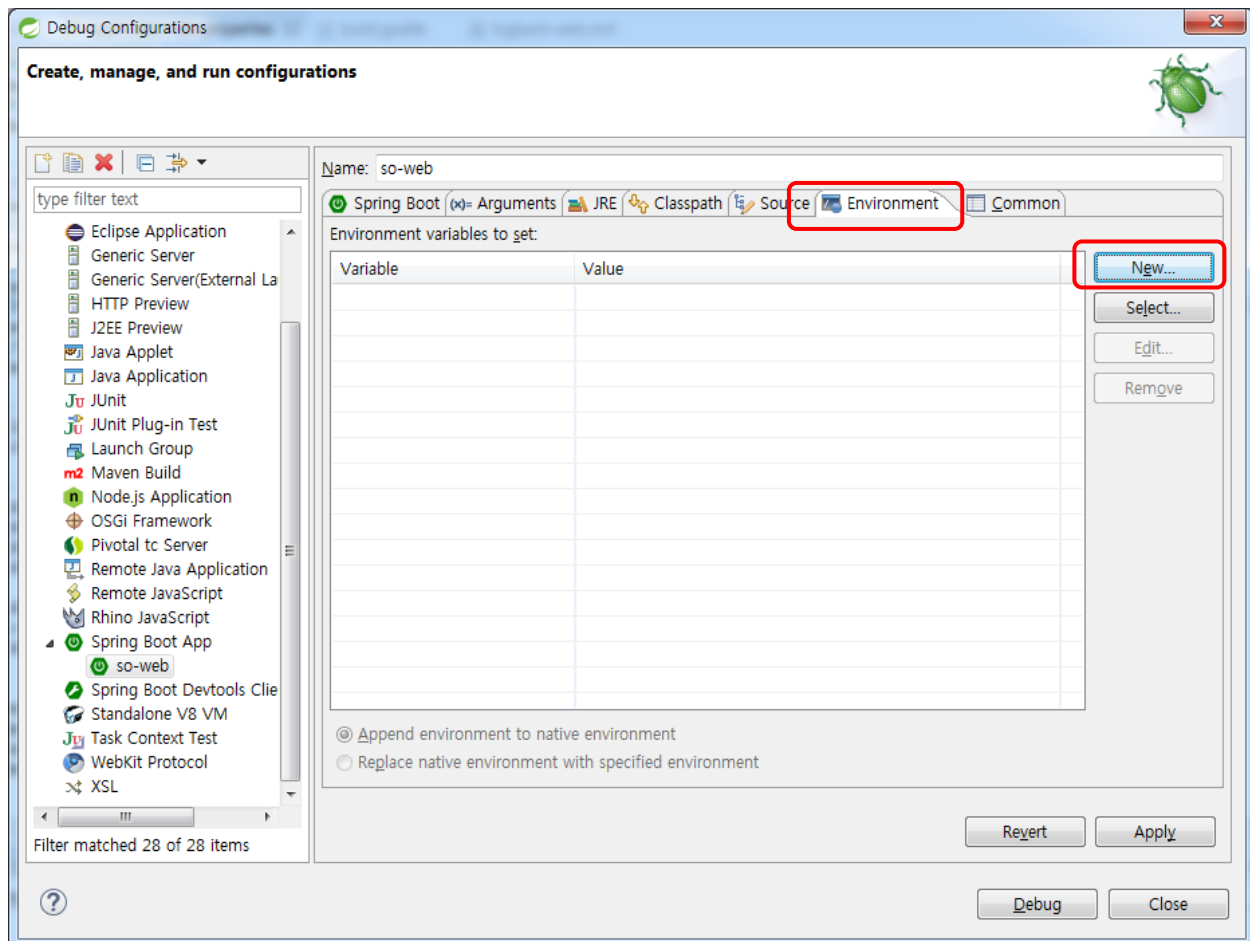


Project name:

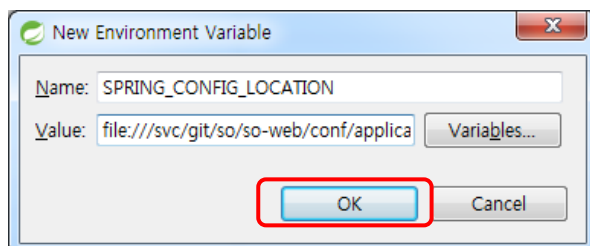
Step:

Team in Charge:

6) Select the Environment tab and press the New button.



7) Enter `SPRING_CONFIG_LOCATION` in the Name field and `file:///svc/git/so/so-web/conf/application-dev.properties` in the Value field where `application-dev.properties` is located.(modify if the path is different) and click OK.

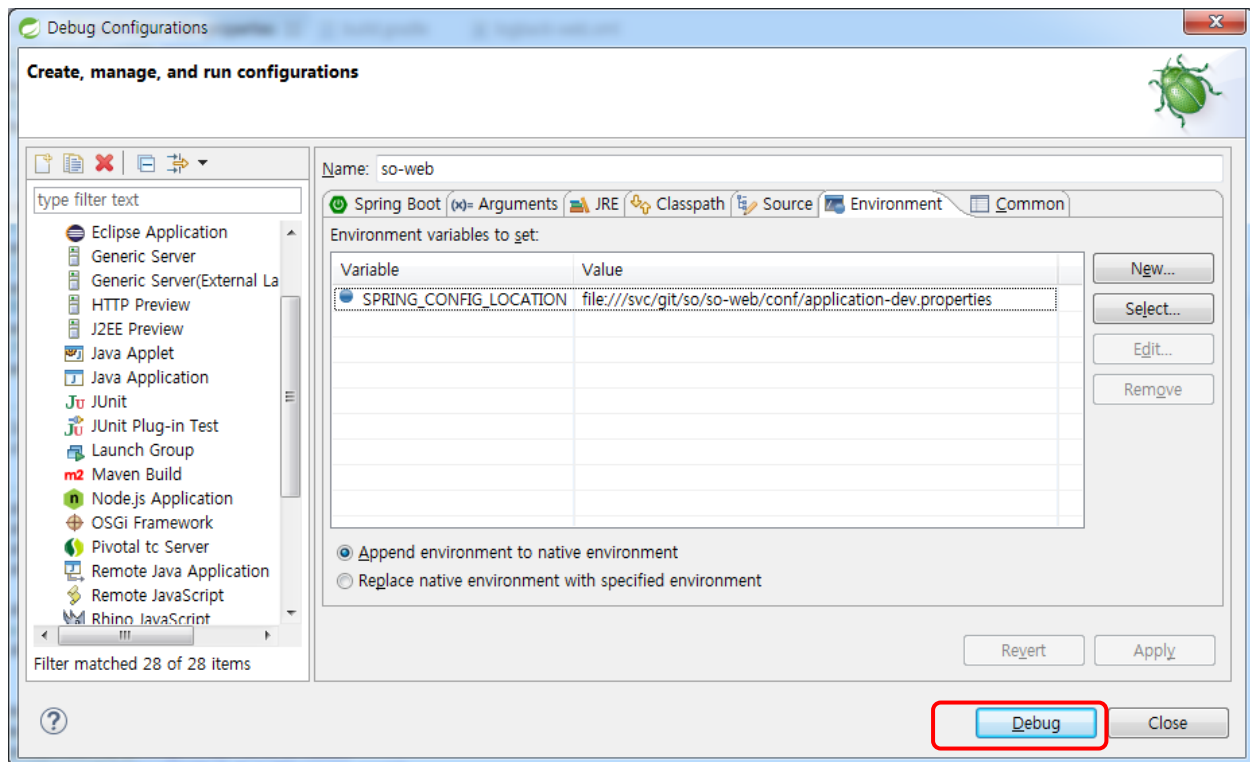


Project name:

Step:

Team in Charge:

8) Press the Debug button to start the server in Debug mode.



4.6 Running the server in a distribution environment

4.6.1 Setting SO running environment

Configure the environment only once at the first running. If the setting does not change, replace only the so server running file.

- 1) Install the JDK.
- 2) After installing and running kafka and zookeeper, create a topic.
- 3) Install and run MariaDB.
- 4) Create tables and indexes in MariaDB.
- 5) Create a database for the SO service in Maria DB.
- 6) Copy the environment used in the build environment to the server.

```
$ cd /home/myhome/
```

```
$ mkdir -p /svc/apps/so/so/conf
```

```
$ cp application-product.properties /svc/apps/so/so/conf/
```

```
$ cp logback-web.xml /svc/apps/so/so/conf/
```

- 7) Modify the application-product.properties and logback-web.xml files to match the server

Project name:

Step:

Team in Charge:

operating environment.

4.6.2 SO execution

1) Copy the so-3.0.0.0.jar file from the build target folder (so-web/build/libs) to the location of the server you want to run.

```
$ cp so-3.0.0.0.jar /svc/apps/so/so
```

2) Run the server

```
$ java -jar -Dspring.profiles.active=dev /svc/apps/so-3.0.0.0.jar --  
spring.config.location=/svc/apps/so/so/conf/application-product.properties &
```

4.6.3 Check SO Server Operation

- Look at the log file generated in the log directory specified in logback-web.xml.
- Refer to the tracking table created in the database and the contents of the table created in session*.
- Check the session Log using POSTMAN.
- The script file for the POSTMAN test can be downloaded from the release page.
- More information on SO Server testing using POSTMAN can be found on the [Test page](#).

5. SO Framework Server TEST

It describes how to test the downloaded SO Framework server.

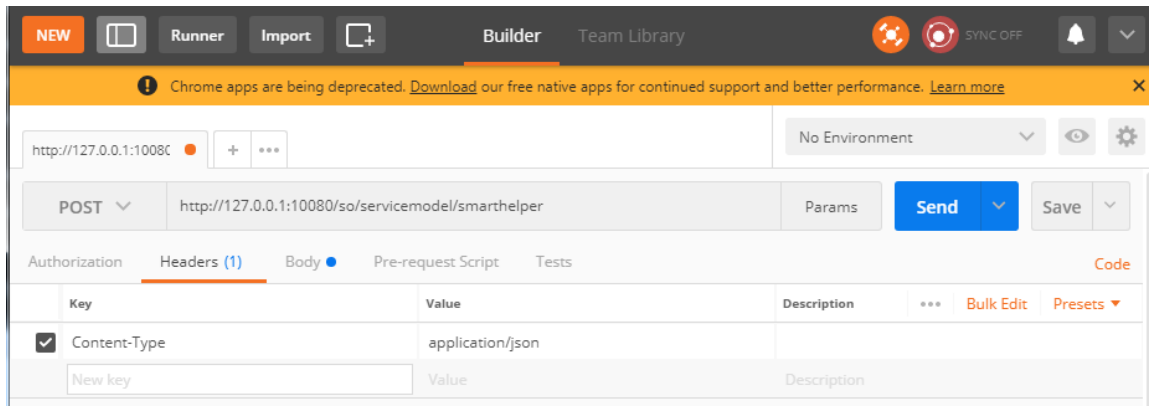
5.1 Service Orchestration (SO) operation check by POSTMAN

- Replace Method with POST.
- Enter **http://127.0.0.1:10080/so/servicemodel/smarthelper** in url
(if remote, enter remote server address)
- Select Headers and enter **Content-Type** and **application/json** in the Key and Value positions, respectively.

Project name:

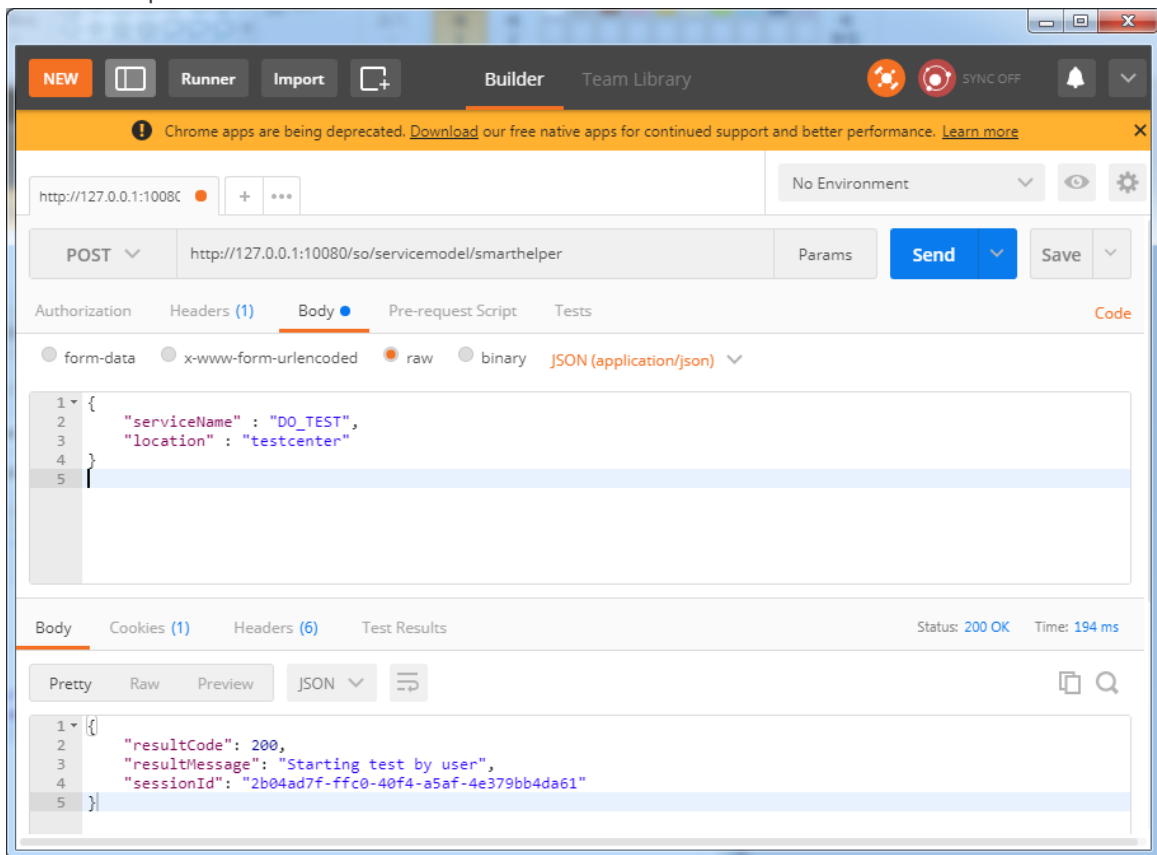
Step:

Team in Charge:



- Select Body, select raw, and type the following:

```
{
  "serviceName": "DO_TEST",
  "location": "testcenter"
}
```
- Click the Send button
- Check the operation result of SO Server.

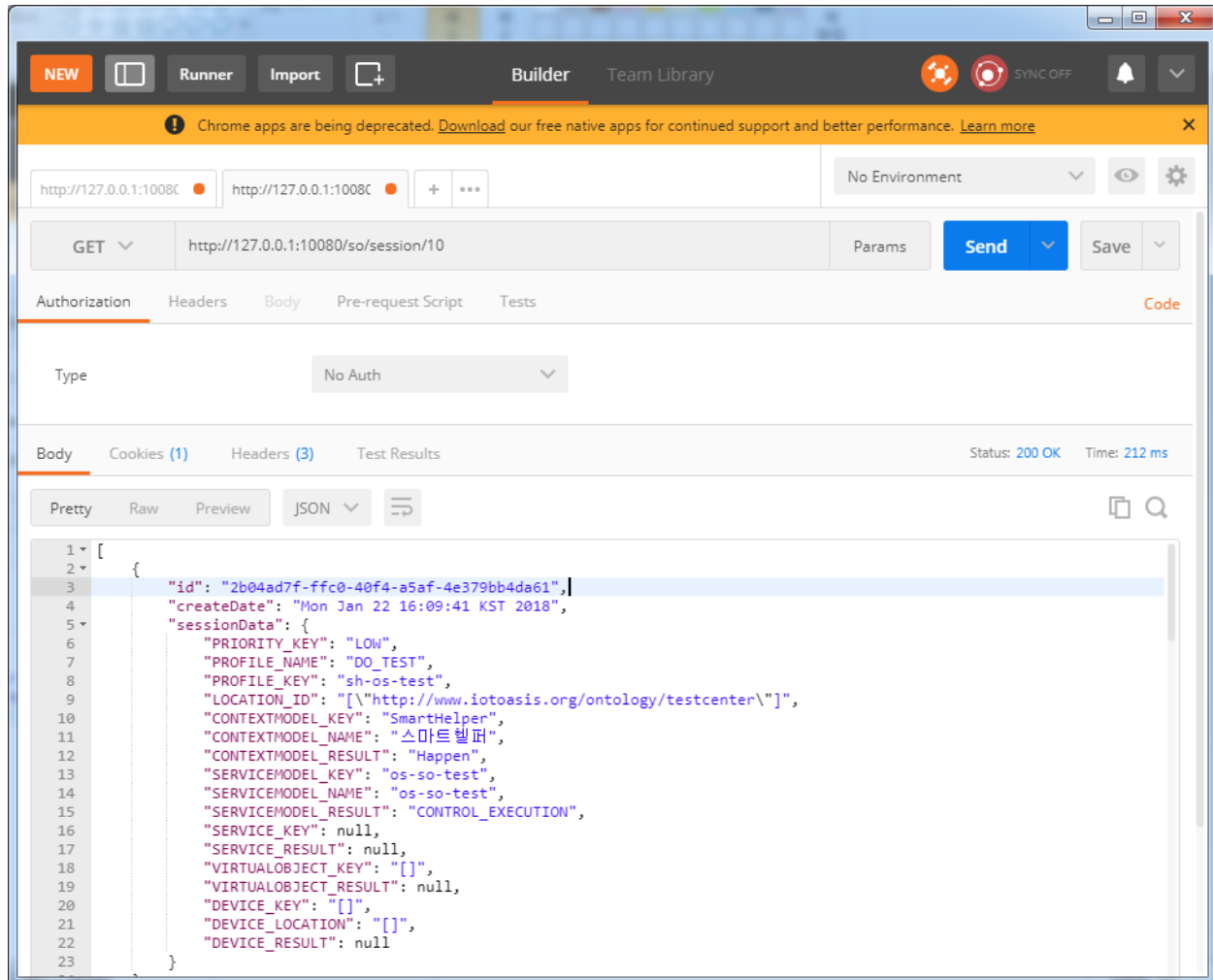


Project name:

Step:

Team in Charge:

- If the above command is successful, press the + button to create a new tab and enter `http://127.0.0.1:10080/so/session/10` in the URL
- Click the Send button
- Check the following session log in the result window.



Project name:

Step:

Team in Charge:

6. Q&A