MOZART Management Console

MMC Index

Introduction

MMC Overview

MOZART Management Console(MMC) is a MOZART Server management tool. One of the features in MMC is job scheduling which user can schedule jobs to the server to run models developed from MOZART IDE, sending e-mails or to run certain programs. These jobs can be triggered on a certain time or start/end of events. Other features of MMC are registering jobs, uploading model files, history management and distribution management

---

Main Concept of MOZART Job Scheduling

Job Scheduling consists of three management items like Job Type, Job, and Trigger.

Job Type

Job Type is type of Job that can be executed by Job Scheduler. There are three job types such as sending e-mails, running a program and Simple, Model, Collaboration tasks developed through MOZART. Among these three tasks, Model and Collaboration Task execution contents depend on target Model(check **MOZART Model Overview**) and arguments so Job Type has to be specified in advance. Only Simple Task does not need to be pre-defined because the task is not based on Model. (**How to manage Job Type**)

Job

Job is an execution object task configured with Job Type parameters. Jobs can only be triggered. JobType can be configured as several Jobs that works differently according to Argument configuration. As a summary, JobType and Job has 1 : N relationship. Job type for sending e-mail message for instance can have multiple jobs to send e-mails if the arguments for contact point and contents are different. (**How to manage Job**)

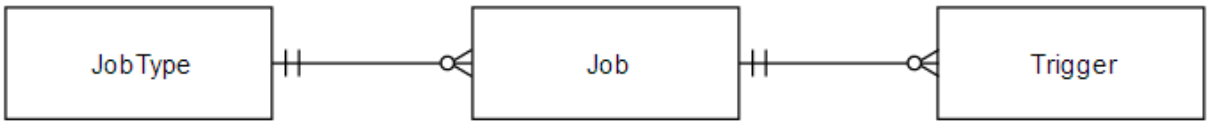
Trigger

Trigger is a set of information that defines conditions and its execution method for executing target job. Trigger can define and create a target job and its execution condition.

Job should be registered by MMC and its Job condition can be defined in two ways such as time-based or event-based. (**How to manage Trigger**)

- Time Scheduler(time-based) : Triggers job on a specific time or in a cycle.
- Condition(event-based) : Triggers job according to start/end of other trigger events.

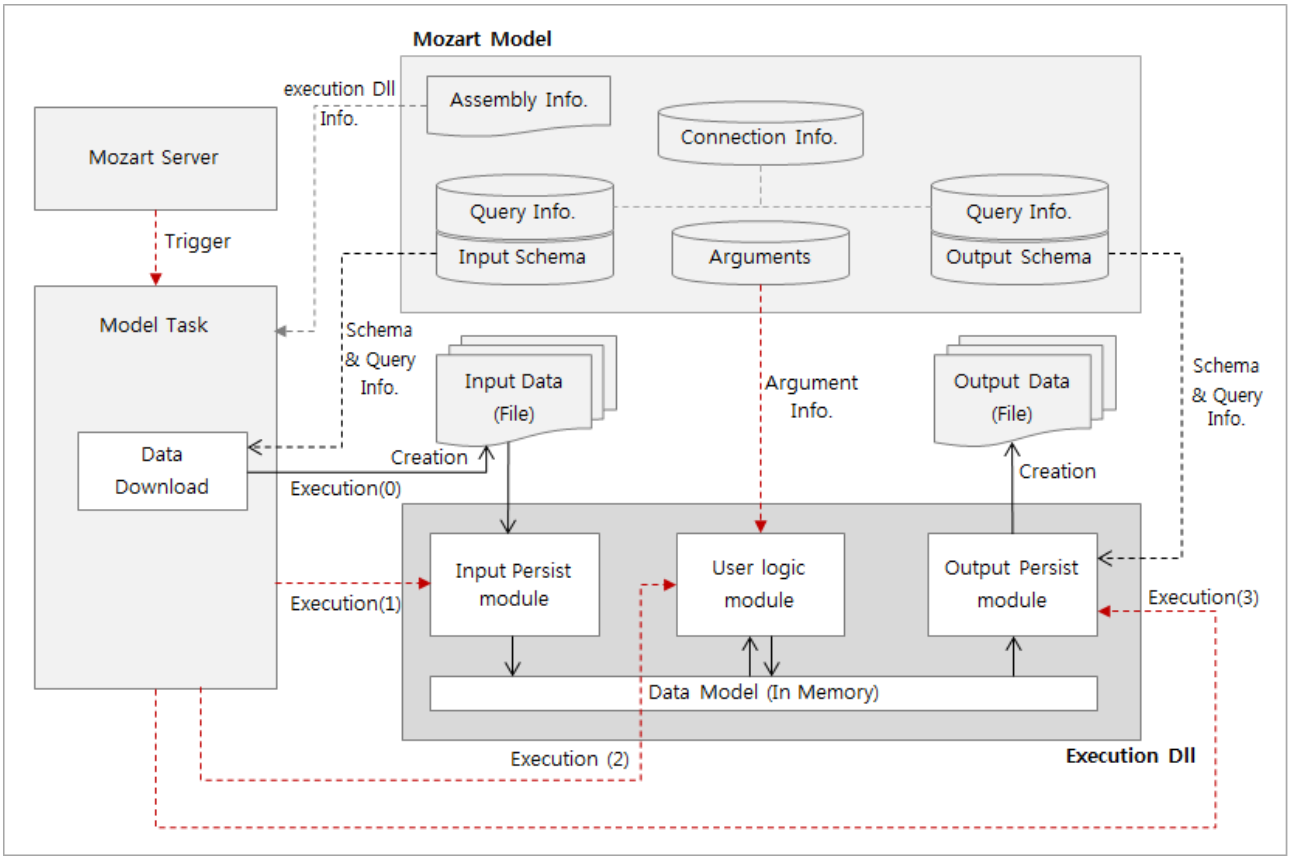
The following figure illustrates the relationship among main concepts of job scheduling.



MOZART Model Overview

MOZART Model is a set of Data that includes definitions about Input/Output data Schema, Query, Data access information that are used in a logic implemented through MOZART, and Arguments that are used for logic control. When a logic is implemented, Schema and Arguments are used. When a logic is executed, data access information and Query are used in order to retrieve and save data. Especially, execution needs Assembly information and access information that includes a logic executing the corresponding Model. So a Model includes these information. That is, executing MOZART Model requires Model file and its execution file.

There are two methods to execute this kind of Model on MOZART Framework. The first is to execute Model through MOZART Studio. This method is normally used when a developer executes Model for testing during development or performs various experiments with the same Model by changing input information. The second method is executed through MOZART Server. This is used to apply Model execution's result to operating system according to user scenario. The following figure illustrates Model's composition and operating structure.



Model Task is a default **Job Type** that is provided by MOZART. This has a role to execute a target Model by entering **Extended Arguments**' value for configuring how to execute tasks (execution(0) to (3)) that are executed by Model Task. MOZART Studio can run Model similarly to a method by Model Task, but basically most Model is executed with already created Input information so that a task to download Input data for executing a logic is executed by a separate menu if necessary. As a result, MOZART Studio executes execution(1) through (3) as a batch.

Job Type

Job Type is a type of job that can be performed by Job Scheduler. Job Type has Arguments that determine the execution method. Job Type can be managed by the Edit Job Type menu on the Job Management window. You also can select a job type when defining a Job.

The following sections are descriptions of the basic job types provided by the MOZART Job Scheduler.

Sending e-mail ($sendmail)

This job type is for sending an e-mail when it is executed. You should specify the sender, recipient, subject, body and attachments in the Job Type Arguments. Furthermore, you also configure the Outgoing Mail Server (SMTP). For administrative purposes, you may set it for sending e-mails when certain jobs fail to run by MMC.

Executing Program ($exec)

It starts a program or script. If you want to execute the program or script specified that the command line arguments are used, you can set these arguments in the “Add arguments (optional) text box”. In the “Start in (optional) text box”, you can specify a working directory on the command line where you run the program or script. This directory should be a path for program/script file or the file path used for the executable file. Programs that are built into Windows and executable files made by users are all executable if a user has a permission for accessing them.

Model Task ($model)

Model Task is a Job Type for executing model-based tasks developed by MOZART IDE. The Model Task type has default arguments (see **Extended Arguments**) for setting the

model's behavior (see **MOZART Model Overview**). The most basic argument for execution is the model information, which specifies a model in the Working Folder of the Server. The model to be executed defines the arguments to be set for execution as internal arguments, and these arguments should be set when defining the task.

Collaboration Task ($cola)

The Collaboration Task is a task that enables multiple tasks to collaborate with others through communication between tasks at the time of execution. You need to select the base model and set other models for the collaboration. (see **Extended Arguments**)

Job

Job is an execution object task configured with Job Type parameters. Jobs can only be executed through Triggers. The Job Type can be set to multiple Jobs that operate differently depending on the Arguments setting. Thus, Job Type and Job are in 1: N relationship.

The following are used to define Job.

Basic Information

Job Name : Name of Job. Jobs are categorized by job names.

Description : Job description.

Job Type : Job Type to be executed by the Job. You can select from a predefined Job Type (see **Job Type**) or user-defined Job. Job Type's Argument is changed according to the selected Job Type. For the Model Task and Collaboration Task (see **Model Overview**), you can select additional attributes. In order to execute a model-based task, a model should be specified by default with the following additional properties according to the execution method and collaboration method of the model.

- **Model file** : Path of Model file executed by Model Task. List of Model files will be displayed when Job is mapped to Project.
- **Model dll file** : This designates the developed dll file to execute Job.
- **Log dir** : This designates a folder that saves the corresponding Job's execution log. Log folder of Working folder is configured as base folder.
- **Additional run count (Optional)**: This configures the number of repeatable executions of a Job. If it is set as 0, no extra execution is triggered. If it is set as 1, the Job is executed twice. (refer to **How to configure More Run**)
- **Collaboration Count (Optional)**: The number of target jobs when the collaboration with other jobs is required when the job is executed. For Collaboration Task, multiple Jobs communicate and collaborate during execution. This attribute means the number of

Jobs to collaborate with. You can set the number of jobs to collaborate by this value. (see **How to configure collaboration task**)

Disallow Concurrent Execution : Option whether to allow simultaneous Job execution. If this option is enabled(checked), Same Jobs cannot be executed although Jobs are planned to start simultaneously in several Triggers.

Arguments

Each Job Type's Arguments

Configured Job Type's Argument. Sending e-mail has Argument like Sender, Receiver, Subject, and Main body, etc. and user-defined Job Type has Arguments that were made for configuration when Task was developed. Pre-defined Argument(refer to **Extended Arguments**) that is used in Model Task or Collaboration Task is also considered as Job Type arguments.

Model Argument

Model Argument is an Argument defined in a Model that is configured as an execution target Model by Model Task, Collaboration Task. Each Argument value configured in a Job is used as default value for parameter when Trigger is created. However, if Parameter's value is redefined in Trigger, the redefined value is reflected when the task is executed.

Trigger

Trigger is a set of information that defines the conditions for executing a target job and how to execute it. Trigger can be created by defining a target job and conditions for executing the job. This Job should be registered by MMC and Job condition can be defined in two ways, time-based and event-based methods.

The followings are components that defines Trigger.

Basic Information of Trigger and Job Execution Condition (Schedule Tab)

Trigger Name : Name of Trigger

Job Execution Condition(Settings) : Basically this condition can be configured to execute Job repeatedly in a specific cycle from a specific time. (a criterion). Also by connecting other Trigger's execution, user can assign a specific Trigger to be executed at the start or the end of the connected Trigger execution or to be fired when the preceding trigger returns True.

Additional Execution Condition(Advanced Settings) : Additional Trigger's execution condition such Trigger priority, run-time limitaion, etc. can be configured.

Target Job

Target Job : Target Job for execution can be registered by MMC only.

Job Parameter : This is used to configure value of Parameter that decides how to execute target job. Although Job has already been configured with a default value, this can be used when user wants to execute Trigger with the changed value.

Failure Action

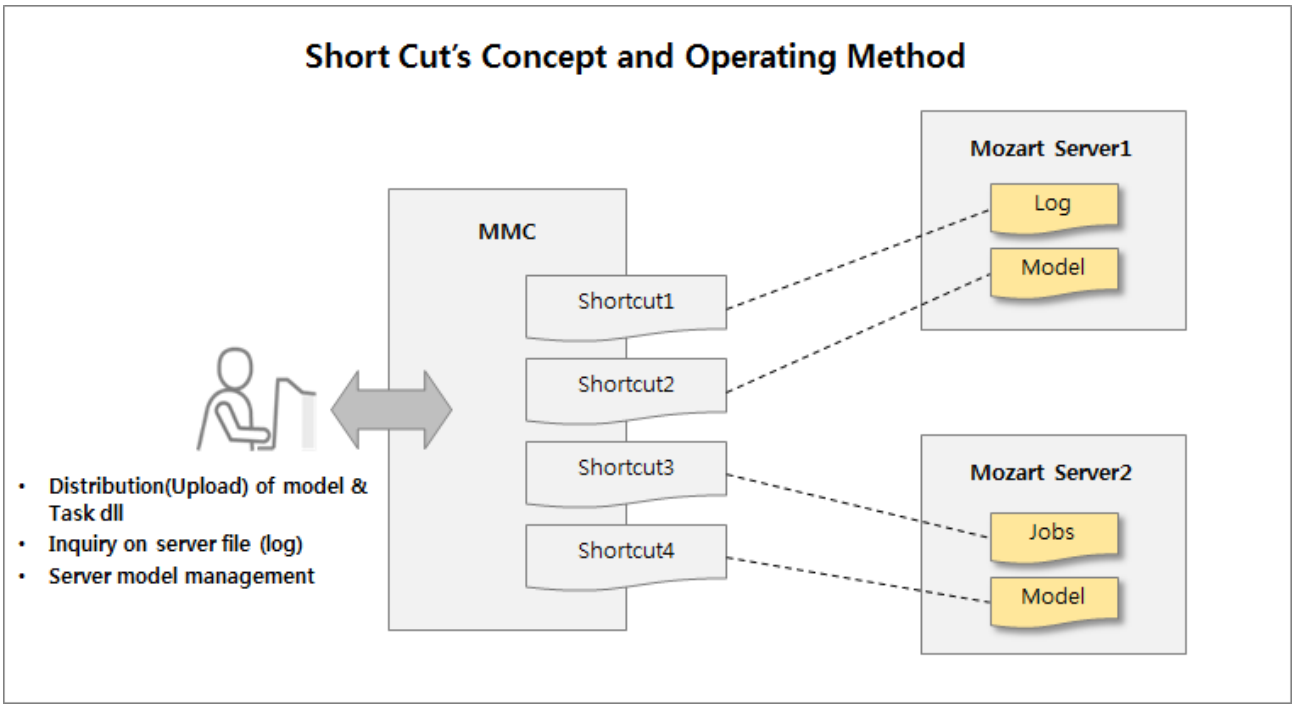
Failure Job : This is Job to be executed when Target job is failed to be executed. As same as other jobs, only the jobs registered by MMC can be selected.

Job Parameter : This is used to configure value of Parameter that decides how to execute Failure Job. Although Job is already configured with a default value, user can change this value for each Trigger.

Refer to **How to manage Trigger** to find more details on how to register/delete Trigger.

Shortcut

Shortcut acts a medium for file exchanges among MMC and server. MMC should be able to add/modify/delete a specific Directory and/or File as a management tool for MOZART Server. However, not all folders can be accessed due to security and management issues. MOZART's operating/management personnel can refer to only the path that is subordinate to Working Directory in MOZART Server through MMC. The following figure shows the concept of Shortcut.



MOZART Server operator can register Shortcuts mapped to main management folders of the servers registered to MMC's Server Explorer. The files can be uploaded and inquired through Shortcuts.

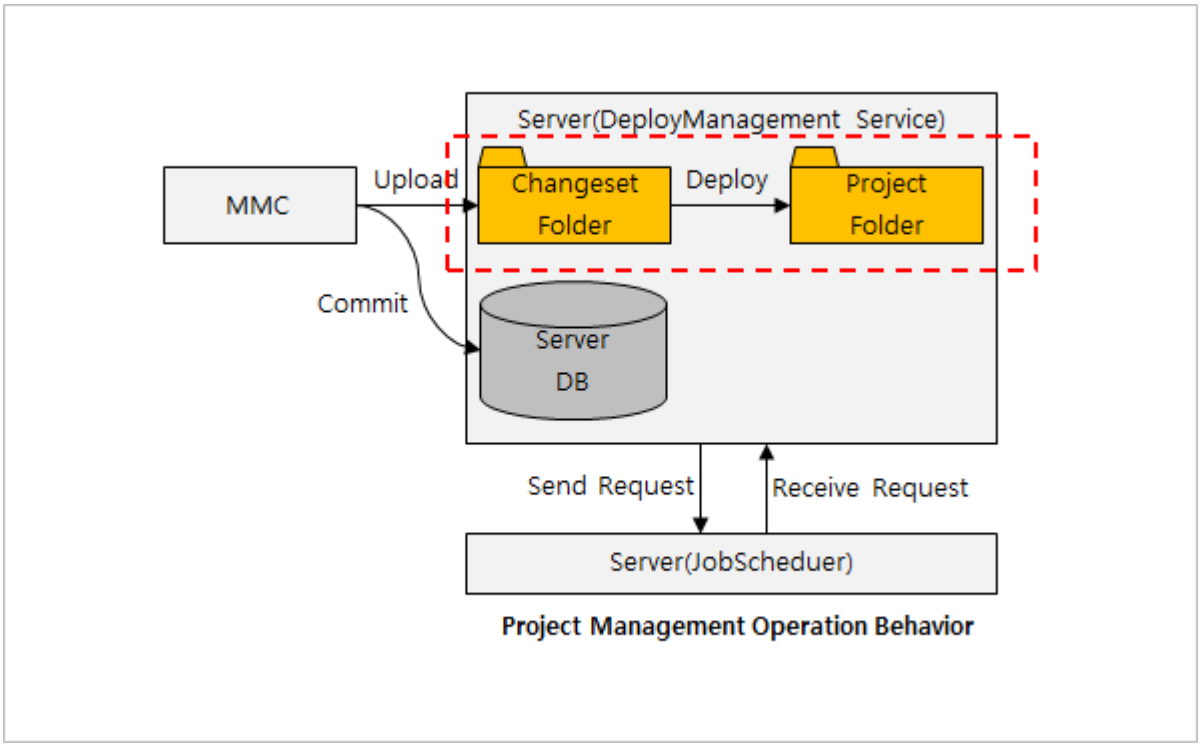
To see how to use Shortcut, refer to How to manage Shortcut.

Introduction of Project and Deploy Management

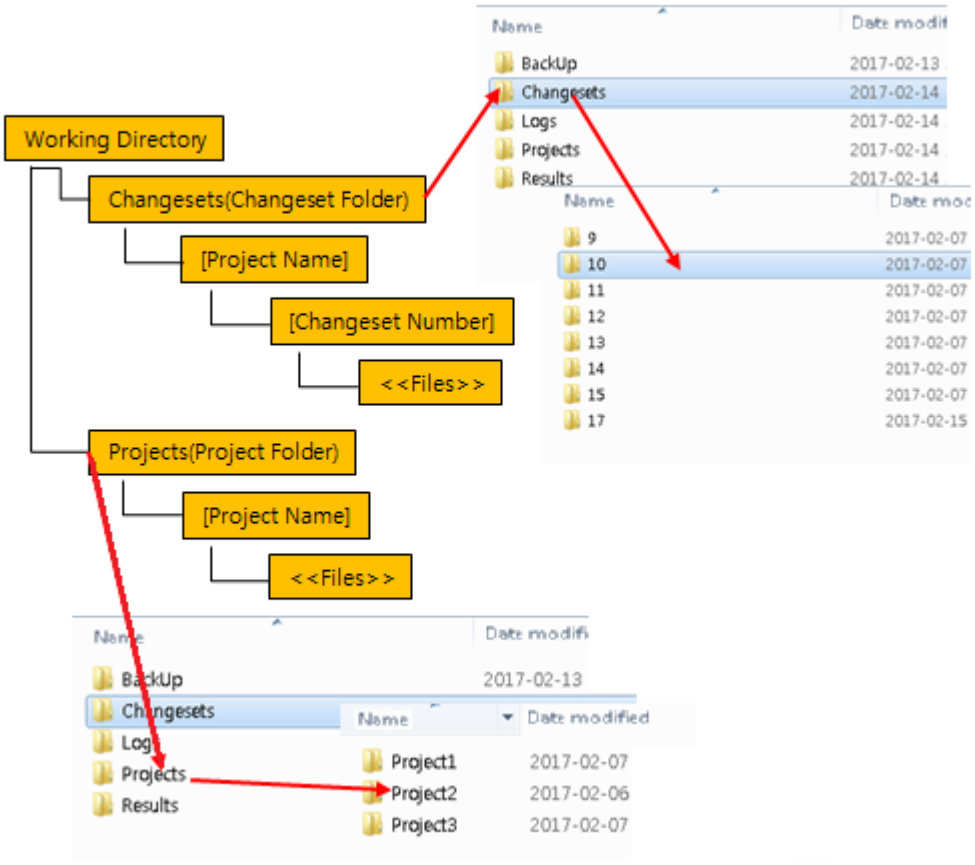
Processes and policies in manufacturing plants usually do not flow consistently from the initial stage, but keep changing over time. For responding these changes, it is essential to update Task dll and vmodel files registered in the server. Because servers refer to these files when executing tasks, unexpected errors may occur if the file is changed arbitrarily while the engine is running. Even if the Job / Trigger is activated according to the scheduled time or event and the end time can be expected, a human mistake can occur. In addition, from the viewpoint of the server management, it is difficult to manage the history and respond to urgent situations such as rollback unless the administrator care about the history because servers refer to the dll and vmodel files registered in the user-specified Working Directory.

To resolve above problems, MOZART Management Console (2.0) Client and Server products provides DeployManagement Service, which allows users to manage files on a project basis and schedule distributions by checking whether jobs / triggers are performed. The following descriptions are about the introduction of project management and scheduled deployment.

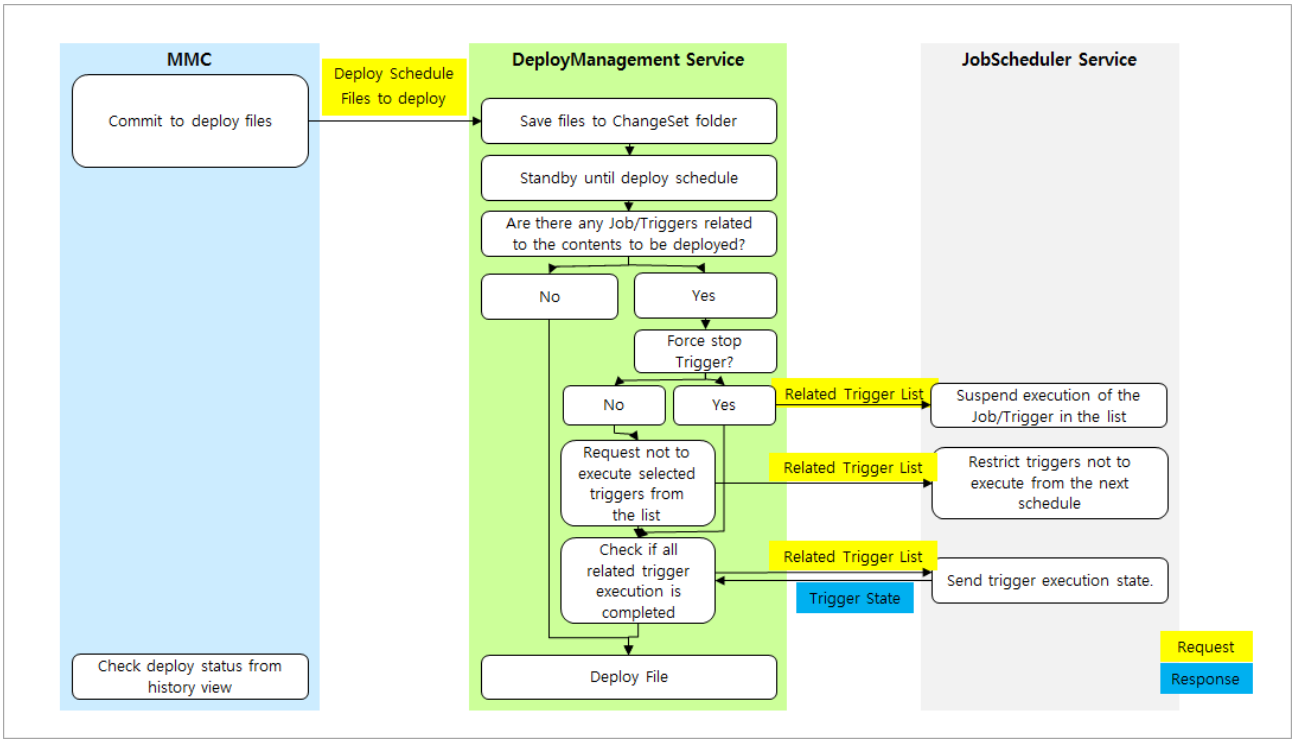
- **Project Management**
  Project is a management element that manages dll and vmodel files, the engine execution log, and the result files of the task by mapping Job/Trigger. A project is operated through Server's DeployManagement Service, and the information related to project is stored in DeployManagement DB(database) file. By saving the information in the database, the management of project is available and administrators don’t need to manually manage the version even if the change occurs in the project. The below figure is about the way to distribute files through projects.



When distributing files by MMC, files are not uploaded directly to the Projects folder. When users set up a distribution schedule for files and commit it, the information of changes and folder (Changeset) are created. The changeset information is saved in the DeployManagement.db file and files to be deployed are uploaded to the Changeset folder and waiting for the deployment time. At that time, triggers are checked by communicating between DeployManagement Service and JobScheduler. When triggers end, files stored in the Changeset folder are uploaded to the Projects folder. The Changeset folder is created when changes are made and it is managed by number.



- **File Distribution Scheduling**
  A job can have multiple triggers. If the distribution of dll and vmodel files is needed to the job, it is difficult to distribute all the files with the consideration of the end time of the triggers mapped to jobs by person. In addition, even if you know that all triggers are terminated, triggers can be executed at scheduled times during file distribution and it may cause unexpected accidents. To solve this problem, in MMC2, DeployManagement communicates between DeployManagement and JobScheduler Service. DeployManagement passes the list of files to be deployed to JobScheduler and JobScheduler checks the list and returns a list of Job / Triggers that use target files. DeployManagement asks JobScheduler to limit the execution of target triggers for file distribution. To prevent the distribution failure, the trigger starts at the scheduled time during the distribution process. JobScheduler restricts the execution of target triggers, and files waiting to be deployed are uploaded from Changeset to Projects folder when it is confirmed that target triggers are not executed in DeployManagement. After the file distribution is complete, DeployManagement asks JobScheduler to release the target trigger execution restriction.


MMC
Commit to deploy files
Check deploy status from history view
Deploy Schedule Files to deploy
DeployManagement Service
Save files to ChangeSet folder
Standby until deploy schedule
Are there any Job/Triggers related to the contents to be deployed?
No
Yes
Force stop Trigger?
No
Yes
Request not to execute selected triggers from the list
Check if all related trigger execution is completed
Deploy File
JobScheduler Service
Related Trigger List
Suspend execution of the Job/Trigger in the list
Related Trigger List
Restrict triggers not to execute from the next schedule
Related Trigger List
Send trigger execution state.
Trigger State
Request
Response

Server & Client Settings

Server & Client Setting Overview

There are 6 services that are operated by MOZART server and these can be modified through server setting.

- **Job Scheduler Service** : This is server's Main Service that executes Job according to Trigger information registered in Server.
- **DeployAgent Service** : This is a service to exchange files. Transfering(upload/download) files among server and MMC is done through this service.
- **OutFile Service** : This service provides MOZART Studio with list of compressed files of MOZART JOB execution results and to download the files to the studio.
- **License Service** : This is a service that MOZART Server issues license automatically to MOZART Studio used by general users (Note : **Local License Service Method**)
- **DeployManagement Service :** This service manages the changeset history of Job, Trigger, file distribution and Project of MOZART Server.
- **TriggerJob Service** : It is a service to let users execute triggers from other than MOZART Management Console. (i.e. Web application).

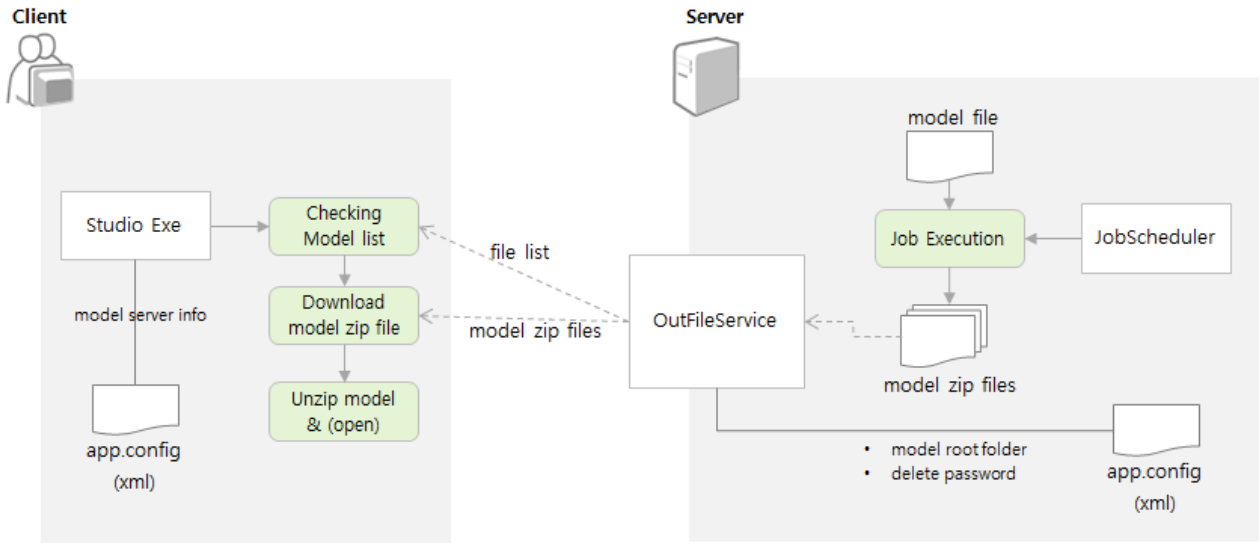
When MOZART server is installed, all of the services above are executed. Only Job Scheduler is executed as an independent Instance and other services are executed in the same instance(Server Service). Configuration in Server and client for each service, refer to the corresponding content's pages respectively.

1. **Configuring Model Download** : This explains about Server and Client(MOZART Studio) configuration methods for downloading a Model from server.
2. **Configuring AutoUpdate** : This explains how to update Client(MOZART Studio - Site's specific Studio (Purchased one)).

Refer to **Server Installation Manual** for information of Server installation.

Model Download Setting

This section explains the configuration steps to inquire and download the Job result files executed from MOZART Server to MOZART Studio.



Server Configuration

In order to configure Model Download in MOZART Server, the **root** folder that saves the compressed Model files needs to be designated, permission granted to **delete the Model file** from studio and **password to delete the file** needs to be set.

- Designate root folder where Download Model in Server is saved :**[Designating Root Folder]** Include the following lines in MozartServiceHost.exe.config file(located in the folder where MOZART Server is installed) to designate root folder.

  1) Configure a Root that a Model executed automatically by JobScheduler is saved in

  Config Section : <appSettings>
  Key : app-output-dir
  Configuration example :

```
<appSettings>
   <add key="app-output-dir" value="D:\MOZARTServer\Models"/>
</appSettings>
```

2) Configure a Root that a Model executed manually by developer or operator is saved.

Config Section : <appSettings>
Key : web-output-dir
Configuration example :

```
<appSettings>
   <add key="web-output-dir" value="D:\MOZARTServer\ModelsManual"/>
</appSettings>
```

- Configure where a Model in Server can be deleted from a client or not and set a password if a Model is deleted.

  The compressed Model file in the server can be inquired through **[File>Download Data From Server]** menu in MOZART Studio. The file can be deleted from the client and a password is required to delete the files. The password can be set through MozartServiceHost.exe.config file by including the following lines. The Key and Section could be set through here.

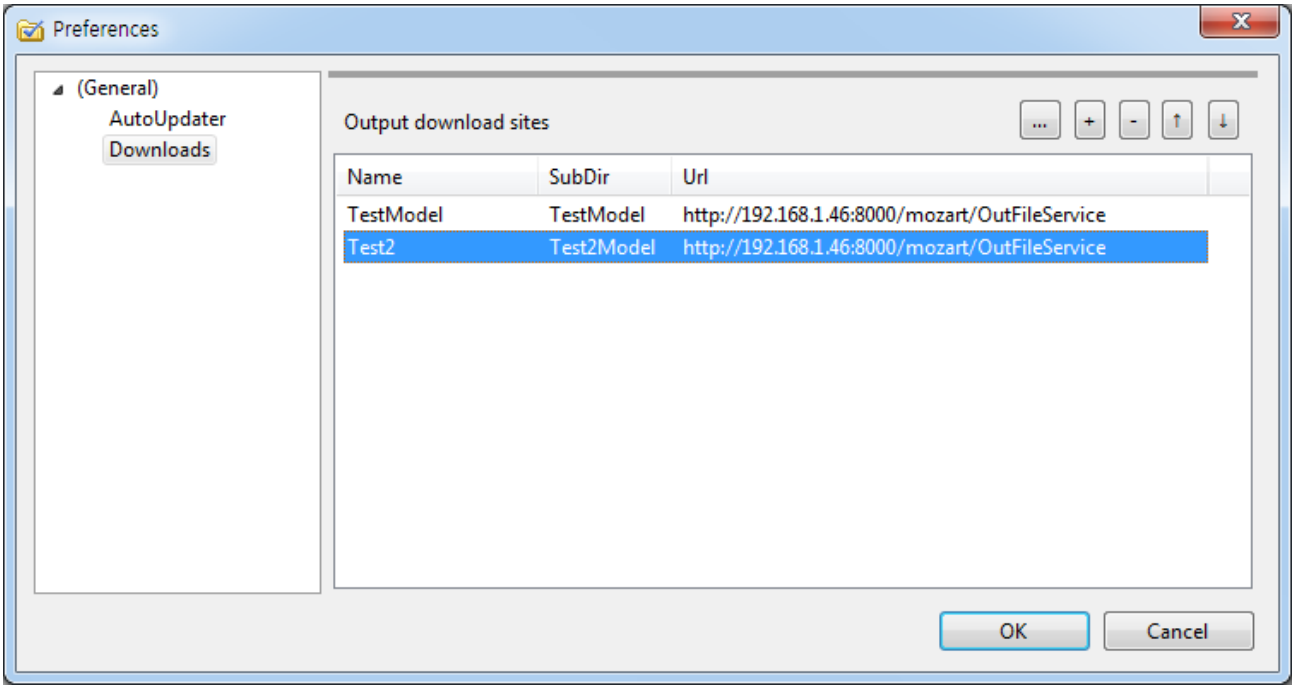
  Config Section : <appSetting>
  Key : password
  Configuration example :

```
<appSettings>
   <add key="password" value="MOZART"/>
</appSettings>
```

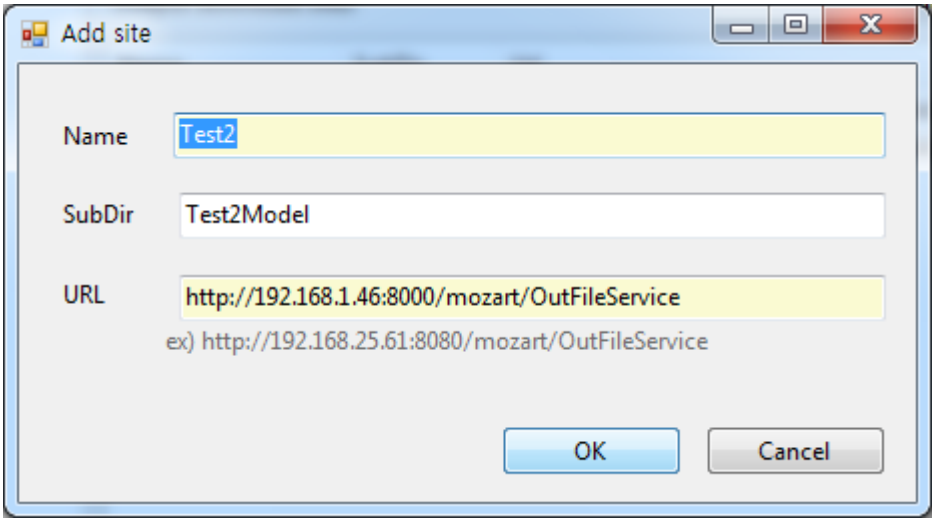
** If "password" is not set, the compressed Model cannot be deleted from client.*

Client Configuration

Client should designate a Server that Model is downloaded from and the folder for each Model that is saved in the corresponding server. In order to configure this, first execute **OOO_Studio** that is purchased by each site and use **Tool>Options** menu.



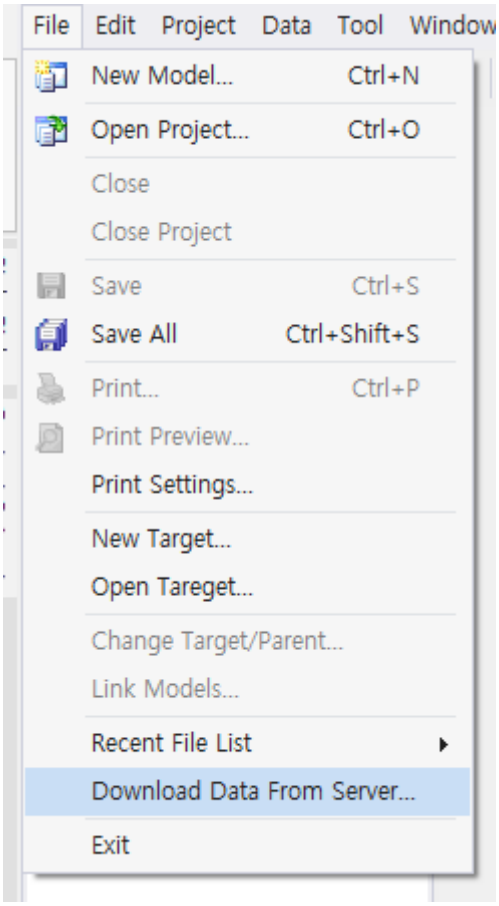
- Select Downloads from the Tree at the left side.
- Add Download site to the list at the right side. Press [+] button at the top to add the site.



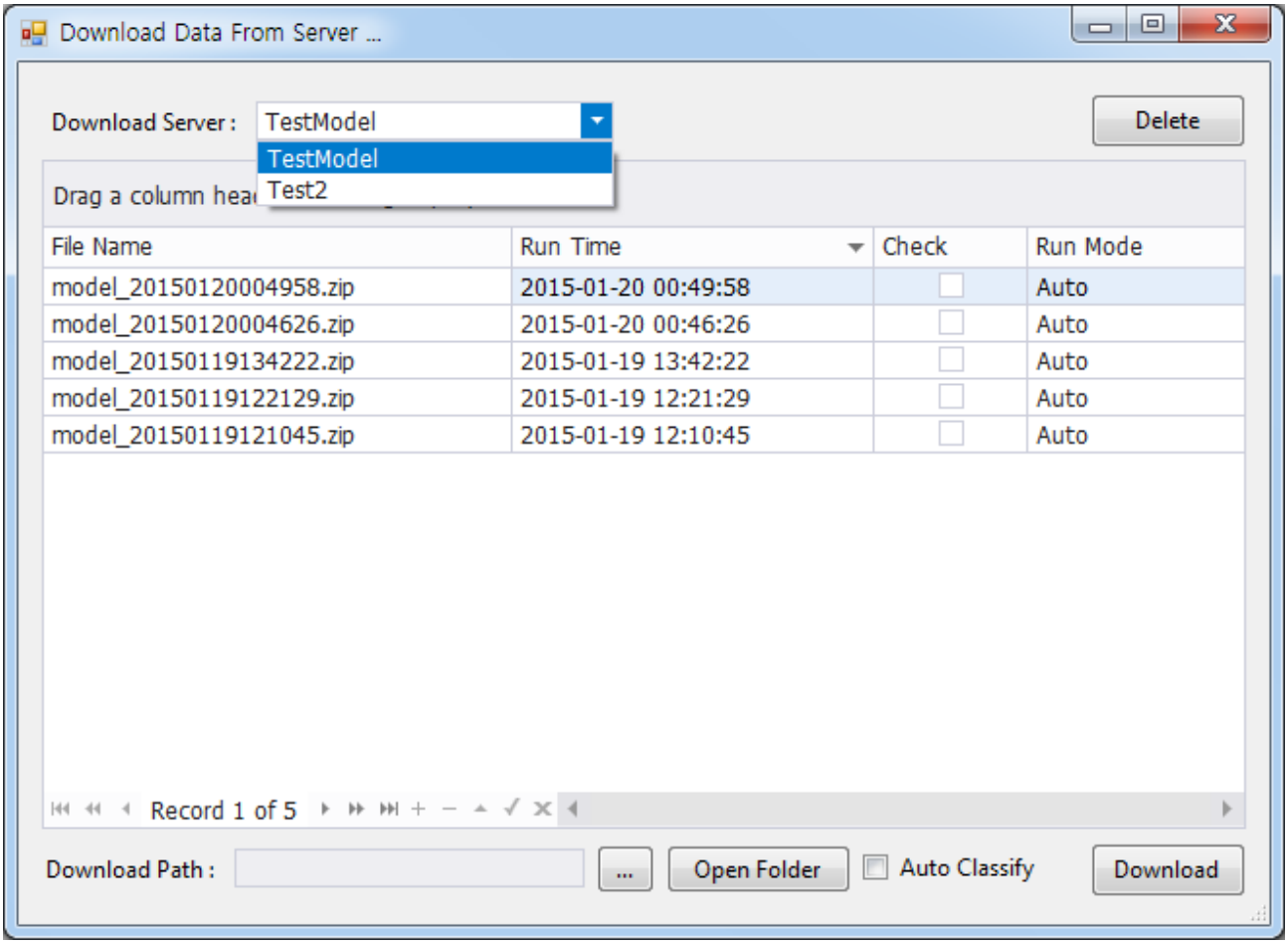
- **Name** : This is a site name that is displayed in a combo box when a Model is downloaded.

- **SubDir** : This is a name of a folder where Models are saved in Server. This is configured as a relative path with respect to Model download base folder that is configured in the Server.
- **URL** : This is a Service URL for Server that provides Model file download service. The format is same as the example above. Input URL that is confirmed by user site's operating team. Generally Port and IP configuration can be different. This should be checked with the operating manager after setting the server.
- Multiple sites can be registered and the display order in Download window can be adjusted through the arrows at the top.
- In order to modify information of the registered site, double-click or use **[...]** button at the top.
- In order to delete any added object, use **[-]** button.

Like the above example, multiple folders in a single server can be registered or multiple servers can be registered. After configuring like above, Model can be downloaded by using Model search window through **[File >Download Data From Server]** menu from Studio.

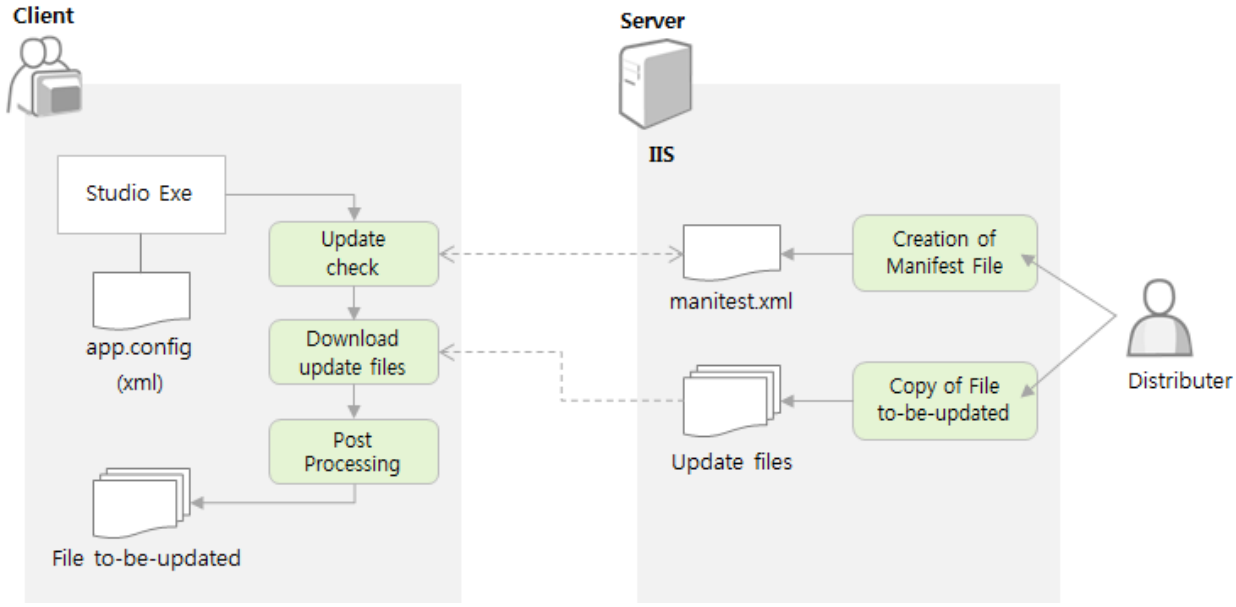


When the menu is executed, a list of every registered server name is displayed in Download Server combo box like the following figure. Then, select a Model file and press download button in order to download a specific Model from a list of Models in the selected server.



AutoUpdate Setting

To update the client version automatically, the update files should be compressed and uploaded to the designated user group specified server and the client should have the update server connection information. The client requests the server for any updates and if the update exists the client will be updated. The update procedure is seen through the following figure.



In a company level, there could be a server machine already existing to distribute updates. Whether using an existing server or a new server for Auto Update, the server should have IIS installed. The following explains how to configure Auto Update server.

[Server Requriements]

- .NET Framework 4.0 or above
- IIS (Internet Information Server) version 6 or above

[How to configure a Server]

1. Designate a folder where target files for update are saved.

2. Execute "IIS(Internet Information Service) Manager".

3. Add an Application Program Pool from **[Application Program Pool -> Add Application Program Pool]** menu. Set .Net version to 4.0 (The name of application program can be defined as you wish. EX) MOZARTUpdateServer)

4. Add an Application Program from **[Site -> Default Web Site -> Add Application Program]** menu. Input value can be set as below.

- **Alias** : Input an alias for the application program to be registered. **Alias is the name required when a Server URI is entered in client. When Download URL value is configured in client, an input format like "host address/[alias]/manifests.xml" is used.**
- **Application program pool** : Add Application Program Pool by clicking **[Select]** that was included from Step 3.
- **Actual path** : Designate a folder where target files for update is saved as explained in Step 1.

5. Edit Mainfest through MainfestEditor. The file is located in [Update]r where MOZART Client installed. Please refer to **How to edit Manifest file**, to find more details how to edit Mainfest.

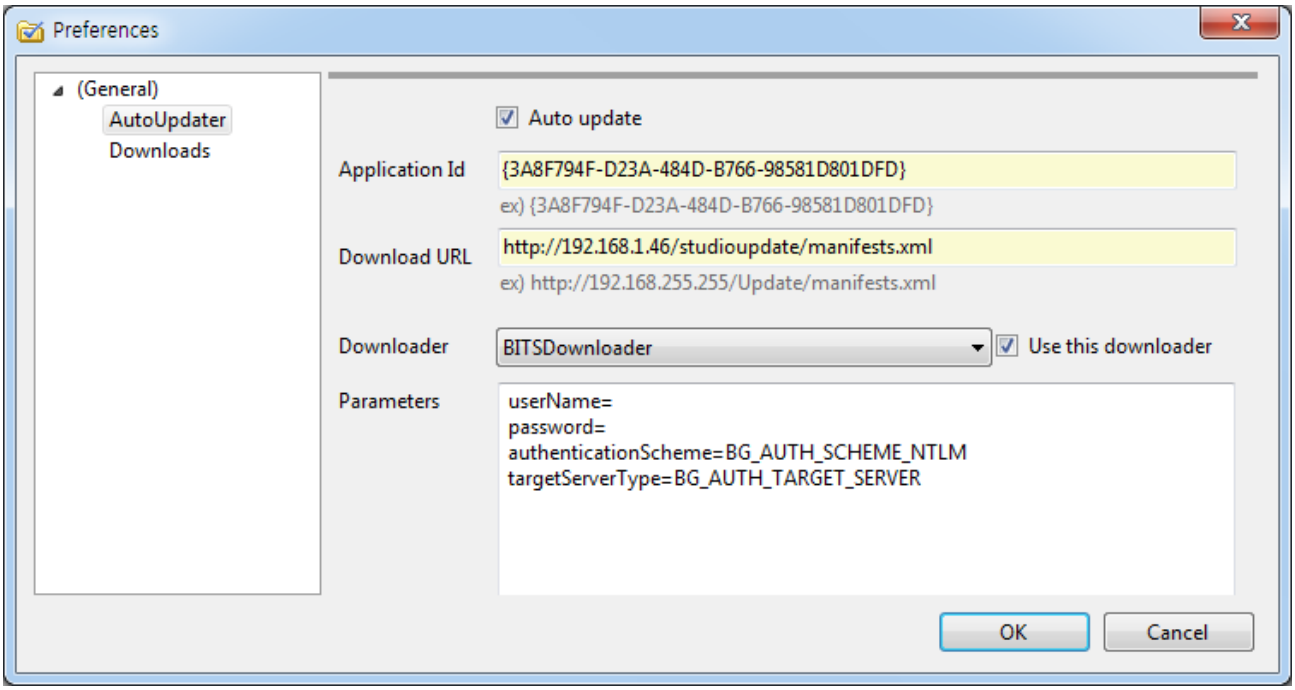
6. An xml file will be generated. Copy the generated xml file to the update target folder. When this is done the setting on server side is completed.

The port used from Default Web Site should be opened. In general, the port number is 80 but it could be blocked according to the server setting. Error may occur when the port is blocked so make sure to check the port setting during server configuration.

[How to configure client]

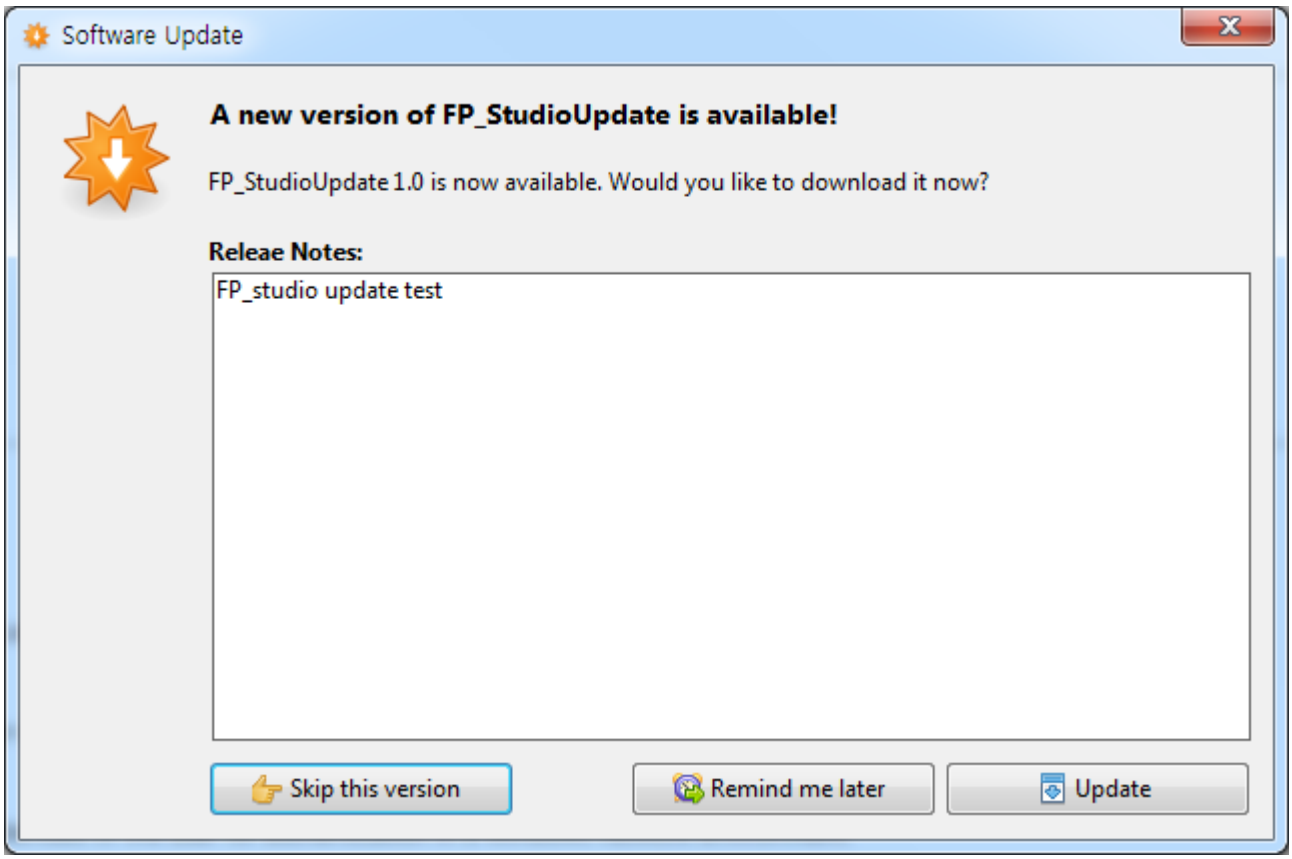
1. On the client side, AutoUpdate and Update Server can be configured through **[Tool>Options]** menu in the Studio.

2. Each configuration item can be configured as below.



- **Auto update** : If checked, auto update is automatically activates according to the following input information. If not checked, auto update is deactivated.
- **Application Id** : Unique ID of Studio program. ***User should not modify this.*** When this is compared with Server's manifests file, only update information for target Application is compared.
- **Download URL** : Update Server's URL. The format should follow as below.
  + format: http://[SeverIP]/ [Alias of Application program used when Server is configured]/manifests.xml
  + [ServerIP], [Alias of Application program used when Server is configured] are required to be edited. Configure the corresponding part after checking it with Client UI Development/Operating organization.
- **Downloader** : Select a Downloader. Default is BITDownloader.
- **Parameters** : Parameter used for Server authentication. This part does not need to be modified.

3. When Studio is restarted after items are configured, the following download window is activated.

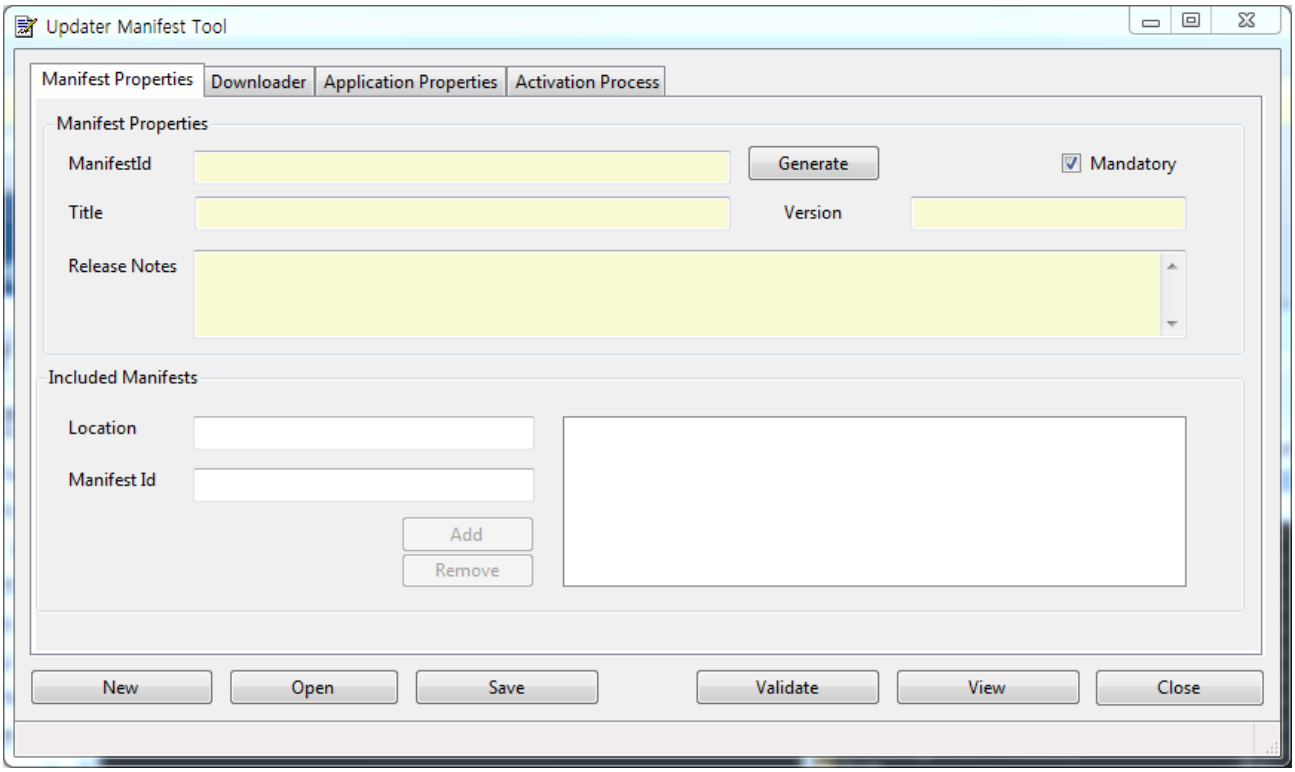


- **Skip this version** : Skips to check for any updates on the next start.
- **Remind me later** : Asks to update the version on the next start.
- **Update** : Download, updates the version and restarts the Studio.

Manifest Editor

Manifest Editor is an editor for creating/editing Manifest file. When MOZART Client is installed, ManifestEditor is also installed in Updater folder subordinate to MOZART's installation path.

1. Run ManifestEditor.exe from the folder where server execution file is located. MainfestEditor consists of four tabs as shown below.



2. Fill in the information through Mainfest Properties Tab.

- **ManifestId** : This has the same ID as the distribution ID that is changed whenever a new update file is distributed. Client discerns whether AutoUpdate should be executed or not by comparing the corresponding ManifestId's value. New GUID can be created through **[Generate]** button at the right side. It should be changed during each distribution. **(※xml document key = manifestId)**
- **Title** : Name of the corresponding Manifest file **(※xml document key = title)**
- **Version** : Version of the distributed product. When it is distributed, update is executed according to its rule. **(※xml document key = version)**

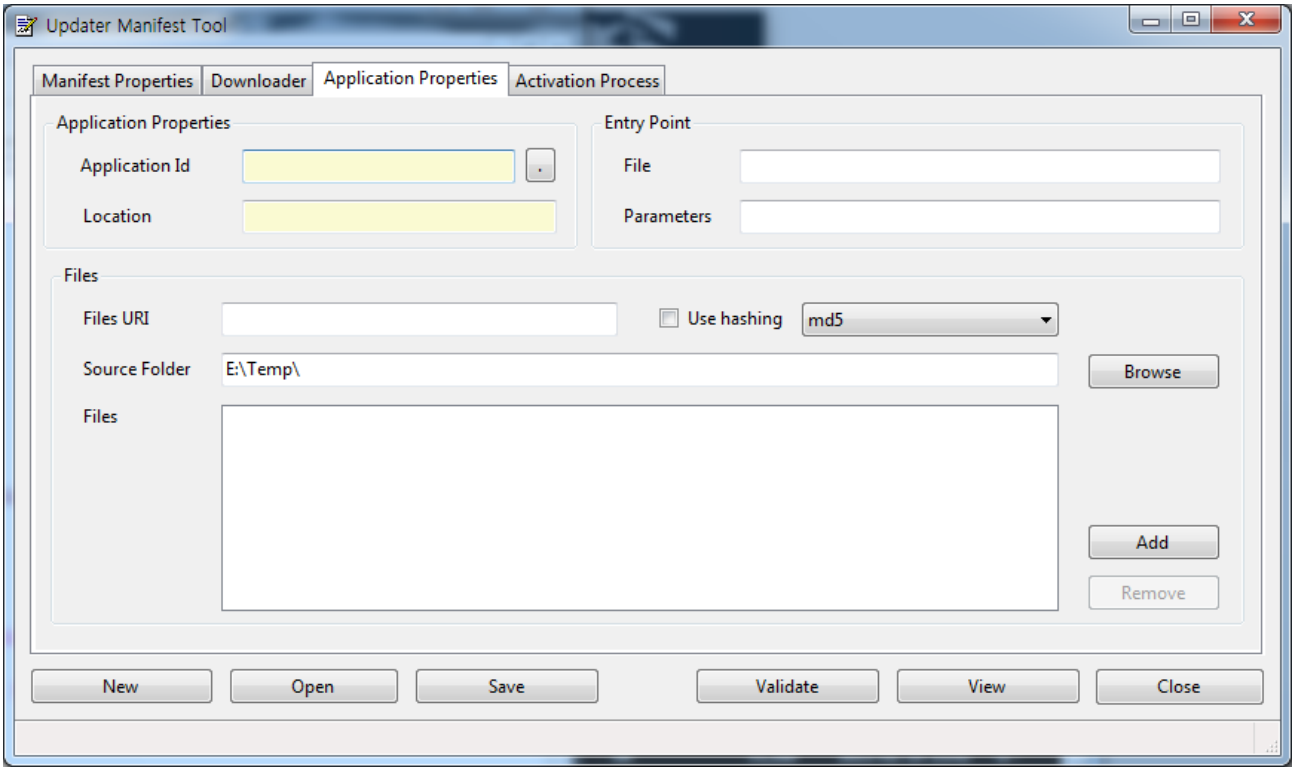
- **Release Note** : Brief notifications about the fixes in the distributed version. This is updated according to the distributed contents. **(※xml document key = description)**

Example of Manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
manifestId="{AF7A3BD5-10A1-4155-BBF8-631906D86DAE}" mandatory="False" xmln
  <title>FP Studio</title>
  <version>1.0.3</version>
  <description>first release</description>
  <application applicationId="{3A8F794F-D23A-484D-B766-98581D801DFD}">
    <entryPoint file="FP_Studio.exe" parameters="" />
    <location>.</location>
  </application>
  <files base="http://xxx.xxx.xxx.xxx/MOZARTUpdate" hashComparison="No">
    <file source="update-files.zip" transient="No" />
  </files>
  <activation>
    <tasks>
      <task type="MOZART.AutoUpdater.ActivationProcessors.WaitForApplicati
      <task type="MOZART.AutoUpdater.ActivationProcessors.ApplicationDeplo
    </tasks>
  </activation>
</manifest>
```

The above example can be found from **[MOZART Client folder>Updater>Server]** in **manifests.xml** file.

3. Fill in the information through Application Properties tab.
This configures information of applications to be updated. The mandatory configuration items are seen below.



Application Properties

- **ApplicationId** : This is a GUID of main update program and uses GUID in App.config file of the corresponding target file. Two Guid should be always configured with the same value. **(※xml document key = application/applicationId)**
- **Location** : The location where the downloaded file is saved. **(※xml document key = application/location)**

Entry Point

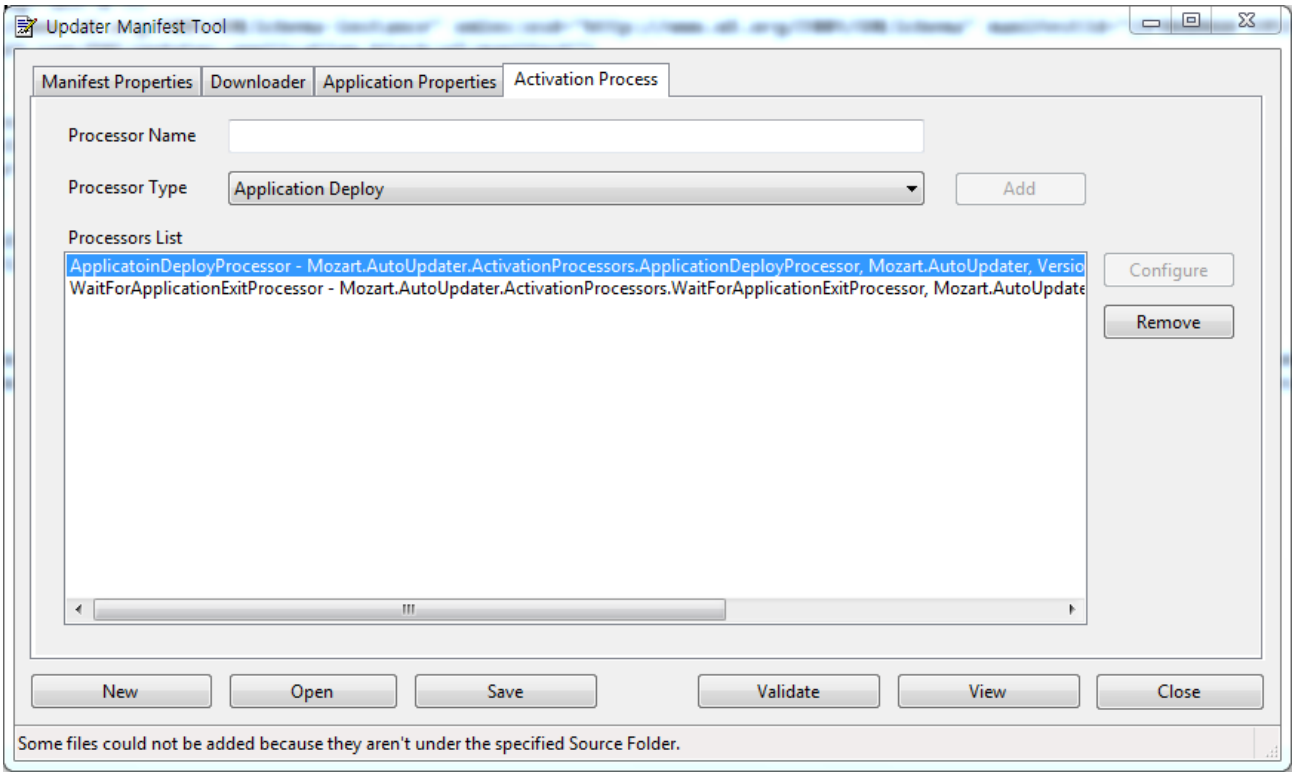
- **File** : Name of execution file. For instance, if update for FP_Studio is configured, the name should be set as 'FP_Studio.exe'**. (※xml document key = application/entrypoint file)**
- **Parameters** : Parameter configured at execution. Seperate parameter is not necessary for Studio update.

Files

- **Files URI** : To set URI where downloaded files are located. In general, the path set in local host is used for the server. **(※xml document key = files/base)**
- **Source Folder** : A local folder where the files are stored is selected.

- **Files** : The section to input files to download from local folder. In normal cases, the compressed updated file is selected. **(※ xml document key = files/source)**

4. Fill in the information through Activation Process Tab
Select the processor to be used for update. Multiple processors can be selected.



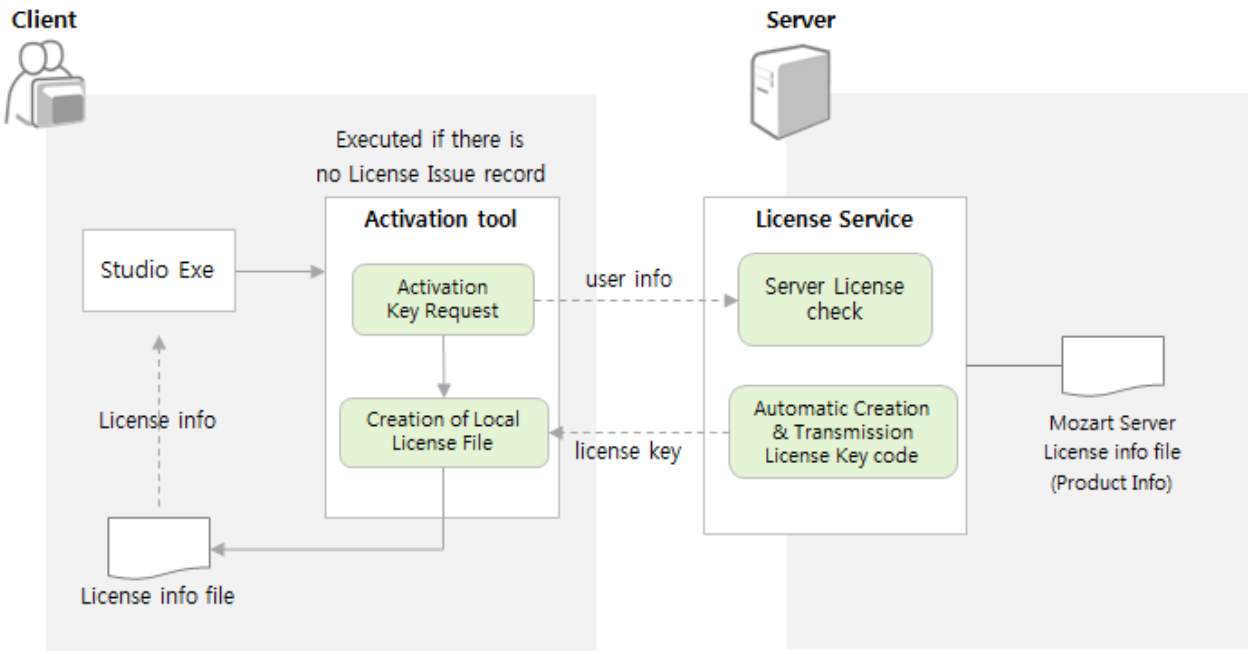
Enter Processor Name and select a Processor Type. Then, add the Processor by clicking **[Add]** button. Processor Types can be seen below. Several processes can be registered together. The processors for auto update of MOZART Studio is Application Deploy and Wait For Exit. Refer to manifests.xml file that is distributed together in server installation folder. **(※xml document key = activation/tasks/task)**

- **Application Deploy** : Copies downloaded files into target folder for update. In case of compressed file, it will decompress the file after copying is complete.
- **File Copy** : Copy a specific file to a specific location described in config file.
- **File Delete** : Delete files with a specific format in a specific location described in config file.
- **Folder Copy** : Copy a specific folder to a specific location described in config file.
- **Folder Delete** : Delete a specific folder describe in config file.
- **GAC Util** : Manage GAC as described in config file using GACUtil. (i.e. registration/delete/etc.)

- **Hash Validation** : Compares hash code of a downloaded file described in config file with the source file in the server.
- **Install Util** : Manage Service through InstallUtil.
- **MSI** : Install/delete/patch package configured in config file.
- **Start Application** : Restart application after file is downloaded and updated.
- **Uncompress** : Decompress a specific compressed file to a specific location described in config file.
- **Wait For Exit** : Close client for update and standby until update is completed.

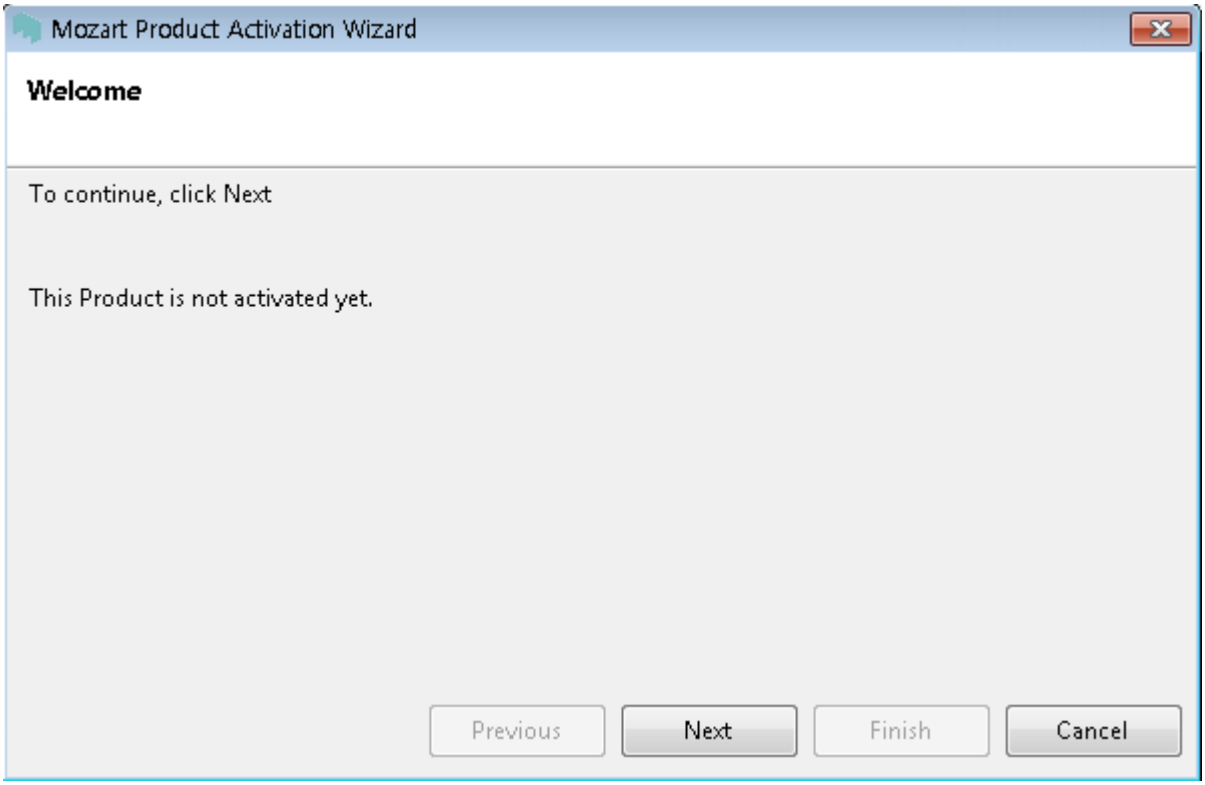
Local License Service Concepts

Local License Service is an authentification service that is provided by MOZART Server. When the license of MOZART Server is authentcated, the server automatically distributes authentication keys to user PC using MOZART Studio.

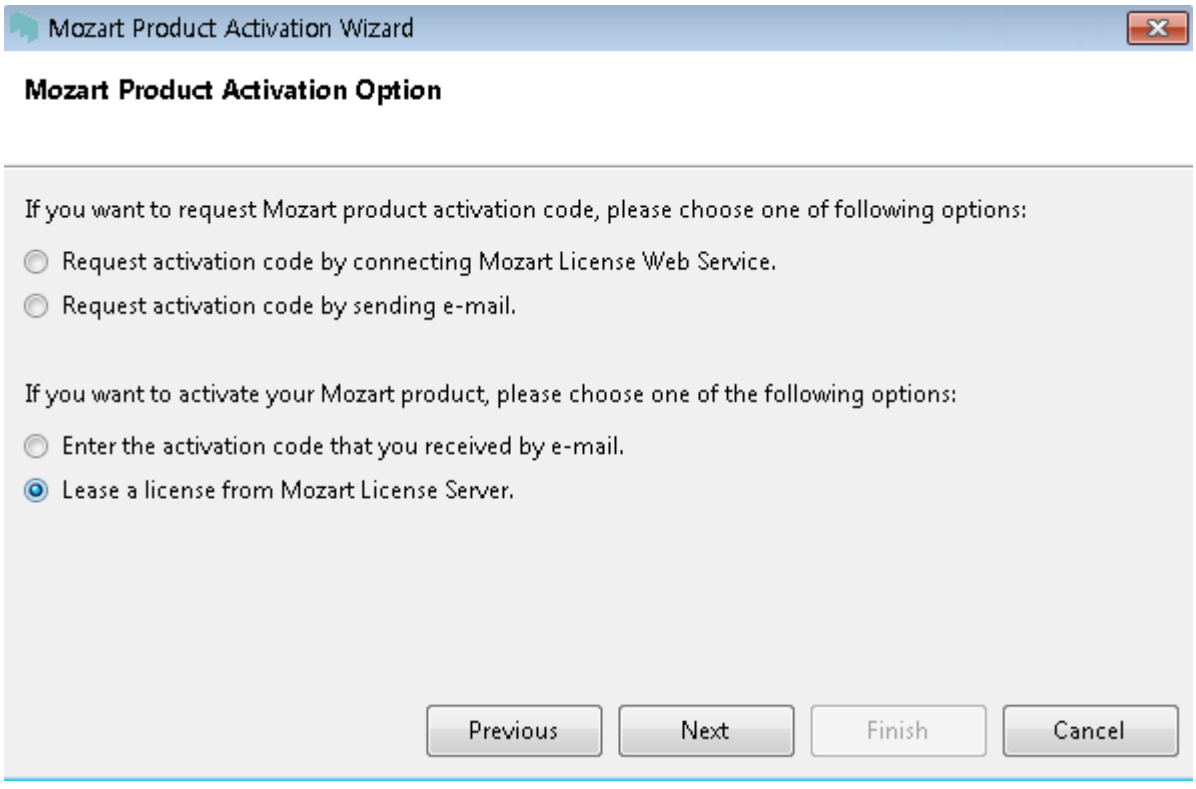


As seen above, the server providing license service should obtain MOZART Server License in advance. Then, newly installed MOZART Studio should get a license authentification through Activation Tool when MOZART Studio is executed without an issued license information. At this moment, license can be issued from Local License Sever by the following procedure.

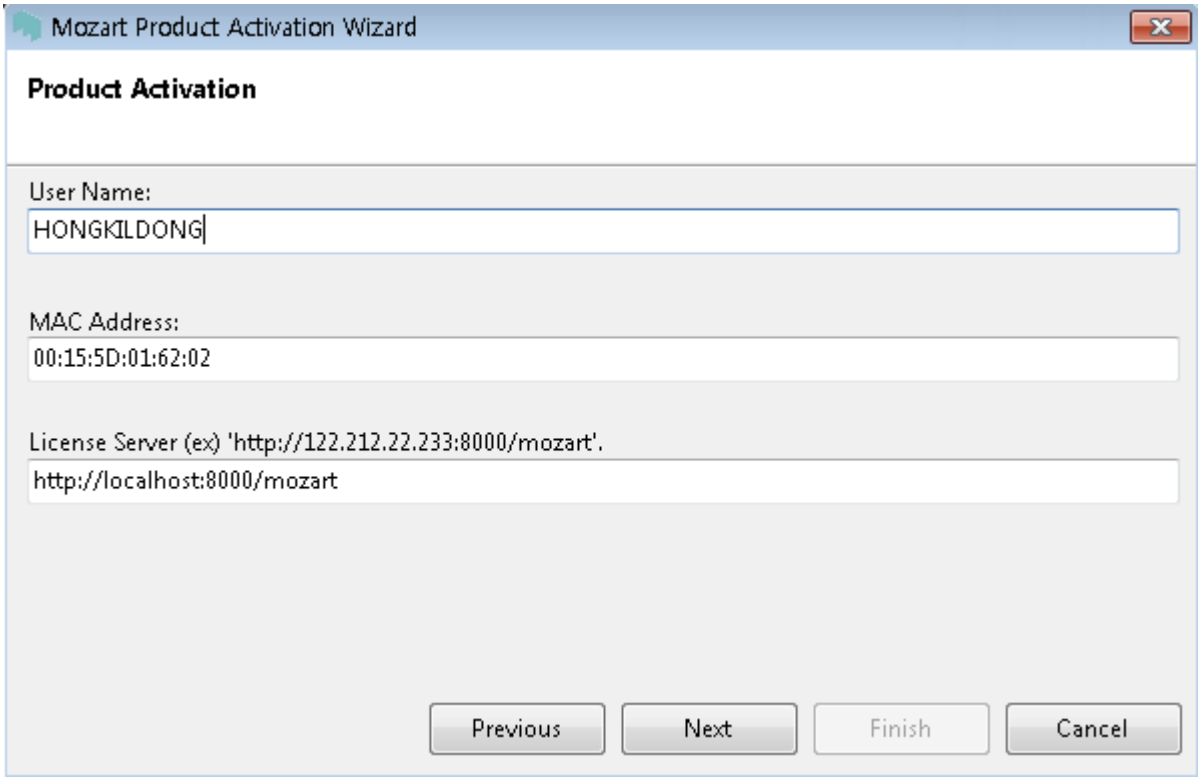
1. If there is no license, Activation tool shall activate. Click "Next" button to proceed.



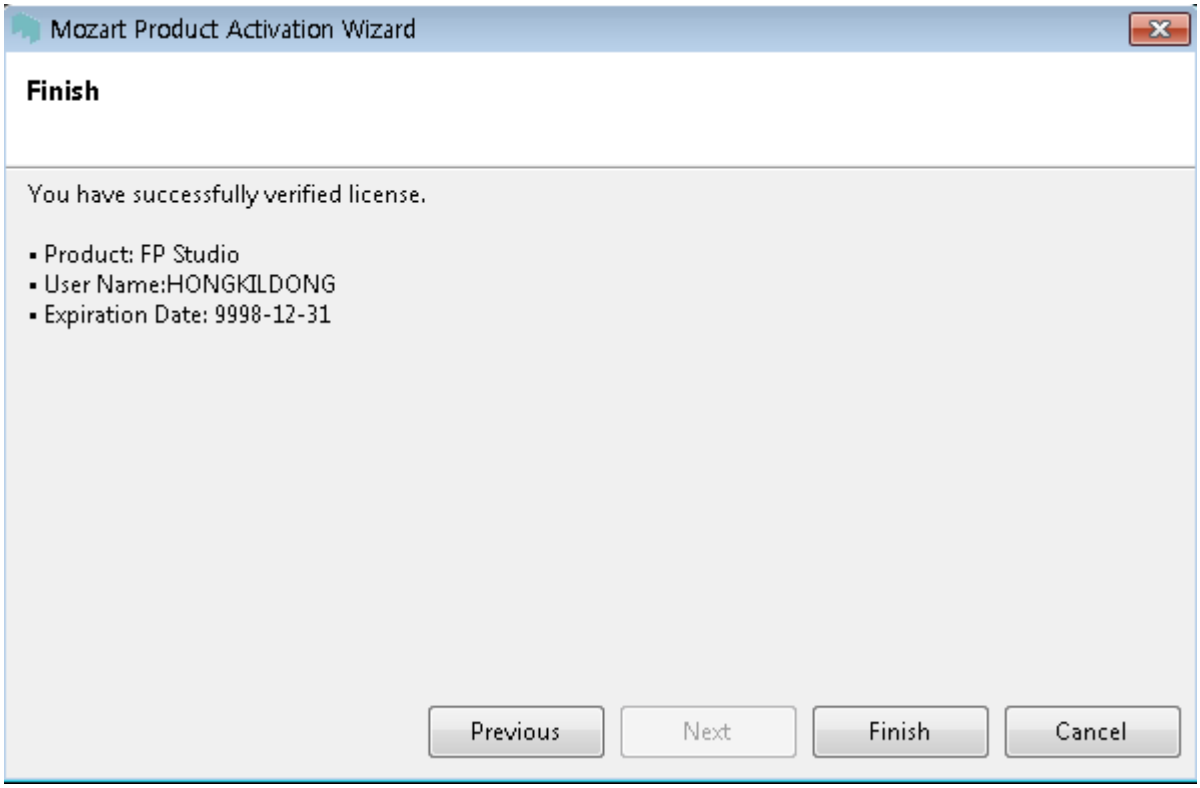
2. After setting Activation Option to "Lease a license from MOZART License Server", click "Next" button to proceed to the next.



3. Input user name and IP address of the server that has the license service.



4. If the license is issued properly, the following confirmation message window should be displayed. Otherwise, check if the server connection information is correct or if the service is working fin



The license service is registered to Window Service when MOZART Server is installed. If license service is activated for the first time through **[Start menu->Start Server Service]**, it is set to start the service automatically from the next reboot. Therefore, the administrator(or operator) does not need to perform additional settings once the service is started.

MMC User Guide

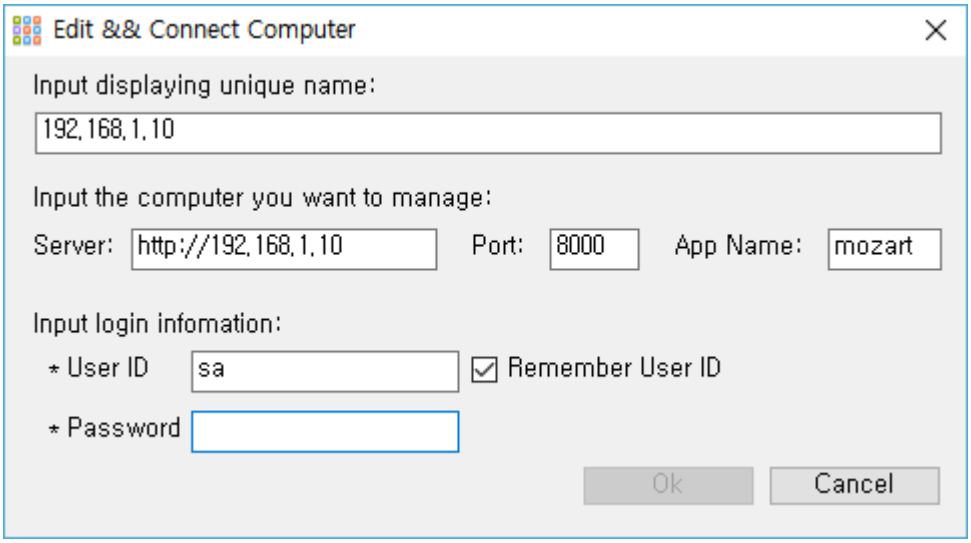
Server Management

MMC enables the job management on multiple servers. To register the target server of MMC, MOZART Server should be installed in the corresponding server. You can use **[Add Server Connection]** button in Server Explorer to add a target MOZART Server.

Adding Server Connection

1. Click **[Add Server Connection]** icon( ) from the Sever Explorer tool bar.

2. Type in the information to add the server connection.



Input name : Enter the name that the console manager uses to manage target server.

Input the computer : Enter the URL of the corresponding server. The format should follow the example shown above but, using the actual IP address.

User ID: Enter the user ID to connect to the MOZART Server. (The default administrator account is **sa**.)

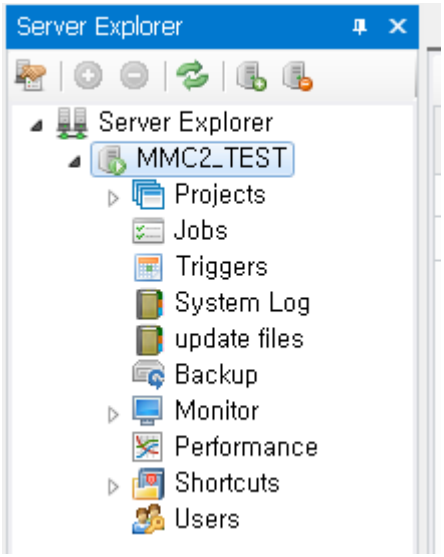
Password : Password: Enter the password required to access the MOZART Server. (The initial password of **sa** account is "**mozart**".)

> When the MOZART Server is installed for the first time, the default administrator account and password are included in the Deployment.db file. When accessing the database for the first time, you can access it with the default account and password. The default administrator account / password is as follows.

- **User ID** : sa
- **Password** : mozart

3. Click **[OK]** after entering all the information.

4. If a server appears with a name through Server Explorer, the server has been created successfully and ends the server registration.

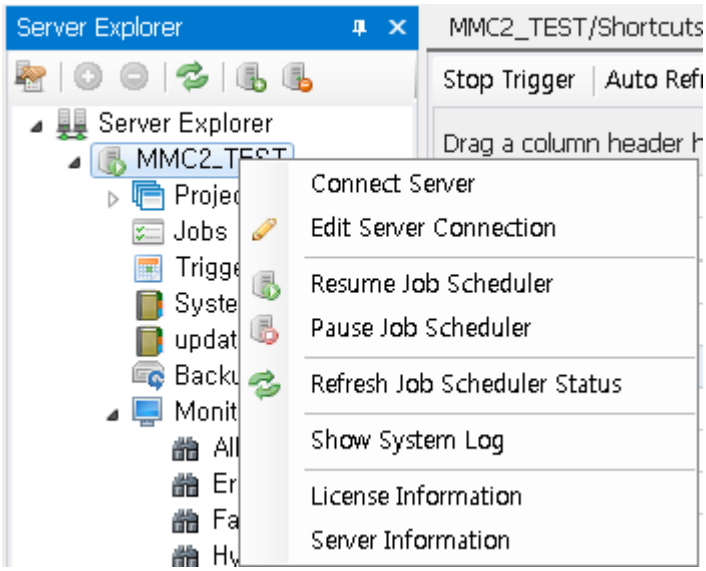


How to Check Server Information

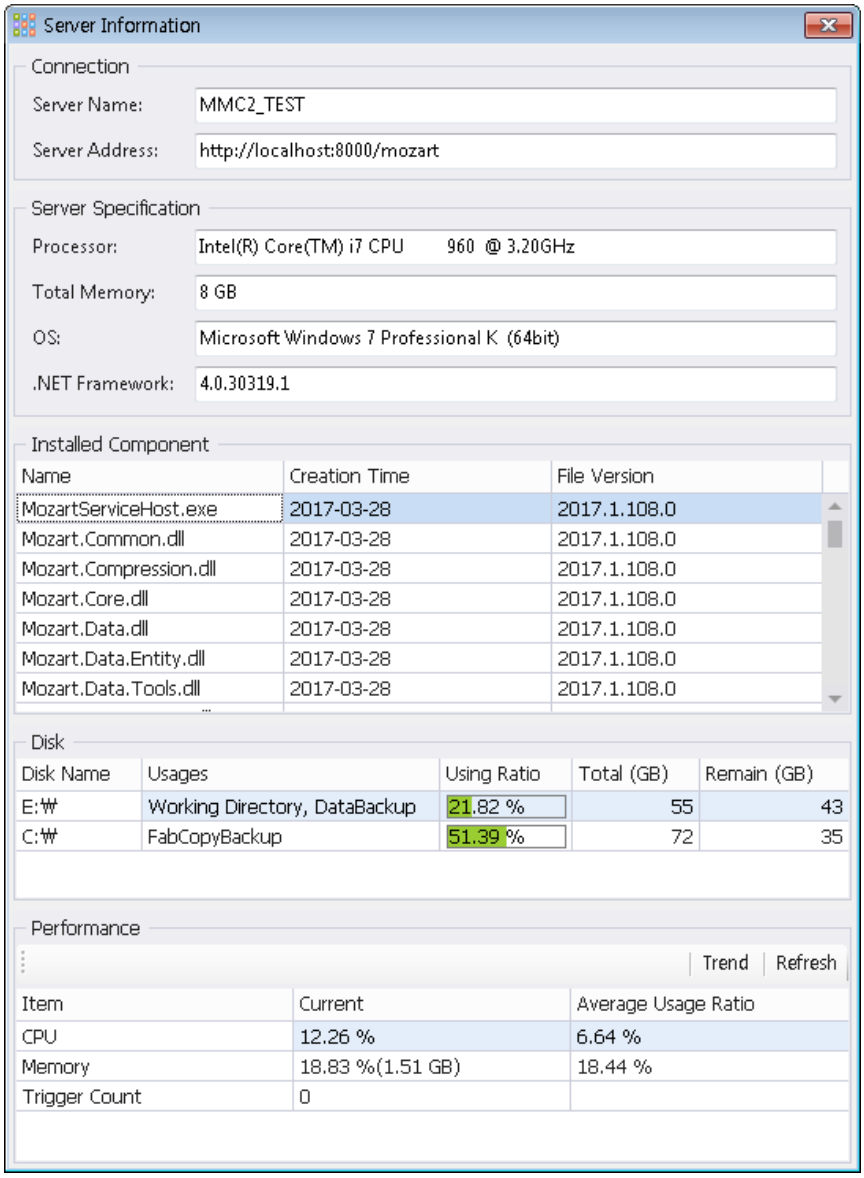
You can check the detailed information of server registered in MMC Server Explorer. In addition to the basic information such as the server name and IP address, you also can check H/W specification of Server machine, MOZART dll version installed on Server, WorkingDirectory and Backup, and HDD capacity status. The below explanation is about how to access Server information in the MMC Server Explorer.

How to check server information

1. In the Server Explorer, right-click the target server for which you want to know the server information and click **[Server Information]** button.



2. Check the server information in the Server Information dialog.

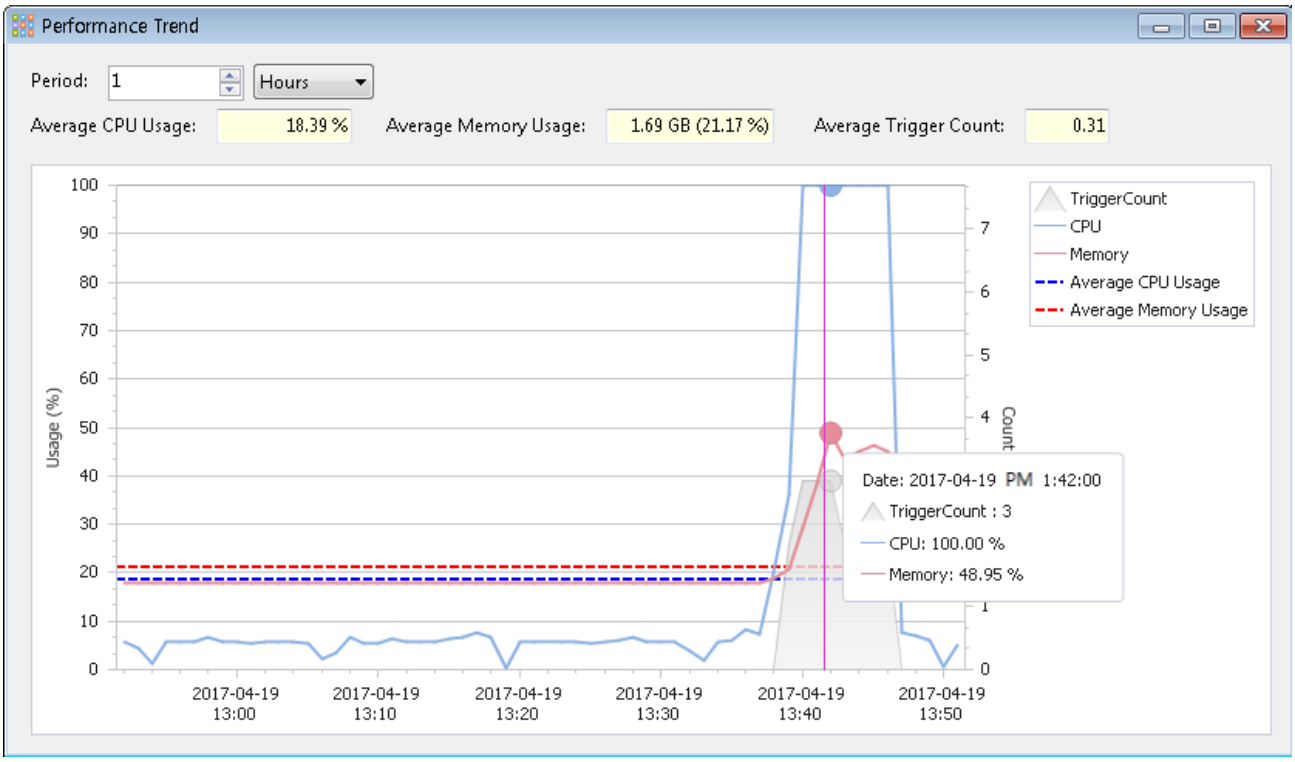


- **Connection** : This section is used to check the connection information of the target server
  - **Server Name** : Server name information entered when registering this server in Server Explorer
  - **Server Address** : The IP address information entered when registering this server in Server Explorer
- **Server Specification** : Displaying H/W specification, OS and .NET Framework version of the registered server.
- **Installed Component** : You can check the dll version of MOZART installed on the target server.
- **Disk** : Displaying the HDD capacity information of the target server. Disk only displays the information of HDD where WorkingDirectory and Backup are located. (Display unit: GB)

- **Performance** : Displaying the current CPU and memory usage of the target server and the number of triggers being executed.
  - **Trend** : In addition to the current usage, you can check the CPU and Memory usage status of the target server. When you click the Trend button, the Performance Trend Dialog pops up.
  - **Refresh** : Refreshing the performance information with the latest one.

How to view Performance Trend

If you click [Trend] button in the Performance area at the below part of the Server Information Dialog, c the Performance Trend Dialog pops up. You can check the overall server resource usage and trigger execution status for the specific period based on the current time through the Performance Trend Dialog. The following image is an example of MMC's Performance Trend Dialog.



- **Period** : Setting the period. The period can be set in hours/days and the server status will be drawn before the entered period based on the current time.
- **Average** : Displays the average CPU / Memory utilization and Trigger execution count correspondent to the configure period.

When the user sets a specific period, the server status is displayed as a graph. The dotted line in the graph indicates the average CPU / Memory usage during the period, and the solid line shows the actual CPU / Memory usage during the period. In addition, when you mouse over the graph, you can see a pop-up window that allows you to check the details of the period.

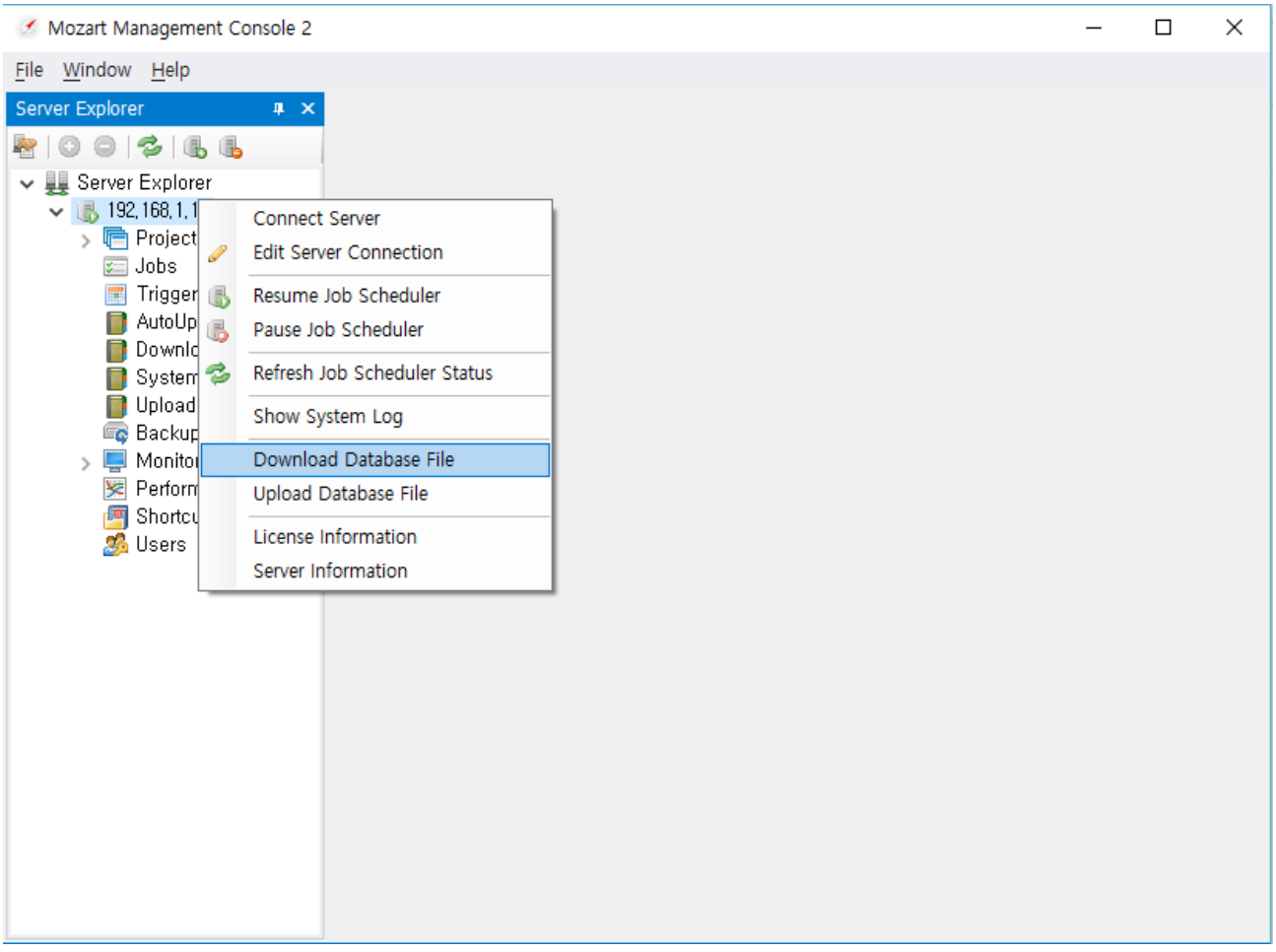
Local DB 파일 다운로드 방법

Project 생성 및 변경 이력, Job 이력, Trigger 변경, 실행 이력 및 사용자 정보 등의 정보들은 Server의 Local DB (SQLite DB 파일)에 기록이 됩니다. 네트워크 문제 등 기타 장애로 인하여 DB 기록 누락 사항이 발생 할 수 있습니다. 이러한 경우 LocalDB 파일로 부터 정보를 제대로 가져오지 못하여 등록된 Job/Trigger가 갑자기 안 보이거나 Project의 등록된 파일들을 읽어 오지 못하는 문제 등이 발생하는 경우가 있습니다. 문제를 해결하기 위해서는 문제가 되는 Row를 처리해야 하며, 그러한 작업 진행을 위해 LocalDB 파일을 다운받아 제품 관리자에게 전달을 해야 합니다. 다음은 MMC를 통해 LocalDB 파일을 다운로드/업로드 방법에 대해서 기술합니다.

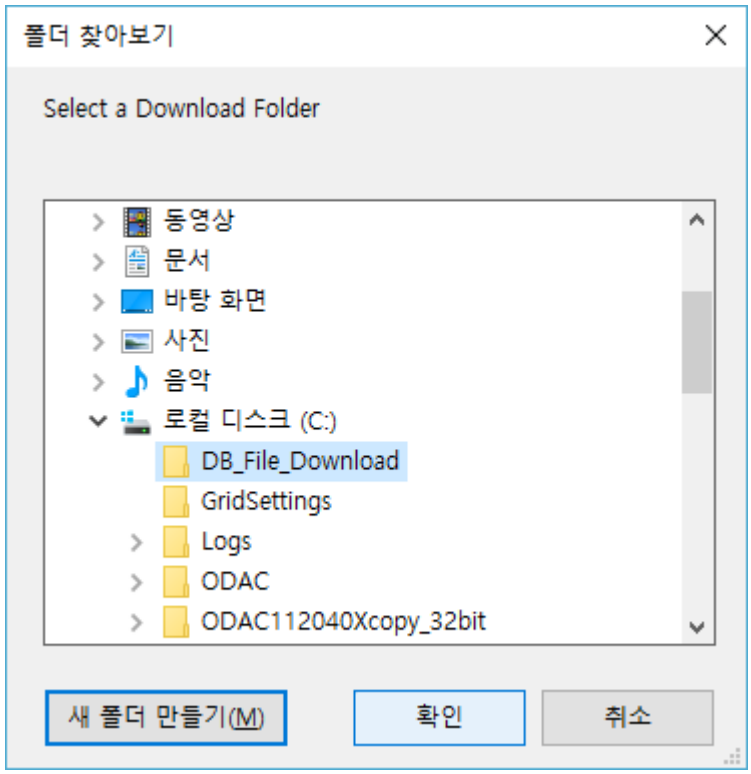
LocalDB 파일 다운로드 방법

1. LocalDB 파일을 다운르도 받을 대상 서버 노드를 Server Explorer에서 선택합니다.

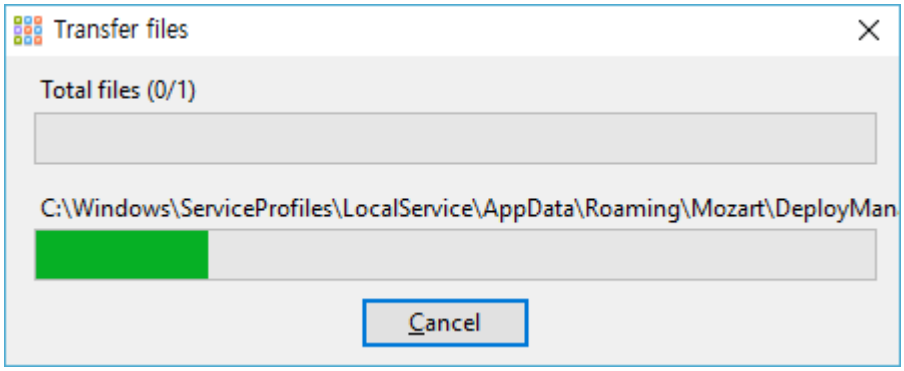
2. 선택한 서버 노드에 마우스 오른쪽 버튼 메뉴를 통해 **[Download Database File]** 를 선택합니다.



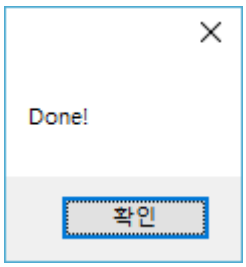
3. LocalDB 파일을 다운로드 받을 경로를 지정하고 **[확인]** 버튼을 클릭합니다.



4. 다운로드가 진행이 되면 아래와 같이 Progress Bar가 나옵니다. 다운로드 중 취소를 하고 싶으면 **[Cancel]** 버튼을 클릭합니다.



5. LocalDB 파일 다운로드가 완료되면, "Done"이라는 팝업 메시지가 출력됩니다. **[확인]** 을 클릭합니다.

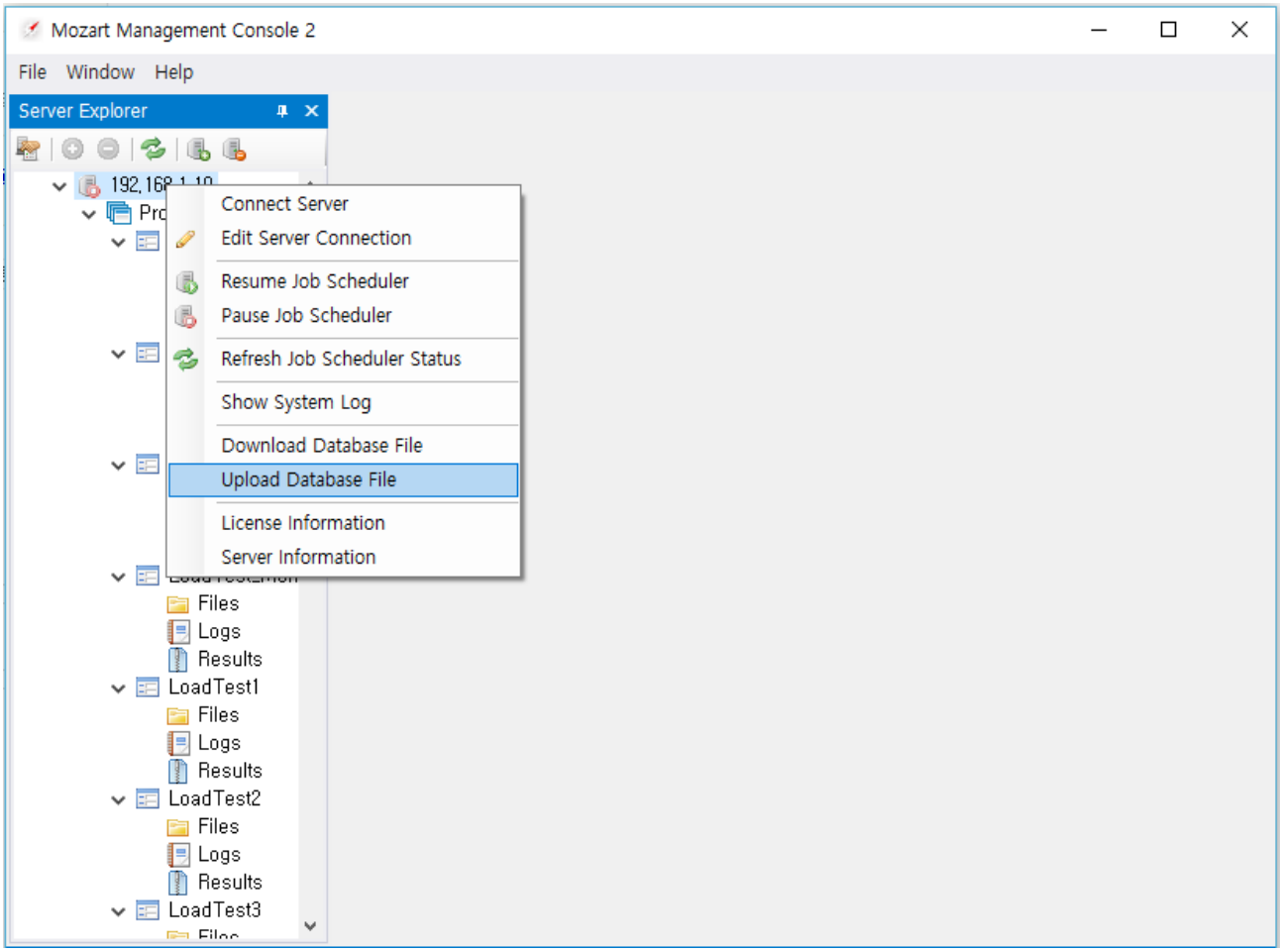


6. 다운로드 경로에 **"DeployManagement.db"** 라는 파일이 존재하면 LocalDB 파일을 정상적으로 파일을 다운로드 한 것입니다.

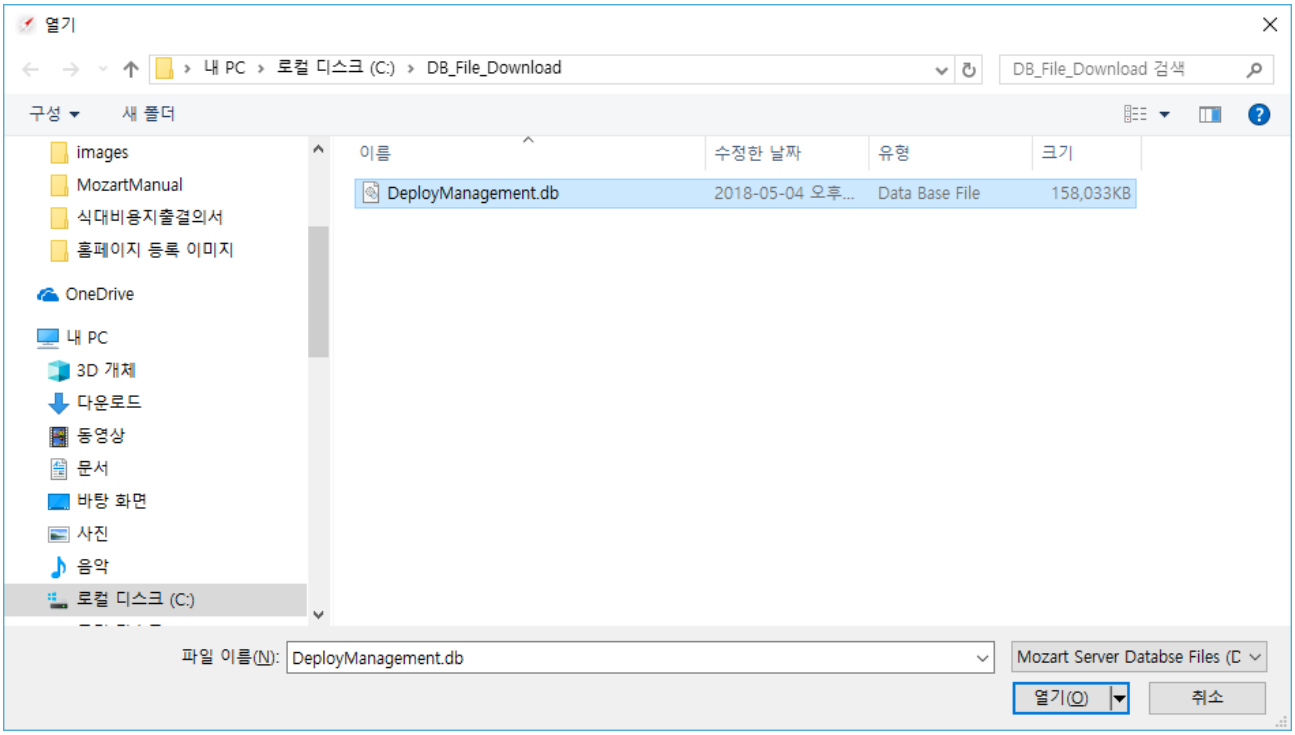
LocalDB 파일 업로드 방법

1. LocalDB 파일을 업로드 할 대상 서버 노드를 Server Explorer에서 선택합니다.

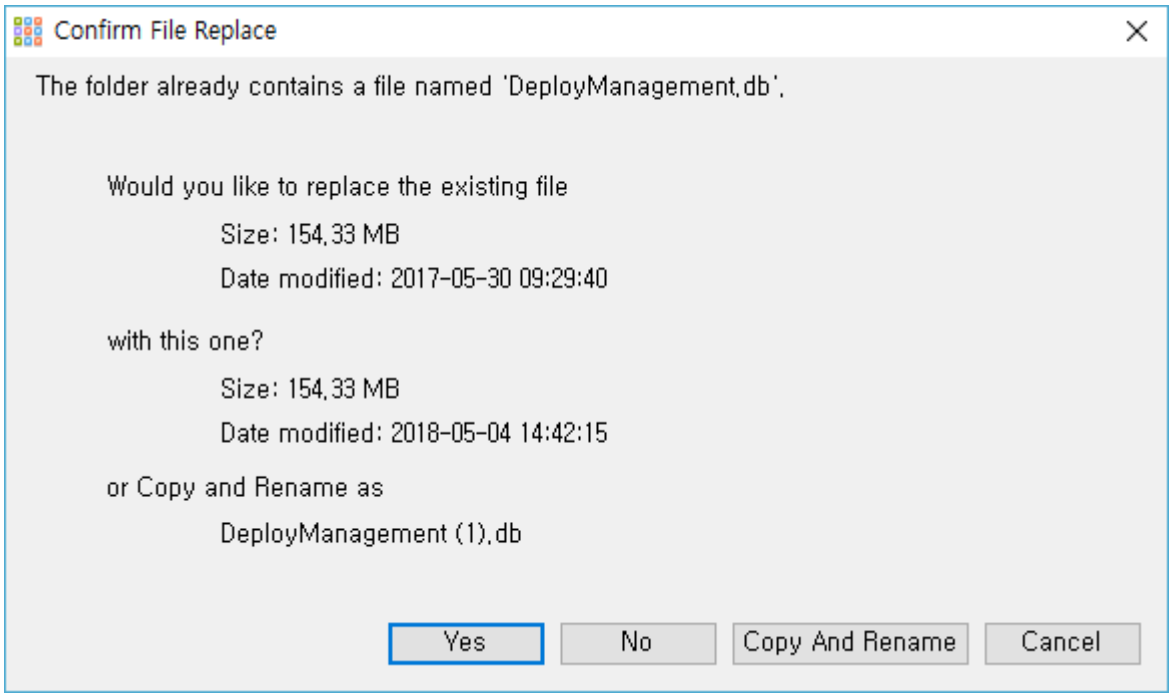
2. 선택한 서버 노드에 마우스 오른쪽 버튼 메뉴를 통해 **[Upload Database File]** 를 선택합니다.



3. 업로드 할 DeployManagement.db 파일의 위치를 지정합니다.



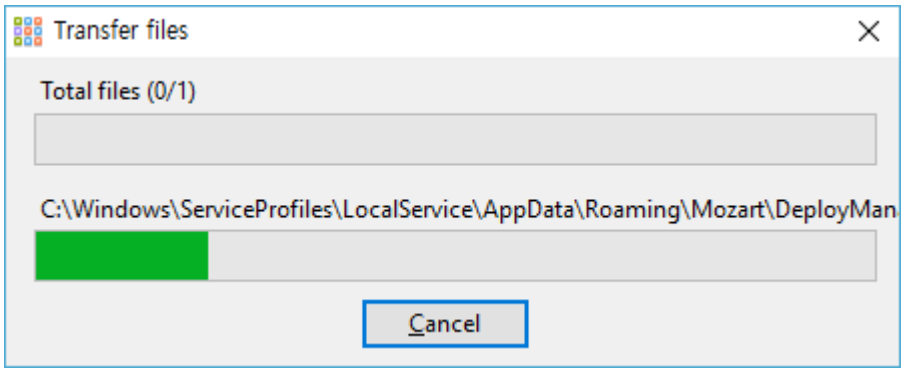
4. Server 파일이 이미 존재하는 경우 파일을 덮어쓸지 복사본을 만들지 선택을 합니다. [Yes]를 클릭 시 기존 파일을 덮어씁니다. [Copy And Rename]을 선택 시 파일의 복사본을 만듭니다.



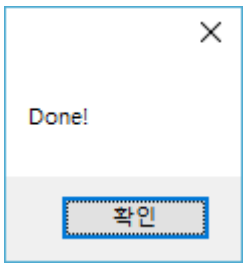
> Server가 운영되는 동안에는 Server의 LocalDB 파일에는 지속적으로 이력이 기록이 됩니다. 만약 LocalDB 파일을 업로드하기 전 Trigger 실행 이력을 제외한 Projects/Job/Trigger

> 에 파일이나 설정 변경이 있는 경우에는 [Copy And Rename]으로 파일을 업로드 하는 것을 권장 드립니다.
>
> 이후 원본과 복사본 파일을 비교하여 테이블을 합치는 쿼리를 수행하여 원본 DB 파일에 오류 수정과 최근 기록을 유지하도록 합니다. 또한, 이러한 작업을 진행 시에는 쿼리가 실행되는 동안 원본 DB파일에 추가적으로 기록되는 것을 피가힉 위해 Server Explorer에서 대상 Server를 선택하여 [Pause Job Scheduler]를 실행하여 Job Scheduler Service를 임시 중단합니다.

5. 업로드가 진행이 되면 아래와 같이 Progress Bar가 나옵니다. 업로드 중 취소를 하고 싶으면 **[Cancel]** 버튼을 클릭합니다.



6. LocalDB 파일 업로드가 완료되면, "Done"이라는 팝업 메시지가 출력됩니다. **[확인]** 을 클릭합니다.



Project Management

Project is a management element for managing vmodel file, Task dll file management, and update history mapped to job target / trigger. Project can distribute vmodel and dll files based on a schedule and it is possible to manage the history by creating changes at each distribution because of the introduction of configuration management. If Job / Trigger is mapped to Project and changes are made in Project, it checks whether the mapped Job / Trigger are executed and distributes files when it is not being executed by mapping Job/Trigger to Project.

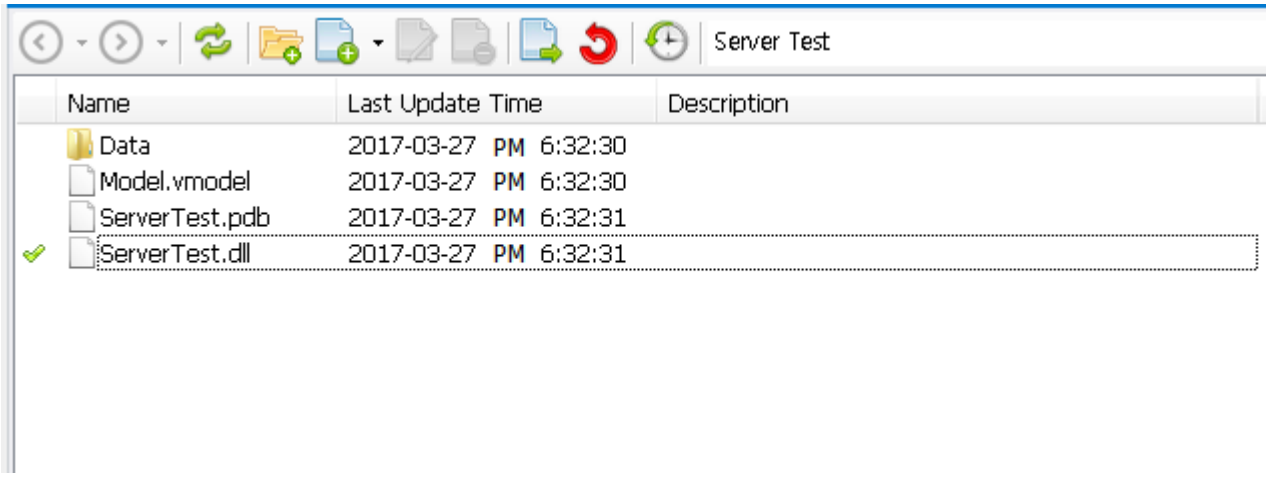
- **Project Registration** : Describing the process of registering a project.
- **Project Modification/Deletion** : Describing how to edit/delete a project that you have added..
- **File Commit and Deploy** : Describing how to distribute files and folders through Project.

File Commit and Deploy

Register File

1. In Server Explorer-> Projects node, double-click the Files folder of the project for registration or select [Open] from the right-click menu..

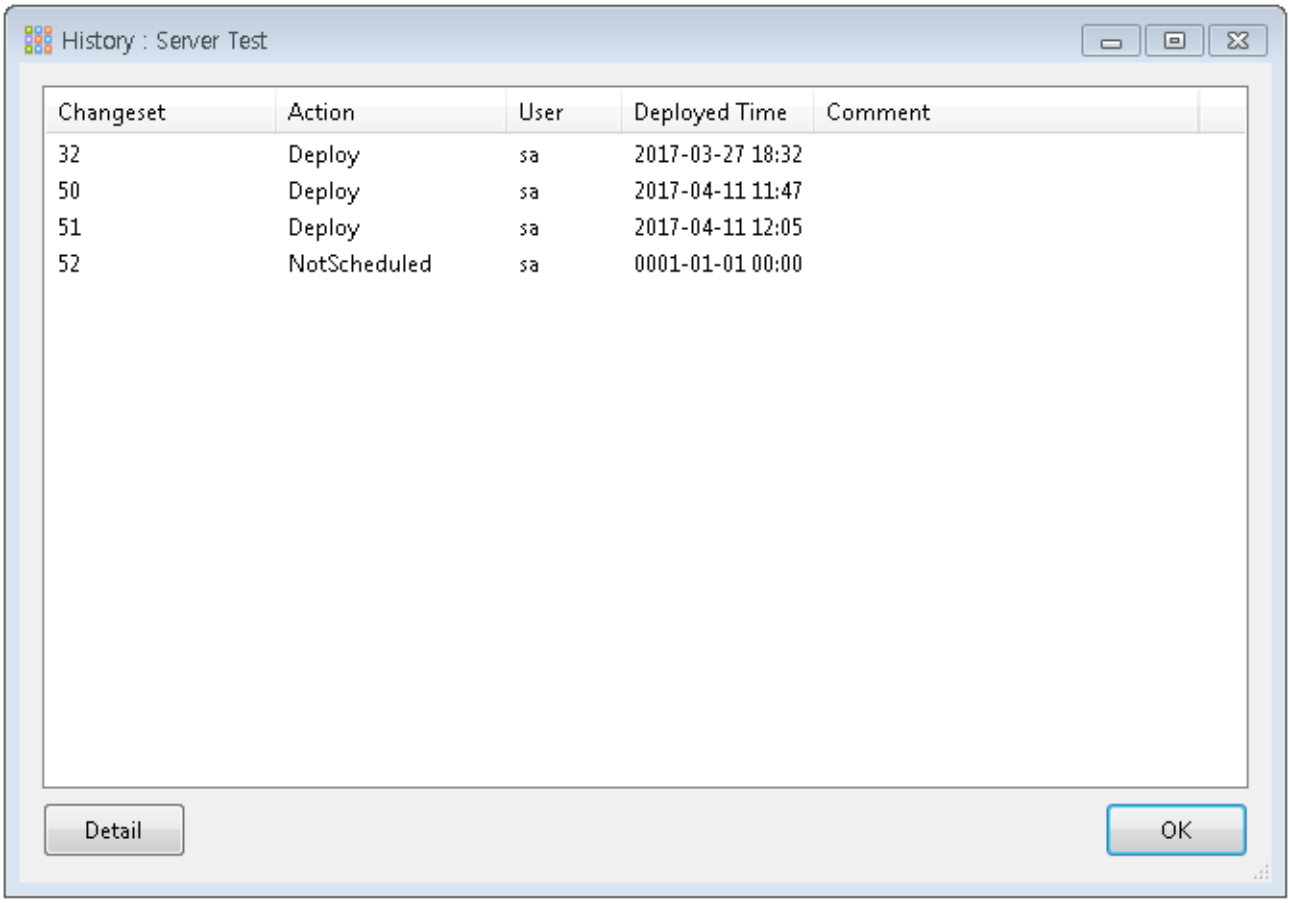
2. Select the files and folders to be uploaded by clicking  **[Add]** button on the top menu. If the file is newly added, "+" is displayed next to the file. If it is deleted, "-" and if it is changed, a check mark is displayed as shown below..



Top Menu Button Description

-  : Refresh button to update Deploy status of files in Files folder.
-  : Create Folder button to create folder in Files folder.
-  : Add button to register / change files / folders in Files folder. If you click the arrow next to the icon, you can choose whether to upload in file or folder.
-  : Update button for changing files in Files folder. The function is the same as Add, but when you select Update, the selected files are only searched in Explorer.
-  : Delete button to delete the selected file from the Files folder.
-  : Commit button to reflect the above Create Folder / Add / Update / Delete.
-  : Cancel button to cancel all changes in Files folder.

-  : View History button for viewing all past changes in the Files folder



View History Window

- **Changeset** : It is the changeset number of the Projects. Changesets are not managed and created individually by Projects. (i.e. Project A : Changeset 1, Project B : Changeset 1,..). Regardless to the Projects, if commits are made from any of the Projects, Changeset number will be increased by 1 from the previous Changeset. (i.e. Project A : Changeset 1, Project B : Changeset 2,..)
- **Action** : This shows the Commit status.
  - *Deploy* : Committed files have been deployed and saved to the server.
  - *NotScheduled* : Files are commited and stored in the Changeset folders, but not deployed to ther server yet. .
  - *Deploy > Rollback* : Files are rolled back and cannot be distributed to the server due to running Job/Trigger. This happens when commited files are trying to be distributed when nothing is set on Related Items option in Deploy Schedule tab while Job/Trigger is still running. An error message will be written to Comment column when Deploy > Rollback occurs.

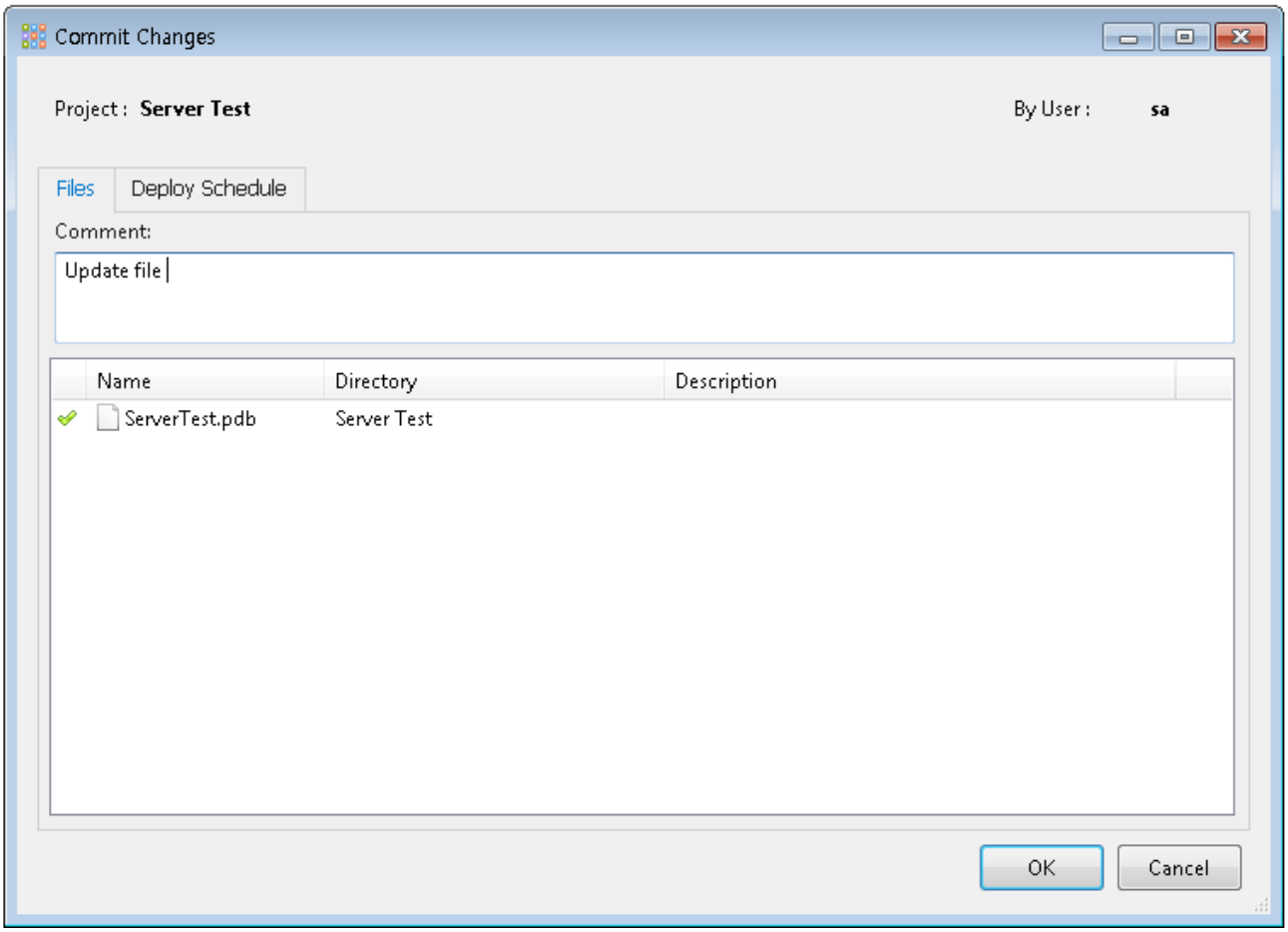
- **User** : Shows the user account that performed Commit.
- **Deployed Time** : The time when commited files have been deployed to the server (Format: YYYY-MM-DD hh:mm).
- **Comment** : Comments left by the user during file commit.

File Commit and Deploy

1. When the file registration is completed, click the  **[Commit]** button on the top menu.

At this point, you can set the time to distribute the files in the Deploy tab.

2. When the setting is completed, click the **[OK]** button.

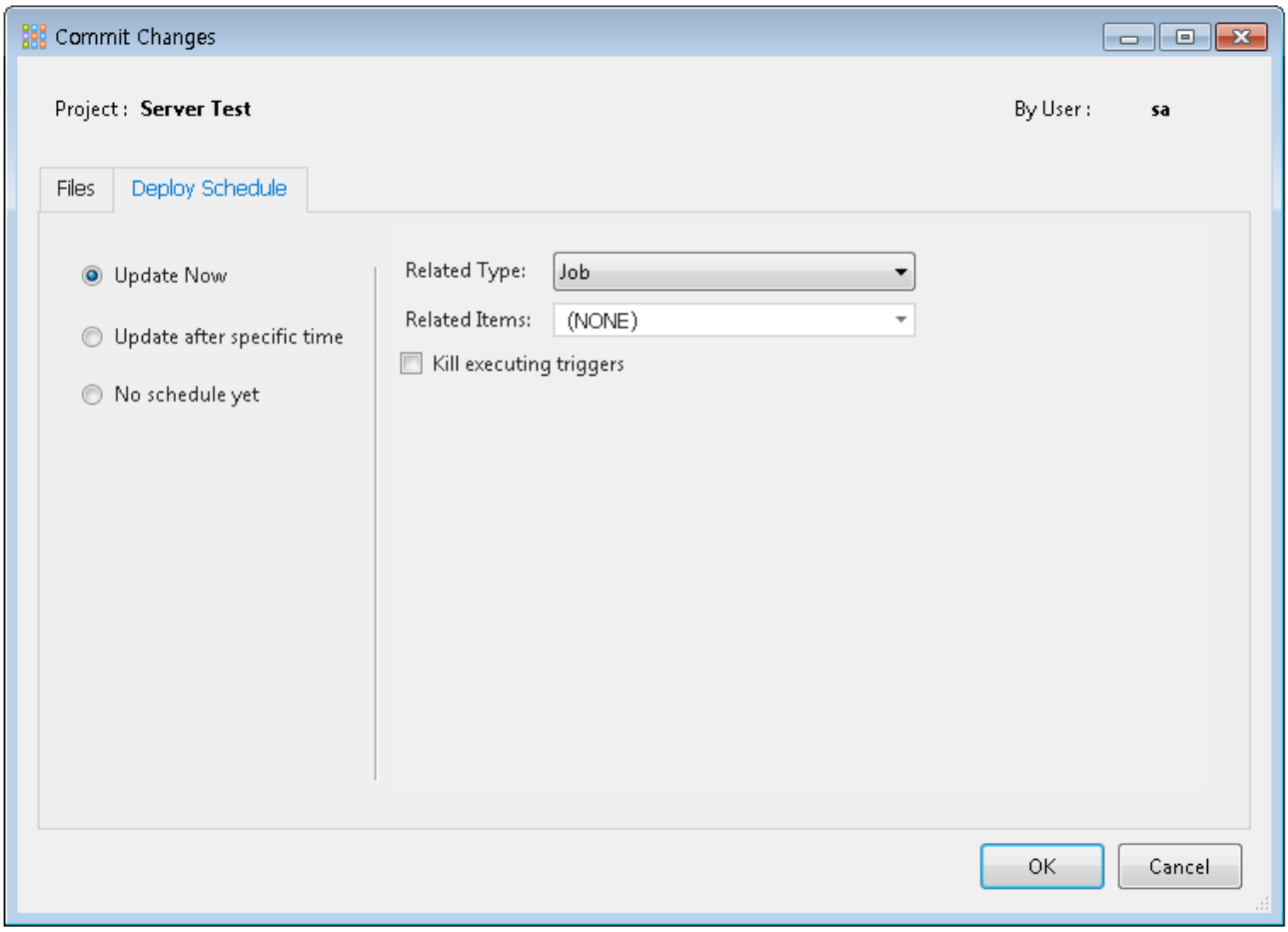


Commit Changes Dialog

- **Files Tab**

- **Comment :** It is used to create user comments on the changed point. You can check comments in [**View History**].
- **Name** : File name to be committed.
- **Directory :** Relative path information for [Project Name] in [WorkingDirectory]/Projects/path.

- **Deploy Schedule Tab**



- **Update Now** : Start the distribution of files from the moment that you press the [OK] button. If Related Items are specified, the relevant Job / Trigger will distribute files after execution. If Related Items is (NONE) and related Job / Trigger is running, the file distribution is canceled and files are rolled back
- **Update after specific time** : Distribute committed files to the server at the user-specified date and time. Like Update Now, if Related Items is (NONE) and related Job / Trigger is running, the file distribution is canceled and files are rolled back.
- **No schedule yet** : This option is selected when the registration / change files are committed but not deployed to Server. No schedule yet files can be distributed to the

Server via Update Now or Update after specific time.

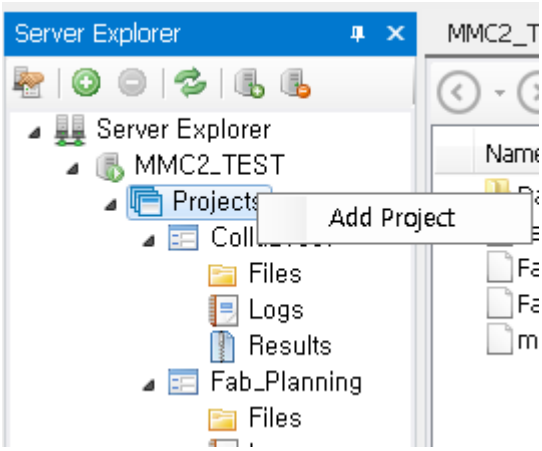
- **Related Type** : Whether to check the related job or trigger is terminated when distributing registered / changed files. If Related Type is set to Job, it checks whether all triggers connected to the selected Job in the Related Items and proceeds to the distribution. If set to Trigger, it checks whether the selected triggers are executed in the Related Items and proceeds to the distribution.
- **Related Items** : Check whether the registered Job or Trigger is selected based on the items set in Related Type.
- **Kill executing triggers** : If the selected Job / Trigger of Related Type / Related Items is being executed and Kill executing triggers is checked, the job / trigger is immediately stopped and distributes files. This option is selected if the priority of file distribution precedes triggers.

How to Register Project

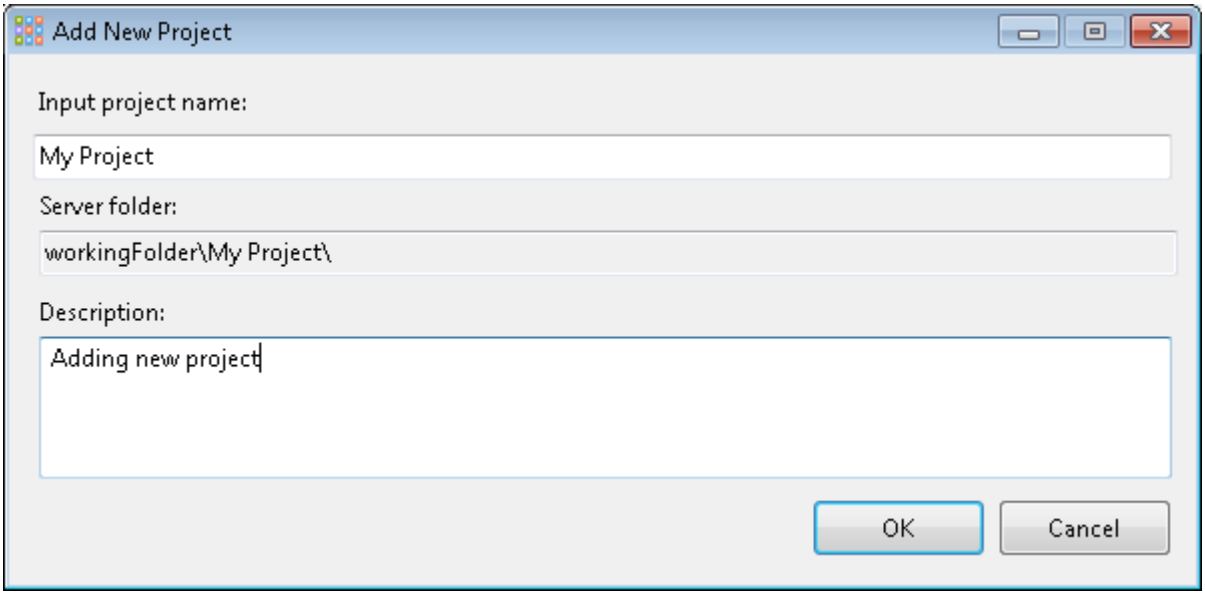
In the previous version of MOZART Management Console, users were required to create accessible folders through Shortcut in order to upload Task model and dll files and to access trigger results and logs. From MOZART Management Console (2.0), creating folders to Shortcut are unnecessary. When a Project is created, Files/Logs/Result folders are created automatically. When the Project is mapped to a Job the logs and trigger results will automatically directed to be stored to the Logs/Results folders of the corresponding Project. The following describes how to add Projects through MMC.

Project Registration

1. Right-click on the Server Explorer -> Projects node and select **[Add Project]**.



2. Enter the information to Input project name and Description textbox, and click the **[OK]** button to save.



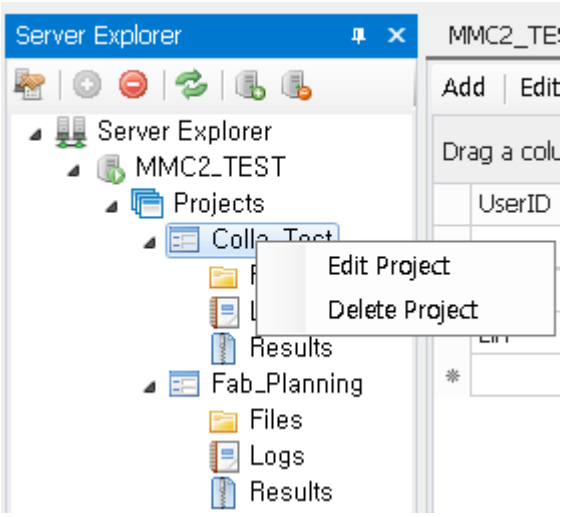
Add New Project Information

- **Input project name** : Name of the project to be created. A folder with Project name will be created to Projects/Results/Logs folders within WorkingDirectory. Project name cannot be edited after creation.
- **Server folder** : The project folder to be created in in [WorkingDirectory]\{Projects/Logs/Results}\. Users cannot edit this item.
- **Description** : Textbox to type in description for the project to be created. The Description section can be edited through Edit Project.

How to Edit and Delete Project

Project Modification

1. Select the Project to be edited among Projects added to Server Explorer -> Projects. Right-click and select **[Edit Project]**.

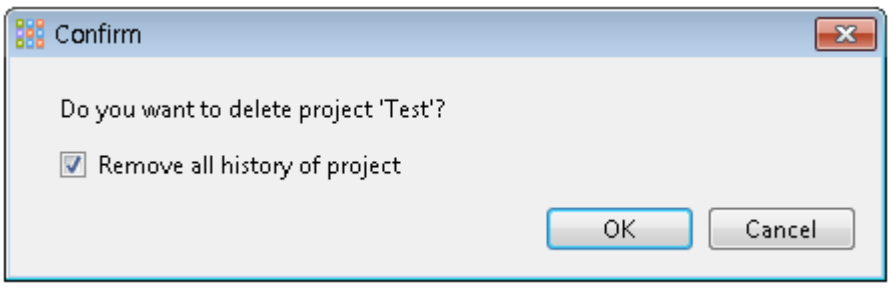


2. Only Description can be modified. After editing, click **[OK]** button to save.

Deleting Project

1. Select the Project to be deleted among Projects added to the Server Explorer -> Projects. Right-click and select **[Delete Project].**

2. Click **[OK]** button in the Confirm popup window to delete the Project, then the Project folder and files selected in [WorkingDirectory] \ Projects will be deleted.



- **Remove all history of project** : This option is to decide whether to delete all histories of the selected Project. If the checkbox is enabled, files in ChangeSet folders and data in Deploymanagement.db will removed when the Project is deleted.

Synchronizing Project Among Mozart Servers

When creating a failover system for MOZART Server, the project, job, and the trigger of the backup server have to be the same from what is in the operation (main) server. In other words, the procedures and steps took to compose the main server have to be repeated from the backup server as well to be prepared. However, when this task is done through human resource only, this may lead to increasing the risk due to human error and longtime consumption for preparation.

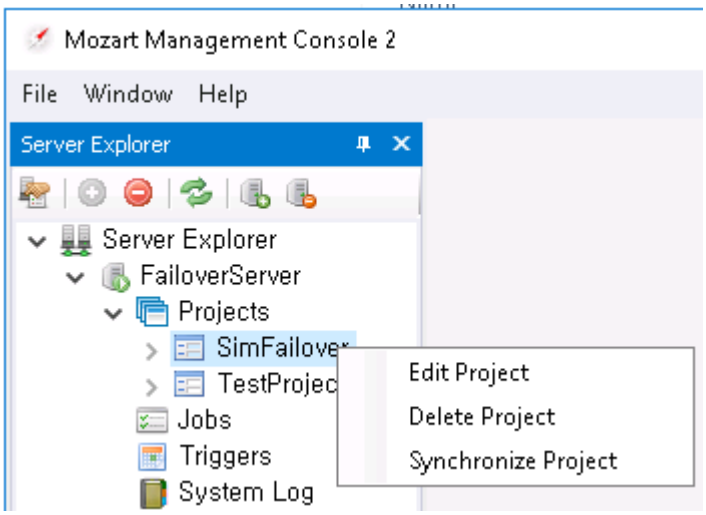
From 2019.115.000.0 version the feature to support to create the backup server for failover is included. The functions that could be used are **1)Replicating the project, job, trigger of the source server to the target server**, **2)synchronizing the source server to the target server in case changes are made from the source**, and **3) leaving logs for the synchronization among source and target**.

How to Use

The vModel file or task DLL file can be updated to the operation server of the MOZART Server. In this case, the updated files should be applied in case a backup server exists. By using the synchronization function, the target(backup) server can pull from the source(operation) server. The following explains the steps on how to use the synchronization function.

1. Connect to the source server from the `Server Explorer` from MMC.

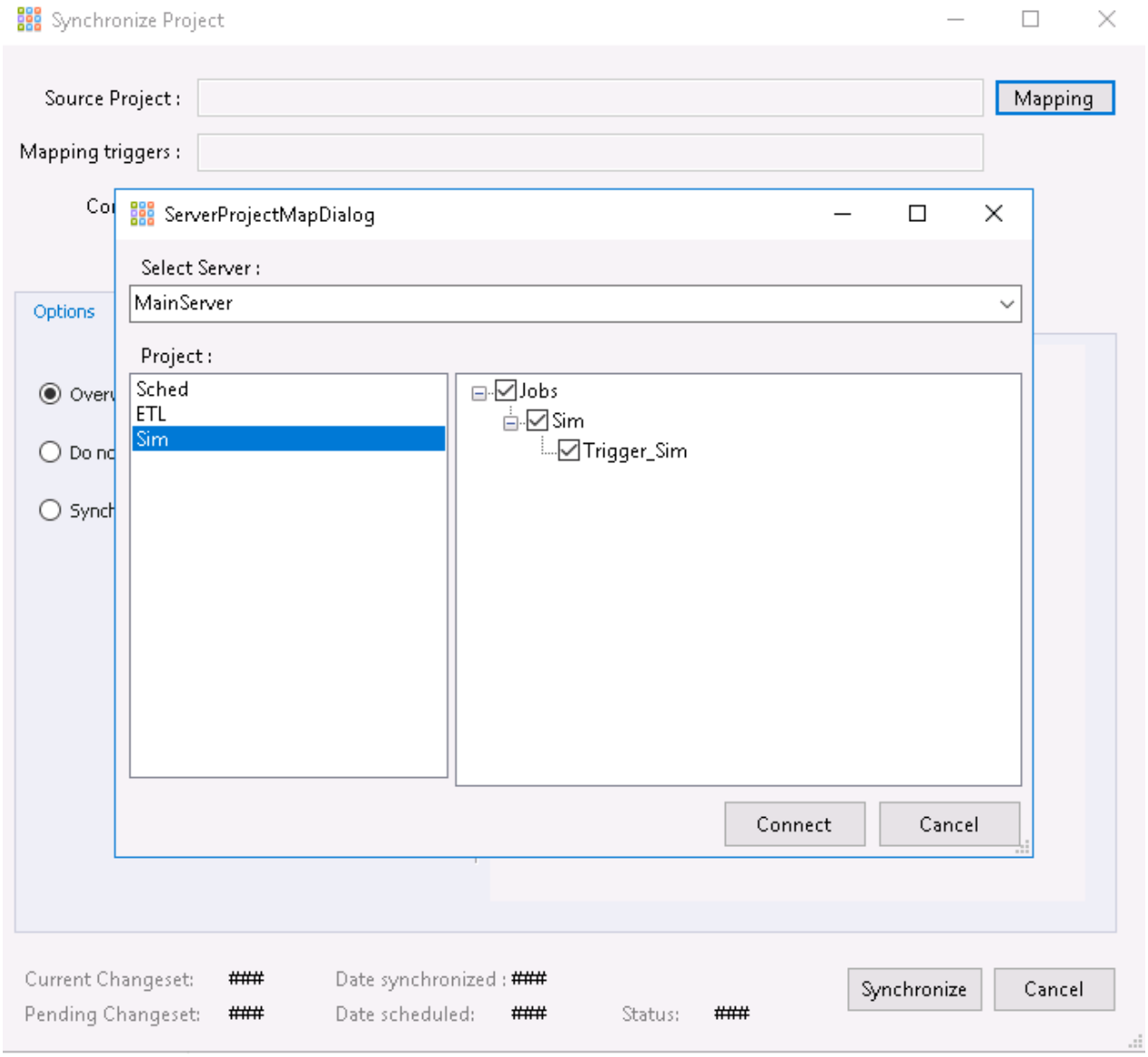
2. Next, connect to the target server from the `Server Explorer` and then select a project to synchronize from the source server. (Ex.SimFailover)



3. Right click on the selected project and select **[Synchronize Project]** from the menu.

4. Click **[Mapping]** button from the `Synchronize Project` dialog to select the source server. In normal cases, when the target project is created should replication, the source server should be already selected.

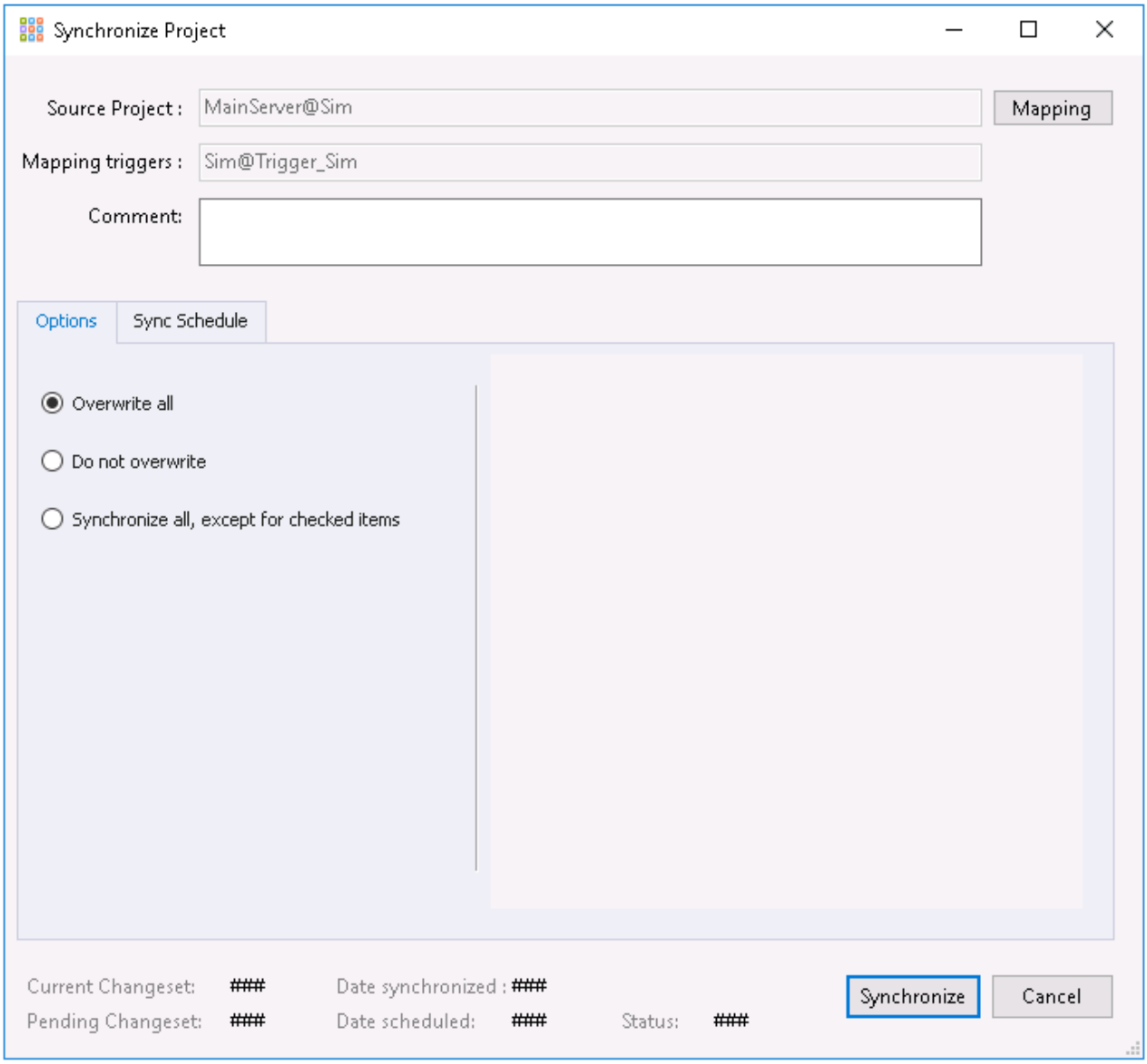
- *Source Project* : Gets the list of the source server and the project to synchronize.
- *Mapping Triggers* : The selected job and trigger to sync, which was chosen from the **[Mappings]** of the `Source Project` .



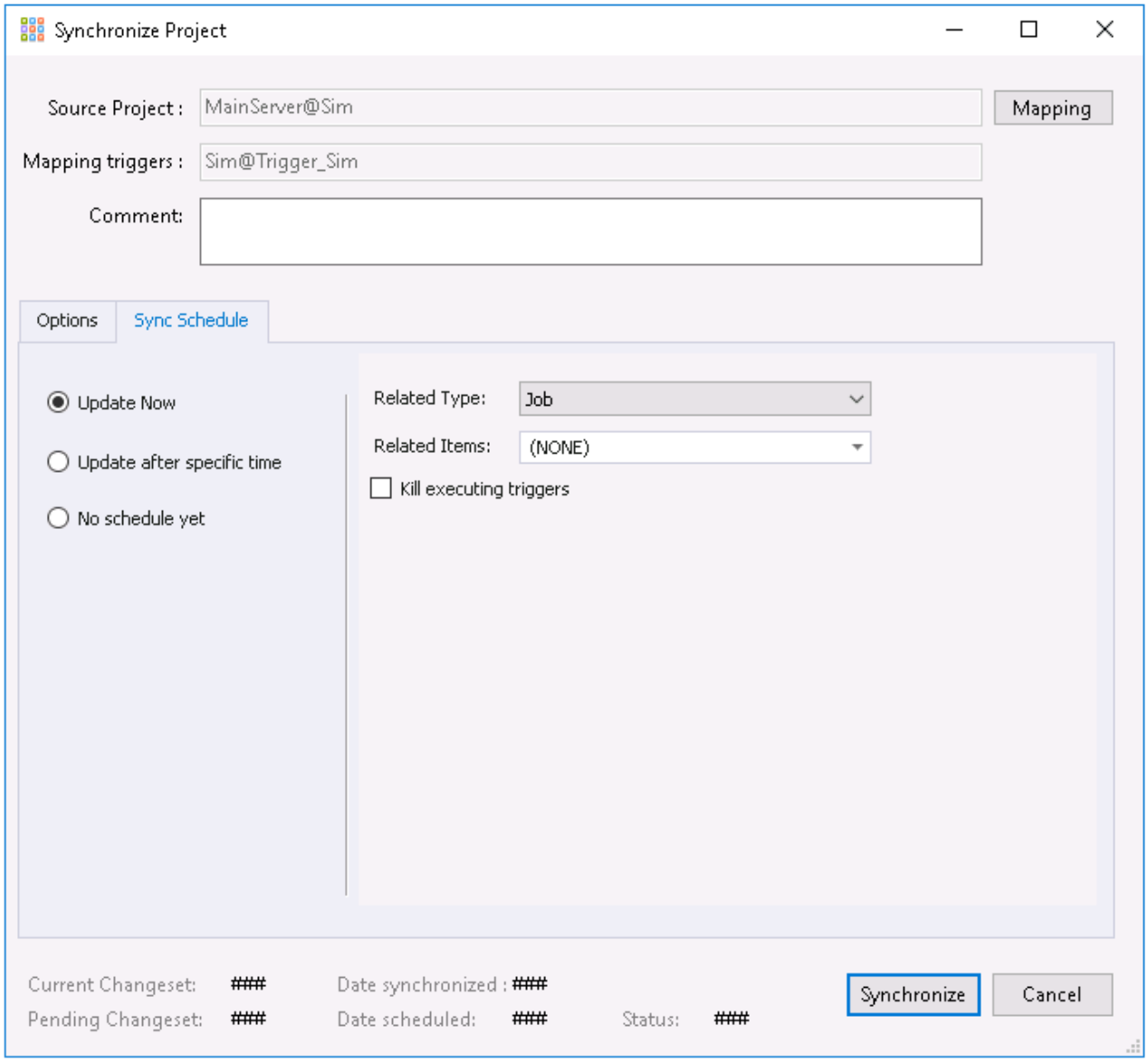
5. Once the source server is selected, next select the project to pull from the `Project` section.

6. When a project is selected, the list of the jobs and the triggers mapped to the project will appear on the right side panel. Select the job and the trigger to pull from the source server.

> **Overwrite all** : Overwrites the existing Job/Trigger, schedule and the arguments to the ones from the source server.
> **Do not overwrite** : Does not perform anything.
> **Synchronize all, except for checked items** : Synchronize all except for the selected arguments of the job and trigger.

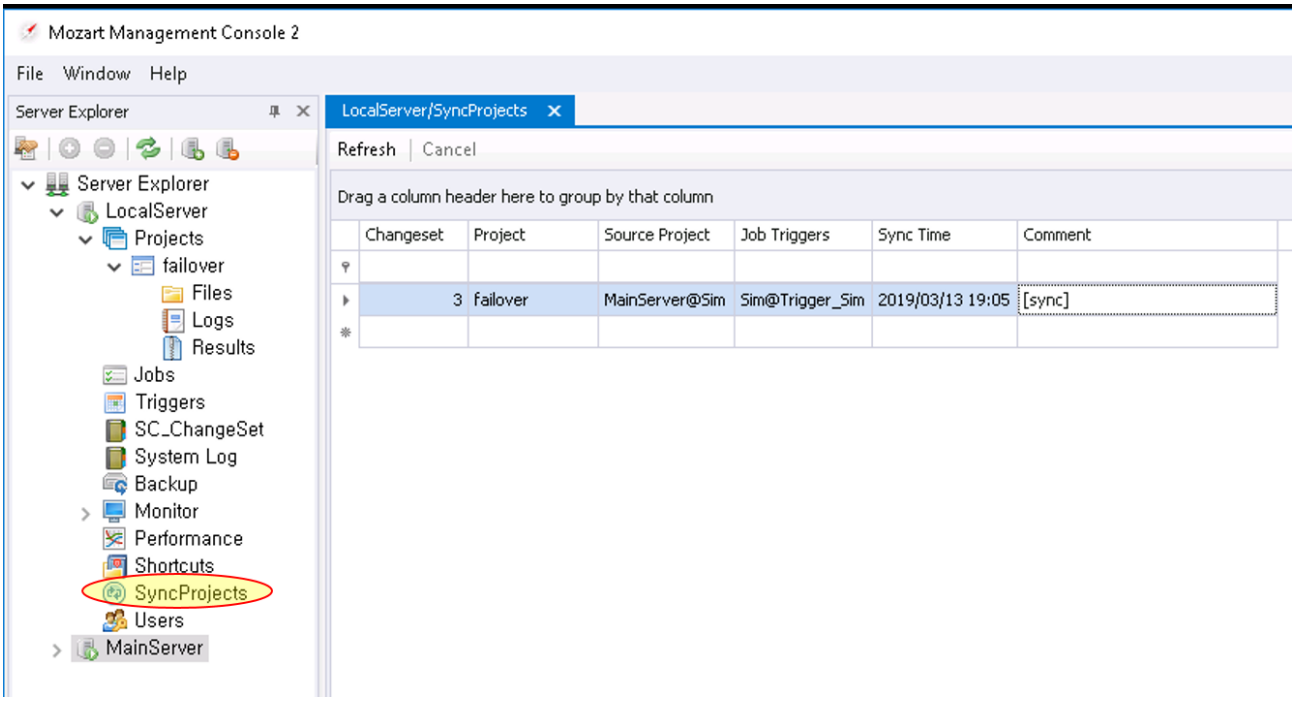


7. Once the settings for synchronization is completed, next, go to `Sync Schedule` tab from the `Synchronize Project` dialog to set the schedule to start synchronization.

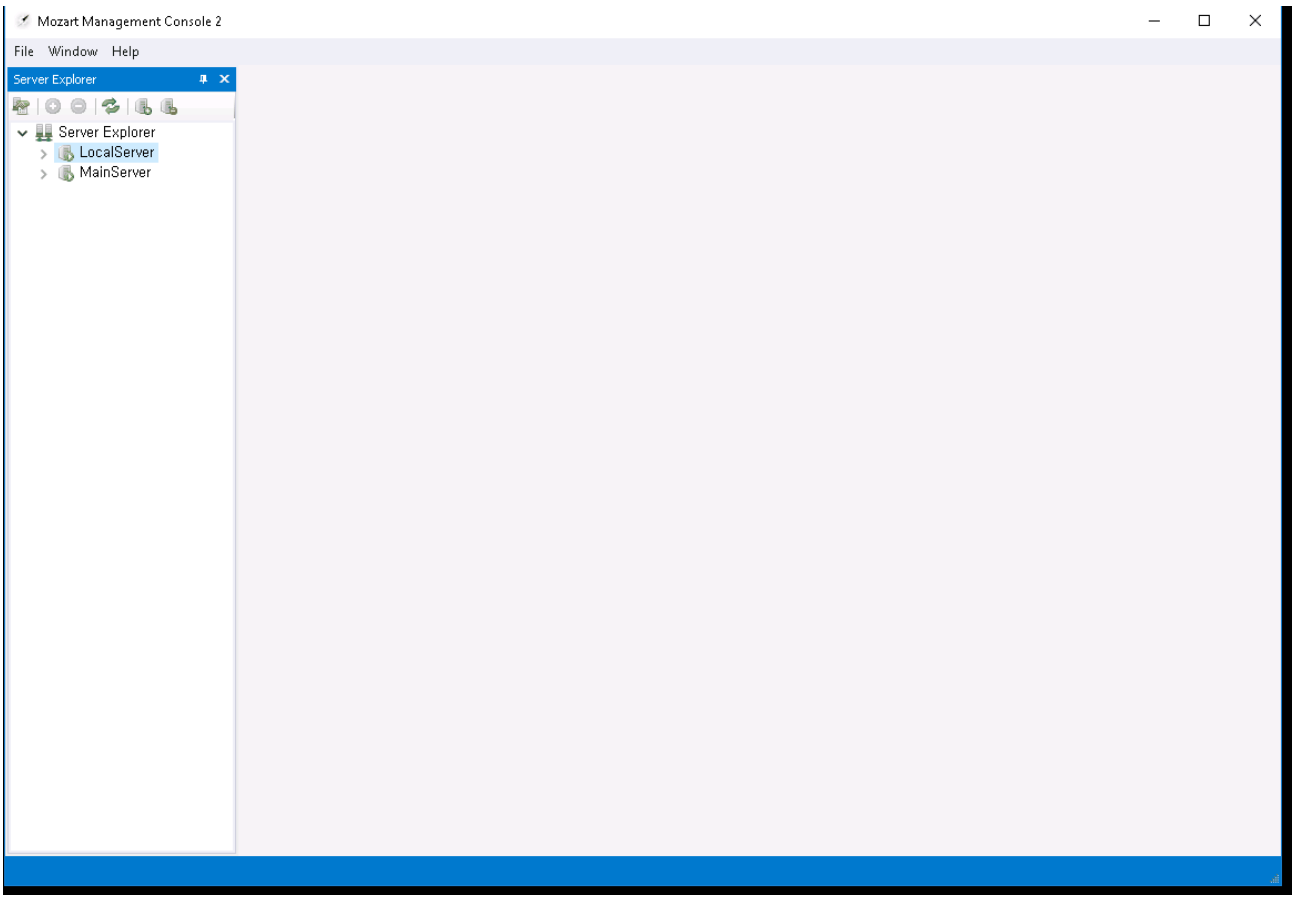


8. Click **[Synchronize]** button to synchronize from the source server.

9. Once synchronization is performed, you may check the history of the synchronization among the source and the target server through `SyncProjects` node in `Server Explorer` of the target server.



Changeset : The unique identification number or changeset ID of the synchronization.
Project : The name of the target project that was synchronized.
Source Project : The name of the source project.
Job Triggers : The name of the job and trigger synchronized. @ is delimiter for job and trigger. (Ex. JobName@TriggerName)
Sync Time : The time synchronization was completed.
Comment : The user comment for the synchronization. If synchronization fails (State : Fail), an error message will be left automatically.


Mozart Management Console 2
File
Window
Help
Server Explorer
Server Explorer
LocalServer
MainServer

How to Use Job Scheduler

The following shows how to manage Job Type and Job and how to schedule and manage Job execution through MOZART Management Console.

- **Registering/managing Server** : This explains how to register and manage MOZART Server through MMC.
- **Registering/managing Job Type** : This explains how to register and manage executable Job Types to the target Server.
- **Registering/managing Job** : This explains how to register a Job to the target Server and how to monitor the Job status.
- **Registering/managing Trigger** : This explains how to register a Trigger and how to monitor the Trigger status.
- **Registering/managing Shortcut** : This explains how to register a Shortcut and how it could be explored through the server.

How to Manage Job Type

Job Type is a type of Job that can be executed by Job Scheduler. Job Type can be categorized into two types. There is general type, which are sending e-mails or running programs. The other is MOZART exclusive job types, which are Simple Task, Model Task, Collaboration Task. These types are developed using MOZART.

The task of Model Task and Collaboration Task depends on the target Model and argument settings. Therefore, these two job types needs to be pre-defined. In addition, general type jobs require being pre-defined as well. Only Simple Task does not need to be pre-defined because the task is not based on Model.

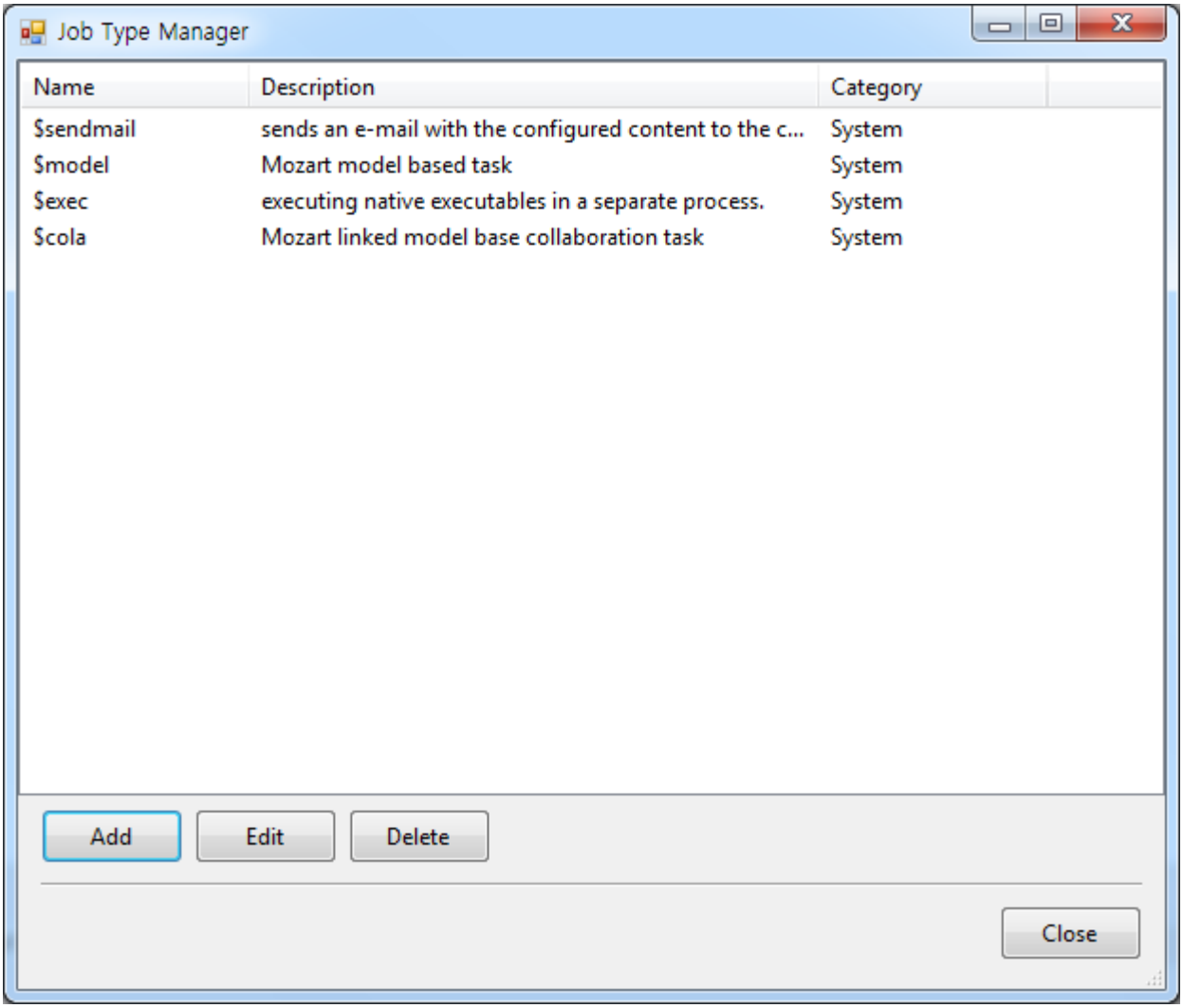
Job Type can be added, modified or deleted through **[Edit Job Type]** menu in Job inquiry page.

Inquire Job Type

1. Select a target server to inquire.

2. Click **[Manage Job Types]** menu in Jobs node.

3. Registered job types and pre-defined job types from the system can be inquired through Job Type Manager dialog.



> Job Types with $ indicator are job types defined by the system. These job types cannot be edited by user.

4. If it is not System Job Type, double-click the name or press **[Edit]** button on the bottom of window in order to inquire or modify the information of the job type.

Registering Job Type

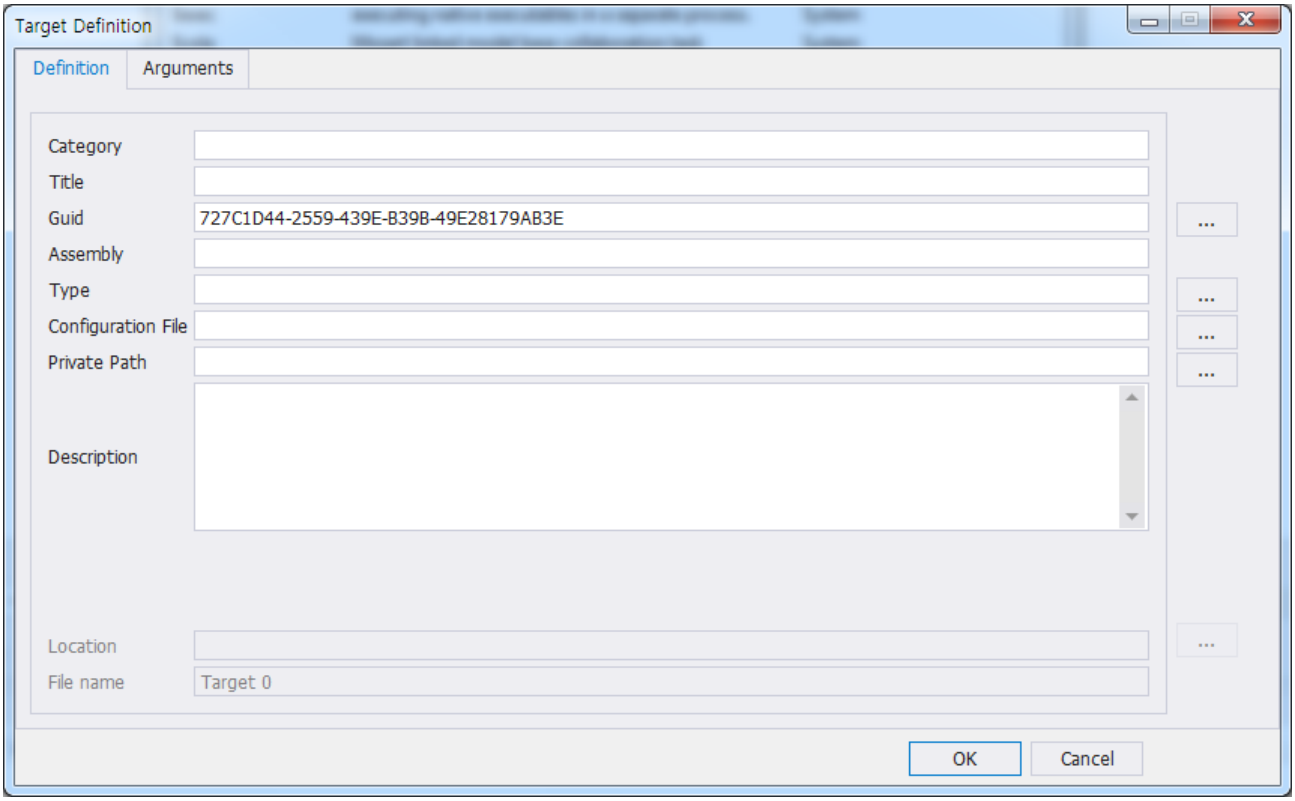
1. In order to register a new Job Type, select a target server in Sever Explorer.

2. Right click on the jobs node of the target server and then click **[Manage Job Types]** menu.

3. Click **[Add]** button on the bottom.

4. A dialog to define the job type will be opened.

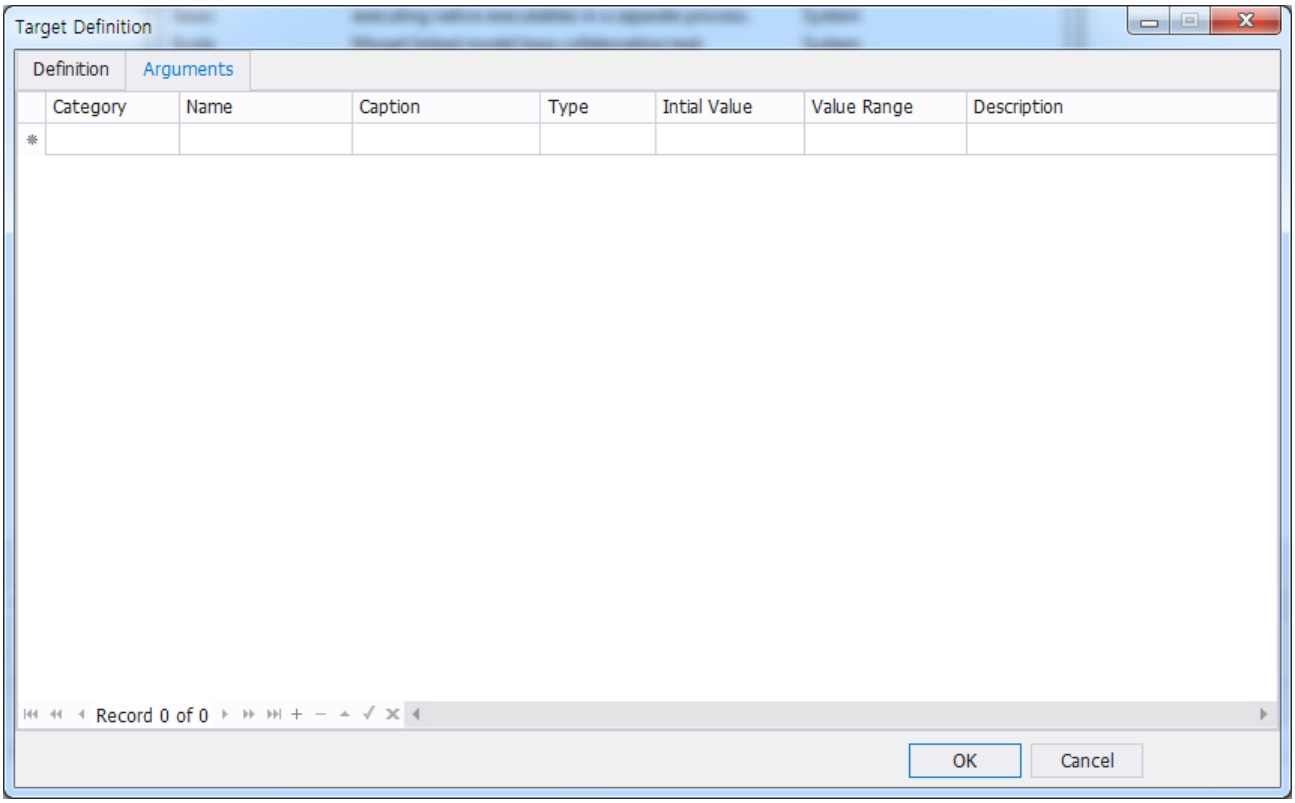
5. Fill in the information through **Definition** tab.



- **Category** : Define Job Type's category. User can input arbitrarily.
- **Title** : Define Job Type name. This is used as an information defining Job Type in UI.
- **Guid** : Input GUID that can define Job Type solely. If a button at the right of Text Box is pressed, new GUID is automatically generated.
- **Assembly** : This designates Assembly/Type that implements an executable ITask in MOZART Framework.
    - Press **[...]** button on the right side of Type Text Box to activate Select a Type Dialog.
    - Click **[Load an Assembly]** to select the type for the new build DLL files.
    - An item that implements ITask in the selected Assembly is displayed in tree.
    - Select a Job Type to be registered and click **[OK]** button.

    - If Assembly, Type Text Box of Definition Tab are filled with the selected Type information, selection is properly processed.
- **Type** : This is selected together when assembly is selected.
- **Configuration File (Optional)** : Config file to configure the log history type of Job type. The config file should be copied to Working Folder in the server first to be registered. When this is done press **[...]**, to select the config file stored in the server.
- **Private Path (Optional)** : Private Path means a folder where the corresponding Job Type's Assembly file and its related file are saved. If path is not designated, Working folder is used. Working folder is designated when MOZART Server is installed and this is the only folder that can be accessed by MMC.
- **Description (Optional)** : Description of the job type can be included in this section.

6. If all required information is added to Definition tab, move to **Arguments** tab to define the arguments for the selected task.



7. The entered value of the corresponding Argument is used to configure how Job Type is working when Job and Trigger is registered.

Deleting Job Type

1. Select a target server in Server Explorer in order to register a new Job Type.

2. Right click on the jobs node of the target server and then click **[Manage Job Types]** menu.

3. Select a Job Type to be deleted from the list.

4. Click **[Delete]** button to delete the job type. Please note that job type provided by the system cannot be deleted.

Job Management

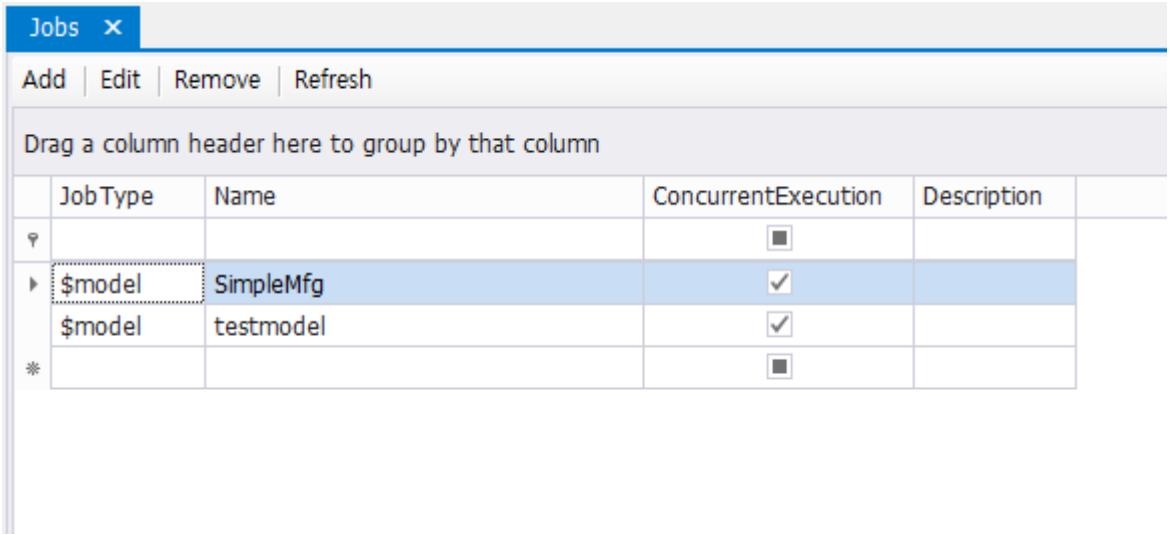
Job is a data defining how Job Type works according to which job types to be triggered and what arguments to be used. Registered Jobs can be searched through Job View for each Server in Server Explorer and add/delete/modify options for jobs can be used from the top side menu. Arguments set in job is used as the default parameter value configuring the Trigger of the job. In other words, when the value in Trigger is not changed, the job will be executed using the parameters set in job.

- **How to Register/Modify Job** : This explains how to register a new job or to modify information of the existing registered Job.
- **How to Delete Job** : This explains how to delete a registered job.

Register Job

1. Select a target server to inquire through **Server Explorer**.

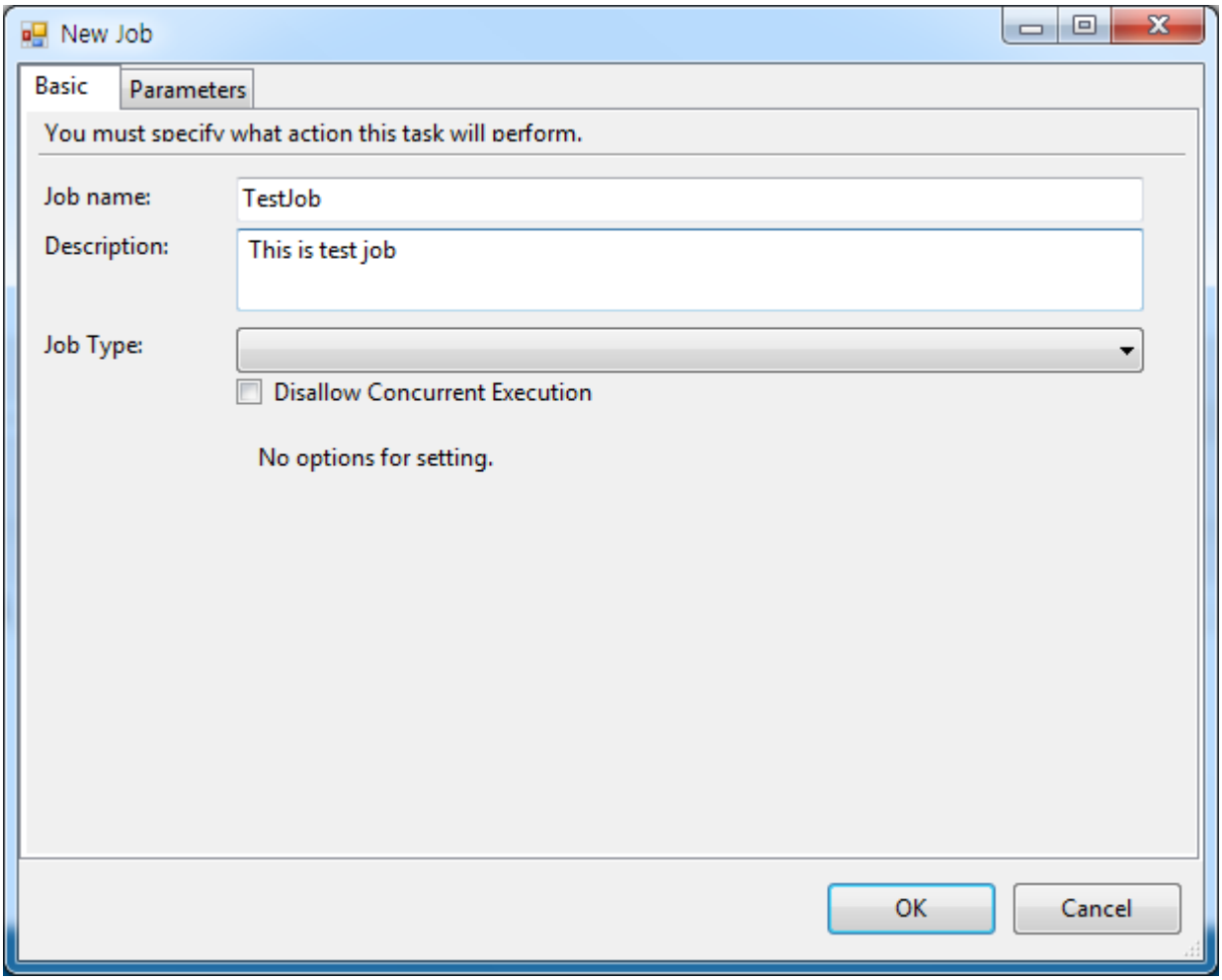
2. Double-click Jobs node of the target server to activate the window to inquire jobs.



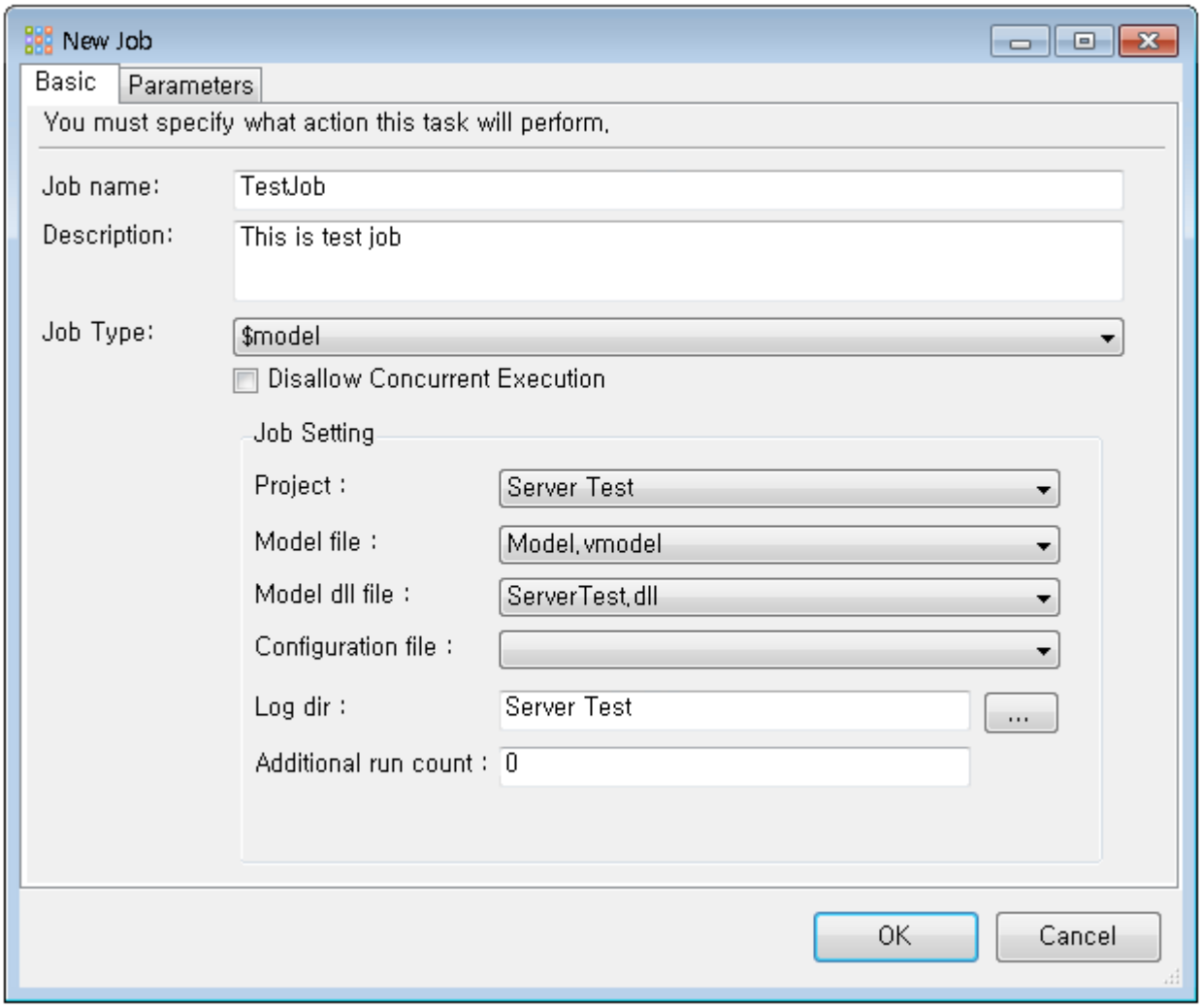
3. Click **[Add]** menu on the top menu bar.

4. Enter information on Basic and Parameters tab in "New Job" dialog.

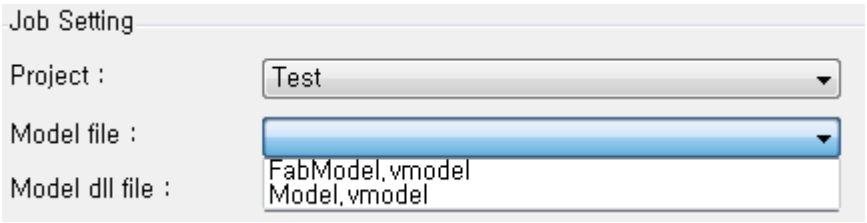
5. First, enter the information in Basic tab.



- **Job Name** : Name of Job. This name is used as an identifier to discern a job from other jobs.
- **Description** : Job description.
- **Job Type** : A combo box to select the job type. There are job types based on Model Task which are $model and $cola. The parameters needs to be changed according to model type and execution option. For these reasons these job types have additional Job setting inputs to decide the changes.

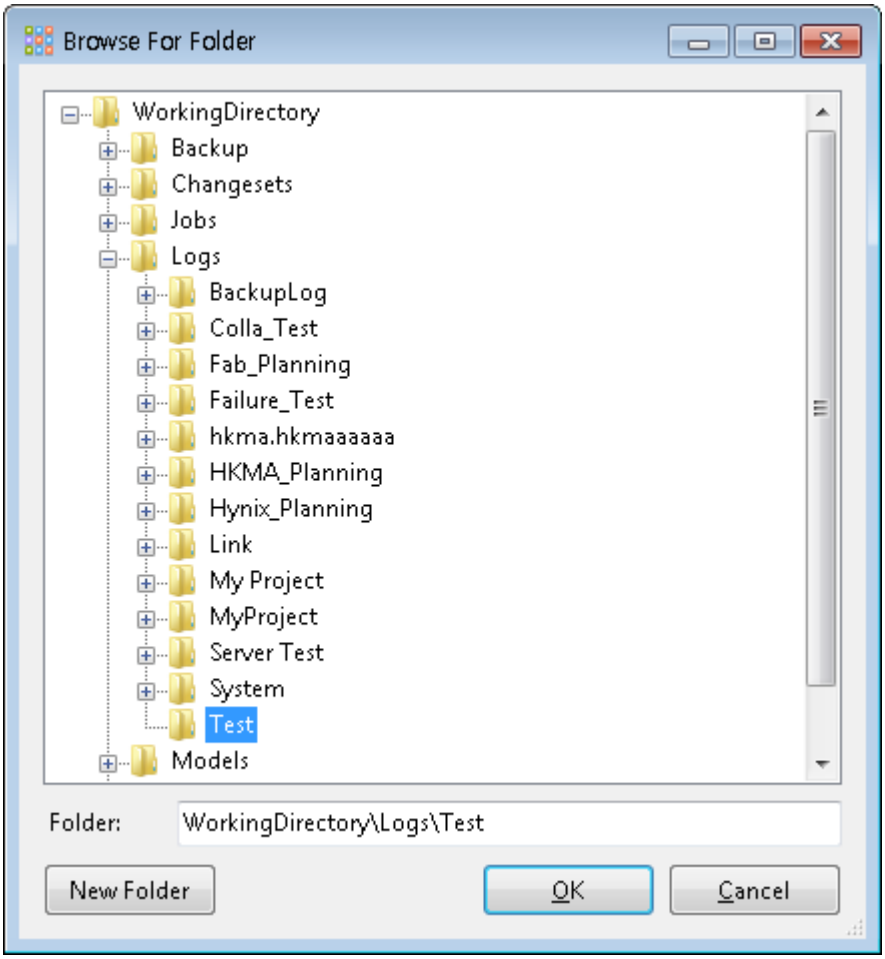


- **Project** : Specify the Project that contains the Model file to be executed by the Model Task and dll file information. When you click the drop-down list, a list of selectable projects appears.
- **Model file** : Specify the model file to be executed by the Model Task in the Project. If there are several model files in one project, only one model file in the list should be selected.



- **Model dll file** : Specify the dll file to run the model. You should upload the file to the server in advance.

- **Configuration file** : The configuration file that Model refers to. User can assign the log Key and folder of the Model execution log files through the configuration file.
- **Log dir** : Specify the folder where the job execution log is to be saved. By default, this is selected in the Logs folder of the project. You can select a folder by using **[...]** button on the right side when changing based on the **[Working Directory / Logs]** folder.



- **Additional run count** : Set a value when you want to run the same model more than once. The default value is 0. If a value is set, Arguments are created to select the setting file to be used for each run.
- **Collaboration Count** : It is displayed when a Job Type of $cola type is selected. In order to select the Job Type to be the Collaboration target when running the Main Model, Arguments are created to set Job Types for each run.

- **Disallow Concurrent Execution** : Enabling this option dose not allow MOZART Server to execute identical Jobs simultaneously.

6. Once the Job Type is selected from Basic tab, the list of arguments that could be configured for the corresponding job type can be seen through Parameters tab. The value of the arguments could be added through this tab.

> Unlike job types such as sending e-mail, running program or user specified job types, Arguments of the Model included from Basic tab can be seen if job type is either Model Task or Collaboration Task. Parameters that is displayed at the top part of the following figure are Input Arguments that are separately created in Model and extendedProperties at the bottom are Arguments that are automatically added in order to provide options about Model's execution and post-processing. For more information of the corresponding Argument, refer to Extented Arguments.

Modifying Job

1. Select a target server to inquire from Server Explorer.

2. Double-click Jobs node of the target server to activate Job inquiry window.

3. Select a job to modify from the list.

4. Click **[Edit]** menu from the top menu bar.

5. Modify each item in the same way that is used to edit in "***Registering a Job***".
+ Please keep in mind that the changes made in the parameters of the job after trigger is created will not be applied to the trigger configuration automatically.

Deleting Job

1. Select a target server to inquire from Server Explorer

2. Double-click **Jobs node** of the target server to activate Job inquiry window.

3. Select a job to be deleted from the list.

4. Click **[Remove]** menu from the top menu bar.

> In order to delete a Job, the trigger of the job should be deleted first. If you try to delete a job while the trigger exists, a warning message saying the job cannot be deleted because trigger exists will be displayed.

How to configure Config file

Config file can be used to assign the location where the Model execution logs to be stored. The following explains how to configure Config file.

1. First, create a Config file for the Model. Add the configuration through normal text file(notepad) and save the file using ".config" file extension.

2. The following is the example. The lines should be included as xml code and a key has to be added for <appSettings> session.

Usable Keyword

- #log-dir : Key designating a folder where Job execution log is saved.
  + How to configure : Create a folder to save the logs as absolute path or a relative path under Working Directory/Logs folder in the server.

Example :

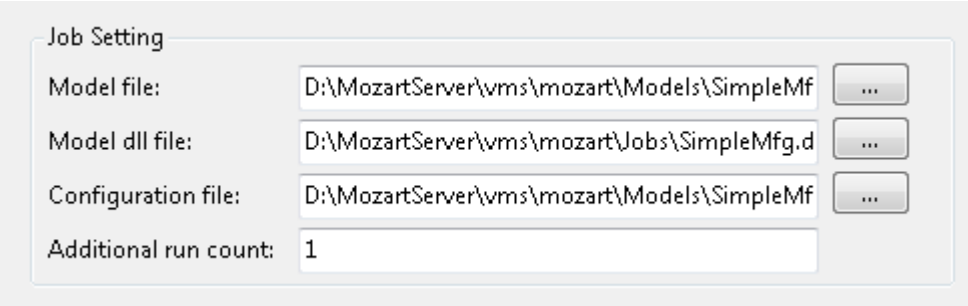
```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <appSettings>
        <add key="#log-dir" value="AModelLog"/>
```

```
    </appSettings>
</configuration>
```

If Keyword is written like the above example, log file is created in Working Directory/Logs/AModelLog folder.

3. Upload the configured file into a specific folder in Server. Normally it is saved in the folder where Model file is saved.

4. When Job is configured, Config file also should be configured.



5. If the configuration is done like the figure above, Task and Persist logs for the job will be created under the assigned folder. The log key of TSK Log file is the input value of "model-name" Argument and if log key is not configured the default key will be "TSK".

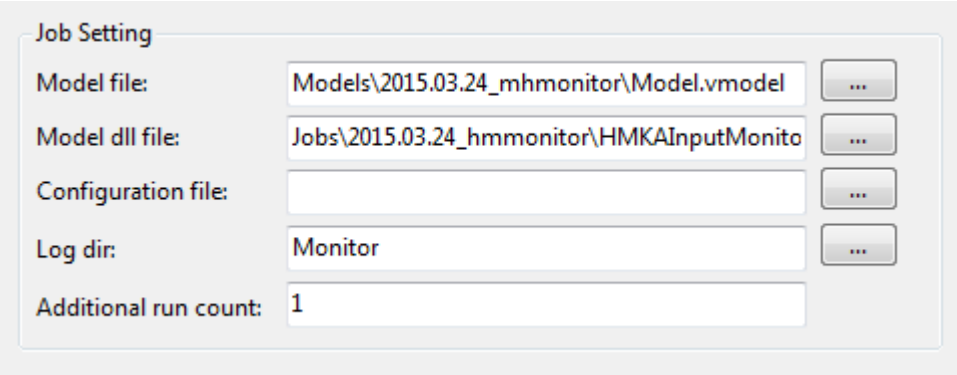
How to Configure more run

If a Model needs to be executed several times through a single Job execution, More Run configuration is used. For example, when a Model executes after downloading data on a particular time and Model has to execute again with the results from the previous Model, More Run can be used to perform this task.

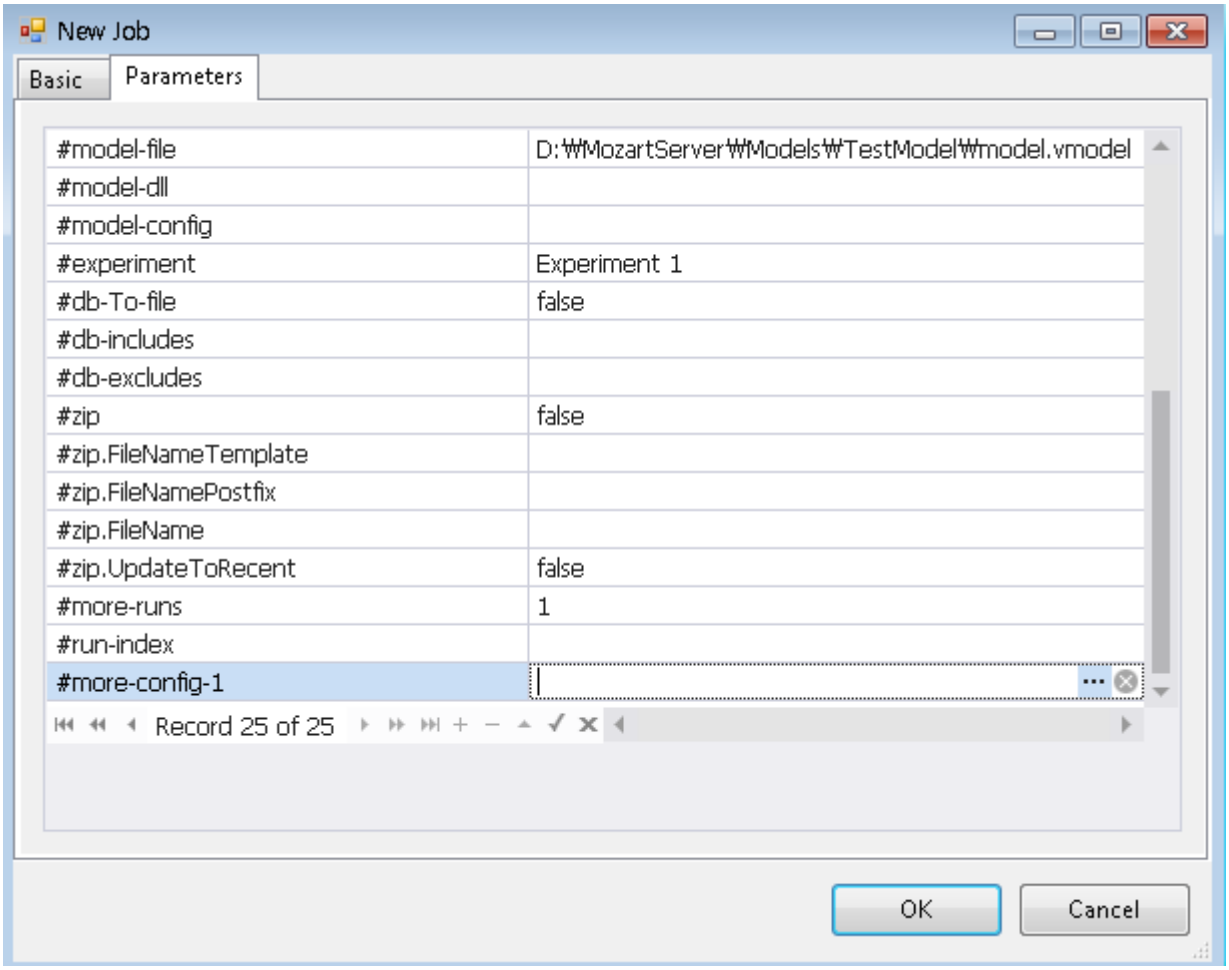
This configuration can be done when Job is configured. The following shows how to configure more run.

1. Set job type from to either $model or $cola from Basic tab. (More Run can be used from these two job types).

2. Configure Additional run count that is displayed on Job Setting when Job Type is selected. If 1 is configured, Model is executed one more time. So total number of executions are 2.



3. In order to configure argument that is used during additional run, move to Parameters tab. Extended argument can be seen at the last part of extendedProperties. This argument is used to set the parameter configuration for the additional run. Like the following figure, configure #more-runs = 1 and #more-config-1 will be created.

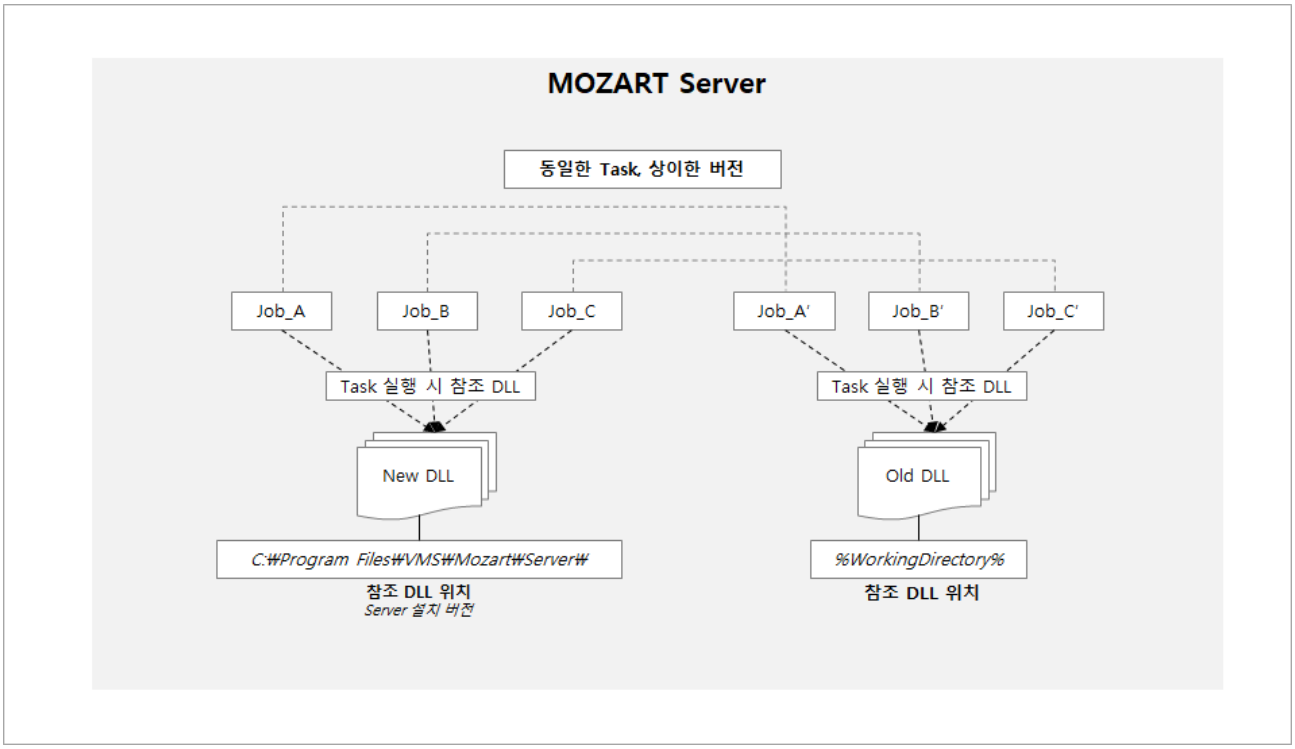


4. When **[...]** of #more-config-1 in the above figure is clicked, a window to write the values for the arguments from Input Arguments included in the Model. In here you may include the input argument values to use during more-run and modify the arguments of Extended argument required to be changed during additional Model execution. For instance, if you do not want to write Output Data to DB, you can deactivate #save-database option.

Server DLL 버전과 상이한 모델 수행하는 방법

Server를 운영하게 될 시 현장에 따라 안정적으로 수행이되는 Task의 DLL 버전들이 각기 다릅니다. 안정적으로 운행이 되고는 있지만 S/W의 특성상 신규 기능 추가 등에 사유에 따라 제품 업그레이드가 불가피한 경우들이 많습니다. 그러나 제품 업그레이드를 하게 될 시 현재 운영중인 Task DLL 및 Model과의 호환성이 안 맞을 수도 있습니다. 제품 업그레이드 이전 까지는 호환성의 문제를 확인하기 어려운 부분들이 많으며 호환성을 확인하기 위해 제품 업그레이드에 소요되는 시간, 또한 문제가 발생하여 다시 호환이 잘 되던 버전으로 Rollback에 소요되는 시간 등, 즉각 대응의 문제와 비효율적인 시간 소모들이 많습니다.

MOZART Server에서는 위에 명시된 발생 가능한 문제들을 예방 및 완화를 하고자 Server 머신에 설치된 MOZART Library DLL과 Task DLL의 버전이 상이하여도 Task가 잘 수행이 되도록 구성이 되어 있습니다. 상이한 버전의 Task를 수행하기 위해서는 MMC를 통해 사용자가 설정을 해야 합니다. 아래 그림은 MOZART Server에서 설치된 Server DLL과 상이한 버전의 Task가 수행되는 방법에 대한 기본 개념도입니다.



상기의 그림과 같이 구성을 하려면 Server 머신에는 **2018.2.113.0** 버전 이상의 MOZART Server가 설치 되어 있어야 합니다. 일반적으로는 Task DLL 및 Model를 Server에서 수행하기 위해서는 Server 설치된 DLL과 동일한 버전의 Client에서 Task DLL 및 Model 파일이 생성이 되어야 합니다. 그러나 Client의 변경 없이 상이한 버전의 Server에서 Task DLL 및 Model를 수행하기 위해서는 수행하고자 하는 Task DLL의 호환되는 버전의 Server DLL를 WorkingDirectory에 생성을하여 MMC에서 해당 Task 및 Model이 WorkingDirectory에 위치한 Server DLL를 통해 수행 할 수 있도록 설정을 하면 됩니다. 다음은 MMC를 통해 설정하는 방법에 대해서 기술합니다.

> MMC를 통해 설정하기 이전 Task DLL 및 Model의 Host 역할을 할 Server DLL 파일들이 WorkingDirectory에 옮기는 사전 작업이 필요합니다. 예를 들어 현재 Server 머신에 설치된 버전이 2018.2.113.0 버전이고 Task DLL 및 Model이 수행되던 버전이 2017.1.108.0 이었을 경우, 2018.2.113.0 버전 MOZART Server 설치 전 2017.1.108.0 Server 폴더 (예: C:\Program Files\VMS\Mozat\Server)를 WorkingDirectory에 복사를 합니다.

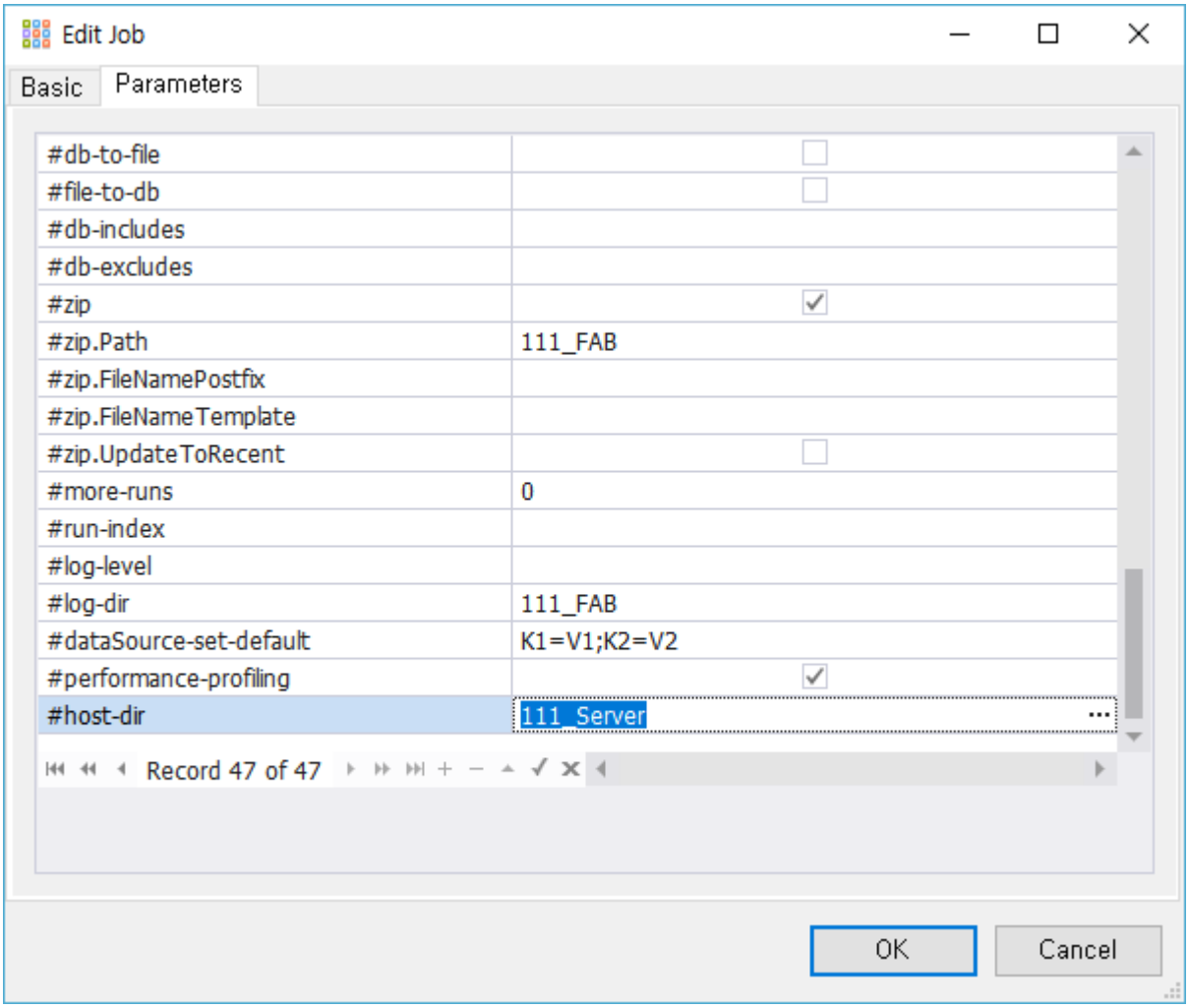
MOZART Management Console를 통해 설정하는 방법

1. Server Explorer에서 대상 서버를 선택합니다.

2. 대상 서버의 Jobs 또는 Triggers 노드 더블 클릭하여 Jobs/Triggers 창을 활성화 합니다.

3. Task의 Host가 변경이 필요한 Job 또는 Trigger를 선택합니다.

4. Job을 선택하였으면 [Parameter] 탭으로, Trigger를 선택하였으면 [Target Job] 탭으로 이동을 합니다.



Get Started

5. Argument 리스트에서 "**#host-dir**"를 찾아 **[...]** 버튼을 클릭하여 Task 수행 할 Host의 경로를 지정합니다.

6. **[OK]** 버튼을 클릭합니다.

Trigger Management

Trigger is Job Scheduler's managing component that is used to define execution target (Job) and execution condition. Multiple triggers can be registered depending on the purpose of the job. For example, if one same job has to be executed at 7:00 on every Monday and at 21:00 on every Friday, this job requires two triggers to perform these events. There are two conditions for the trigger. A condition dependent on time is to trigger on a suggested period or cycle and a condition dependent on event is to trigger during a specified event.

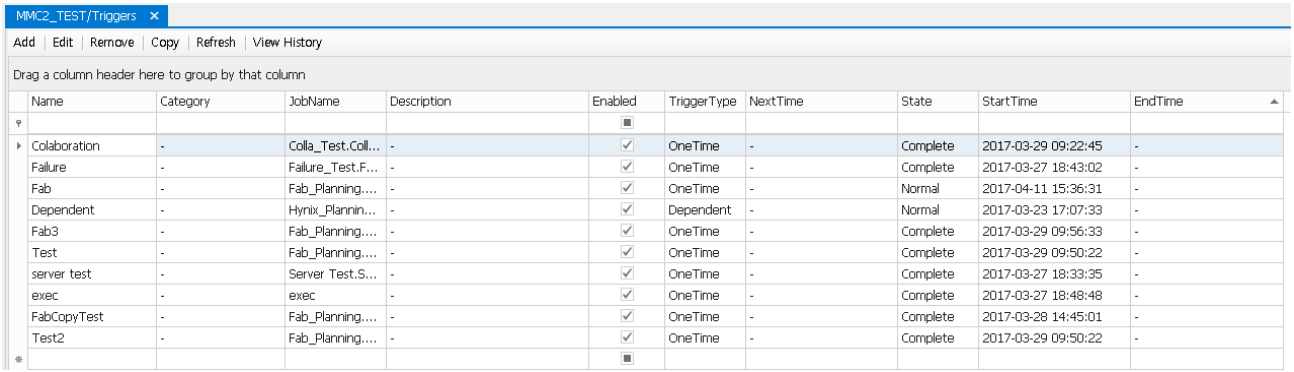
- **How to register/modify a Trigger** : This explains about how to register a new Trigger and how to modify the contents of the existing Trigger.
- **How to Delete a Trigger** : This explains about how to delete a registered Trigger.
- **How to Copy a Trigger** : This explains on how to copy a registered Trigger.

Triggers UI

This section is the description of Triggers UI in MOZART Management Console. In the Triggers tab, you can perform tasks such as register, edit, delete, and copy triggers. In addition, you can check the history of registered triggers, execution log, and so on. The below screenshot is the UI that appears when you activate the Triggers node.

Top Menu Bar and Trigger List

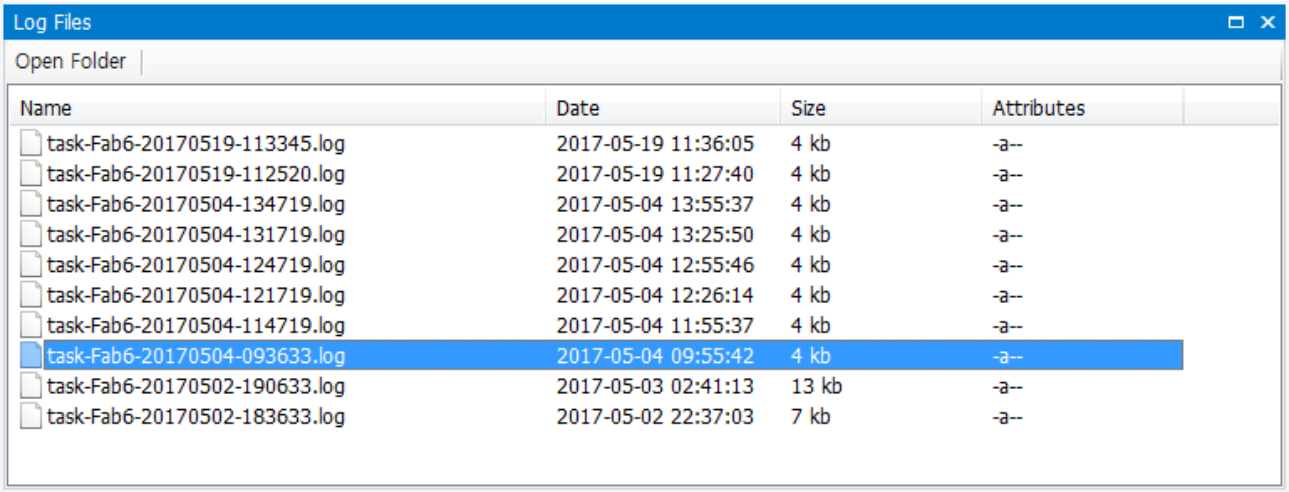
This UI is designed to register / edit / delete / search history of Trigger.



- **Add :** This menu is used to register a Trigger.
- **Edit :** This menu is used to edit a Trigger registered in the Trigger list.
- **Remove :** This menu is used to delete a Trigger registered in the Trigger list.
- **Copy :** This menu is used to copy a Trigger registered in Trigger list and add it to the list.
- **Refresh :** Button to refresh information such as Next Time / State / StartTime / EndTime and Execution History of Trigger.
- **View History :** This menu is used to view the history of triggers registered in the Trigger list. With View History, you can see the history that users have performed tasks.

Log Files Windows

This section is for looking up Trigger log files in \WorkingDirectory\Logs\[Project Name]. When log files are included in the search range of Search Option of Execution History, the list of files is displayed. In the Log Files section of Triggers, the 10 most recently opened files are displayed. To view the entire list, you can activate the Logs tab by clicking **[Open Folder]**.



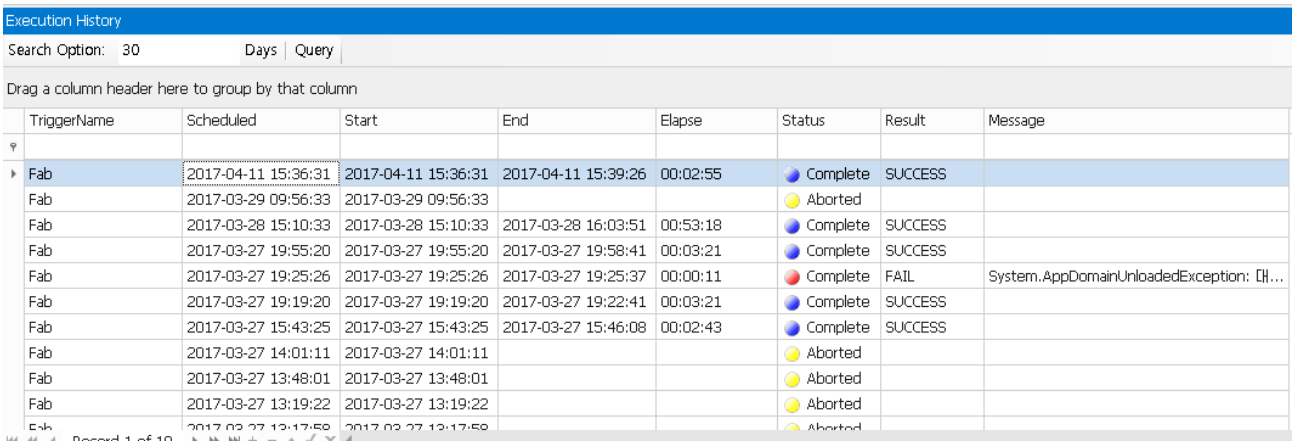
Execution History Window

The Execution History window displays the monitoring information of the selected trigger according to the search condition. This information can also be checked on the Monitoring

node. You can check it by the following steps.

1. Enter the number of days in Search option. The search condition displays the last N days of information from the current time. (e.g. the last 10 days from now -> enter '10' in Search Option)

2. Click **[Query]** button



TriggerName	Scheduled	Start	End	Elapse	Status	Result	Message
Fab	2017-04-11 15:36:31	2017-04-11 15:36:31	2017-04-11 15:39:26	00:02:55	Complete	SUCCESS	
Fab	2017-03-29 09:56:33	2017-03-29 09:56:33			Aborted		
Fab	2017-03-28 15:10:33	2017-03-28 15:10:33	2017-03-28 16:03:51	00:53:18	Complete	SUCCESS	
Fab	2017-03-27 19:55:20	2017-03-27 19:55:20	2017-03-27 19:58:41	00:03:21	Complete	SUCCESS	
Fab	2017-03-27 19:25:26	2017-03-27 19:25:26	2017-03-27 19:25:37	00:00:11	Complete	FAIL	System.AppDomainUnloadedException: [H...
Fab	2017-03-27 19:19:20	2017-03-27 19:19:20	2017-03-27 19:22:41	00:03:21	Complete	SUCCESS	
Fab	2017-03-27 15:43:25	2017-03-27 15:43:25	2017-03-27 15:46:08	00:02:43	Complete	SUCCESS	
Fab	2017-03-27 14:01:11	2017-03-27 14:01:11			Aborted		
Fab	2017-03-27 13:48:01	2017-03-27 13:48:01			Aborted		
Fab	2017-03-27 13:19:22	2017-03-27 13:19:22			Aborted		

> Triggers' Execution History is displayed by paging, and initially displays items in 10 units according to window size. The user scrolls down in increments of 10.

Trigger Execution Log Window

In the Trigger Execution window, you can check the execution time of the Trigger selected in the Execution History window step by step. For more information, please check **Monitoring**.

Trigger Execution Log

Action	Start	End	Elapse
PERSIST_IN	2017-03-28 15:10:54	2017-03-28 15:13:28	00:02:34
ENGINE_RUN	2017-03-28 15:13:30	2017-03-28 15:43:26	00:29:56
PERSIST_OUT	2017-03-28 15:44:22	2017-03-28 15:44:32	00:00:10

Adding Trigger

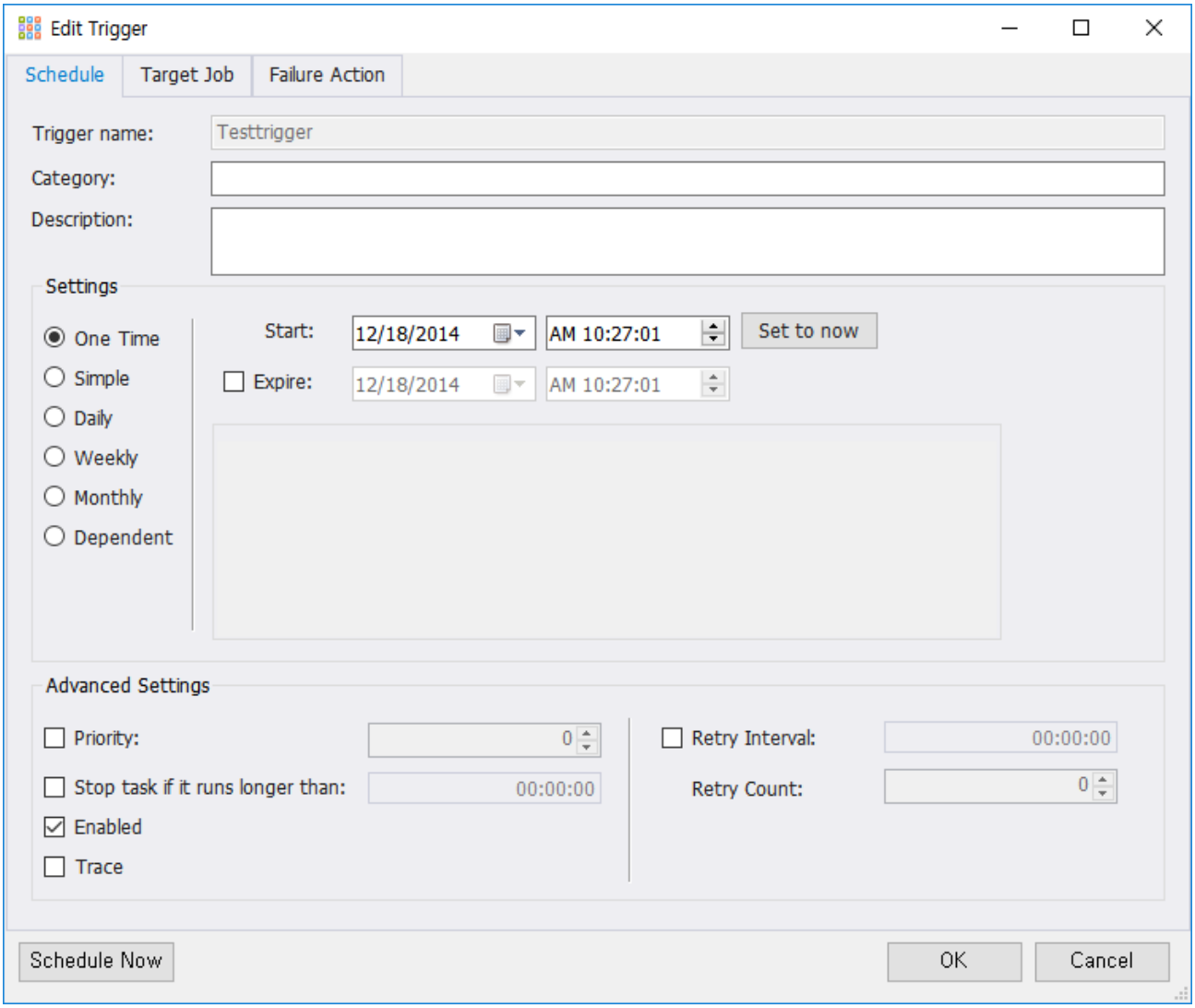
1. Select a target Server where Trigger is registered in Server Explorer.

2. Double-click on Triggers node to open trigger page.

3. Click **[Add]** from the top menu bar.

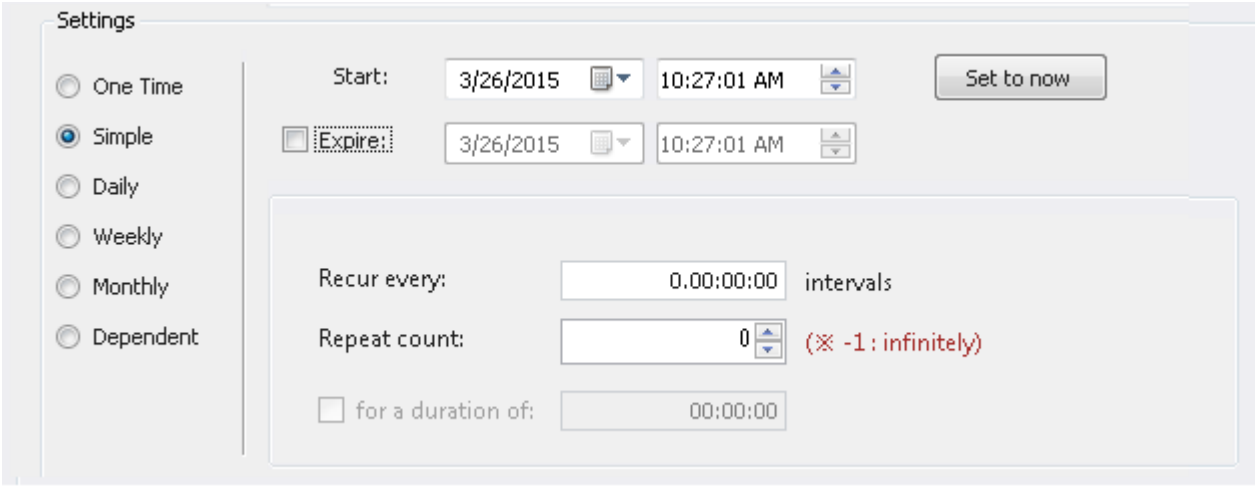
4. Enter information for Schedule, Target Job, Failure Action Tab on New Trigger dialog.

5. Enter information of Schedule Tab.



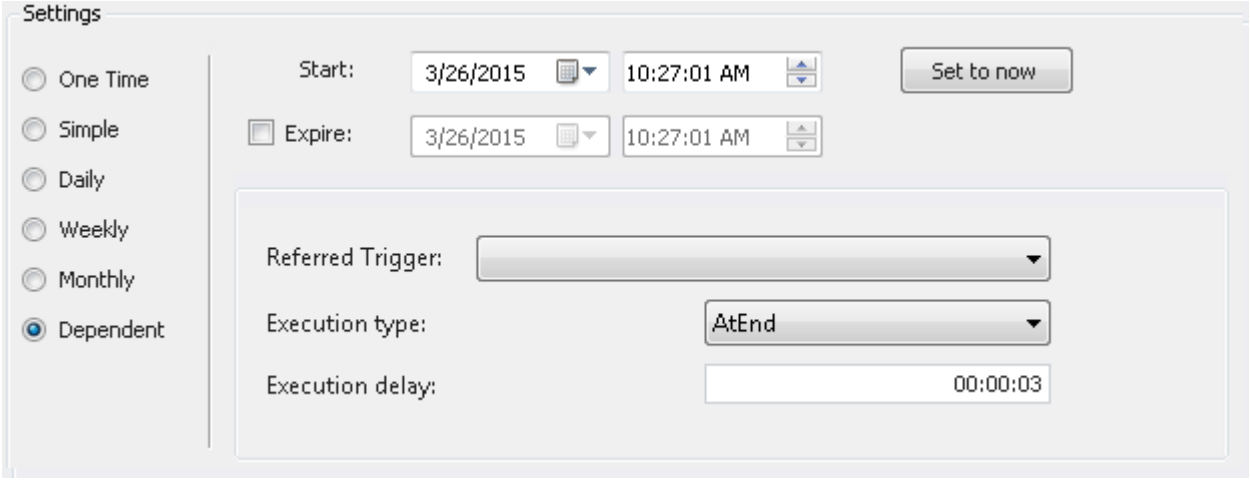
- **Trigger name** : Name of the trigger.
- **Settings** : Set trigger condition.
  - **One Time** : Trigger the job once at its starting time.

- **Expire Configuration** : If Expire check box is checked, Trigger is executed until the configured Expire time.
- **Simple** : Number and cycle of repetitions can be configured. Repeat cycles can be set from second to a day. If repeat count is set the event occurs on the condition set on Recur every.



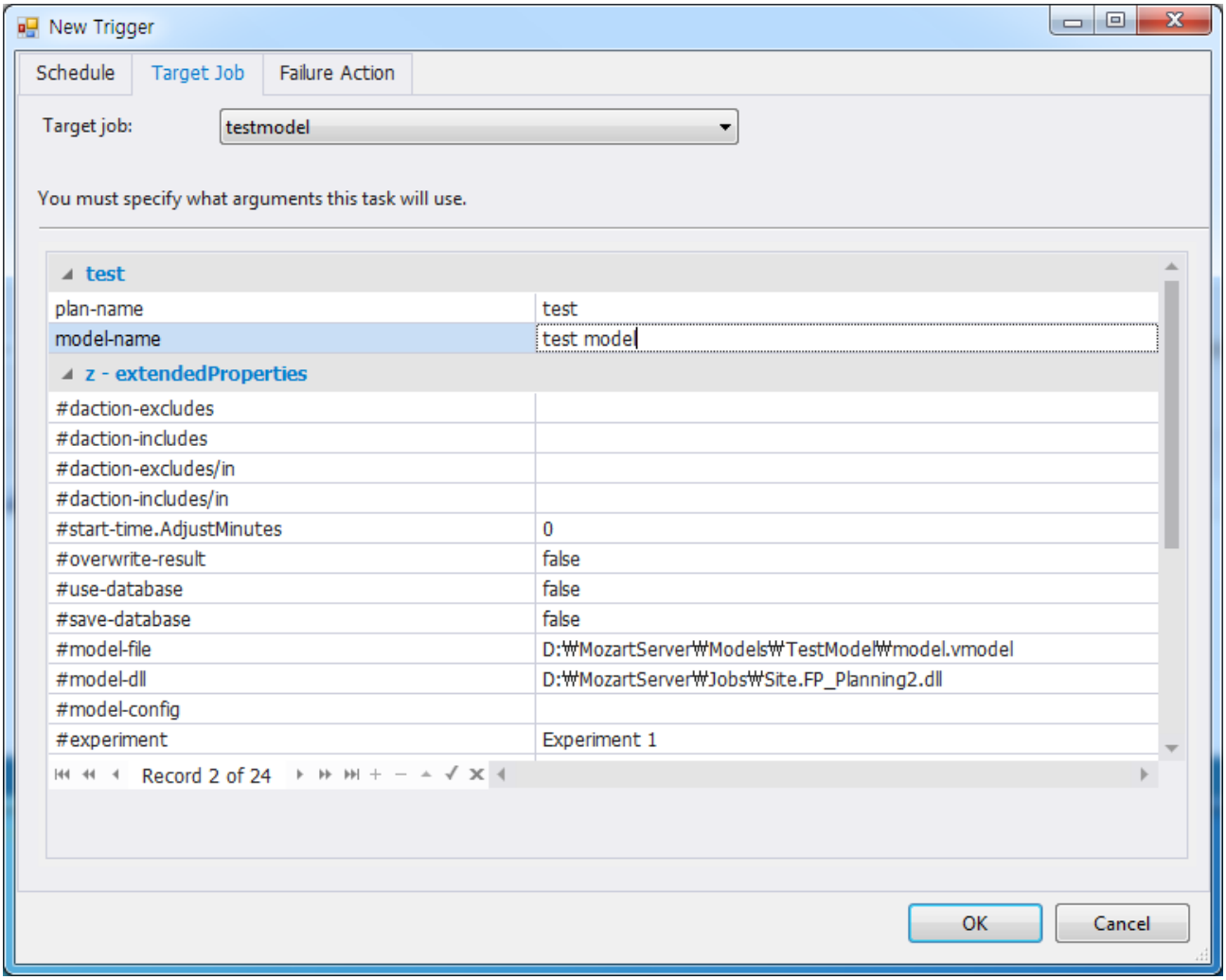
When repeat count is set to -1, the trigger is repeated for unlimited times. In this case, the option 'for duration of' will be activated. Check on this option and enter the time then the trigger event will occur from the interval set on Recur every on the start of each day until the configured duration.

- **Daily** : Schedule can be defined to repeat on a daily basis. Schedule can be made likewise as Simple Type.
- **Weekly** : Weekly Cycle can be configured. The day(s) to repeat can be designated.
- **Monthly** : Trigger can be set to a combination like (Month + Specific day) or (Specific Month + a day in a specific order of the Month).
- **Dependent** : This is an event based condition. A target trigger needs to be selected as a reference and conditions to start the trigger event needs to be set. (Start/End of target trigger) A trigger set as Dependent will be dependent on the target trigger.



- **Referred Trigger** : The target trigger to execute the corresponding trigger. Multiple triggers can be selected.
- **Execution type** : This defines when to trigger according to the condition of Referred Trigger.
  - **AtEnd** : Triggers when the target trigger of Referred Trigger is completed.
  - **AtStart** : Triggers when the target trigger of Referred Trigger starts.
  - **ReturnIfTrue** : Triggers when the target trigger of Referred Trigger returns true. For example, a job to return true can be used when a job needs to be executed when system data changes or when a trigger of Referred Trigger is to inquire a certain data and the condition is met. In this case, the Model Task Return value (true/false) needs to be set from the job triggered by Referred Trigger.
    + Example of How to use ReturnIfTrue Type
- **Execution delay** : Set the delay time when the condition in Execution type is met.
- **Advanced Settings**
  - **Priority** : When multiple triggers start simultaneously, this defines priority of execution order.
  - **Stop task if it runs longer than** : This option is to set the maximum run time of the trigger. If the trigger is performed longer than the configured max time, the trigger will be terminated by force.
  - **Enable** : Activate corresponding trigger
  - **Retry Interval** : Decides retry intervals when trigger is failed.
  - **Retry Count** : Decides retry counts when trigger is failed.

6. Enter information of Target Job Tab.



- **Target job** : The target job to trigger. Only the registered jobs can be selected.
- **Argument** : Input Parameter is configured according to needs when Target Job is executed. Value of parameter that is configured in a Job is used as a default. When a Job is executed, Argument value configured in Trigger is used.

7. Input Failure Action Tab Information . It has the same input format as Target Job's.

- **Failure Action** : A job to execute when target job fails to execute by adding new function or a job as a backup for complement.
- **Argument** : Configures the input parameters of the job in Failure Action. As same as in Target job tab, the default value is the parameters set in job.

Modifying Trigger

1. Select a target server to modify trigger in Server Explorer.
2. Double-click on Triggers node to open trigger page.
3. Select the trigger from the list to modify.
4. Click **[Edit]** from the top menu bar.
5. Change the information as done through ***"Registering Trigger"*** section.

How to Copy Trigger

Trigger Copy

If you want to compare the results of two triggers by changing the part of arguments or if you need to change the name of an already registered Trigger, you can copy Triggers as shown below.

1. Select a target server to be checked in Server Explorer.

2. Double-click Triggers node to open trigger page.

3. Select a trigger from the list to be copied.

4. Click **[Copy]** menu from the top menu bar.

5. Enter the name of the trigger to be copied.

> Please note that an error will occur if the entered name of the copied trigger already exist in the list. Be advised not to use any of the name already in the list.

Deleting Trigger

1. Select a target server to be checked in Server Explorer.

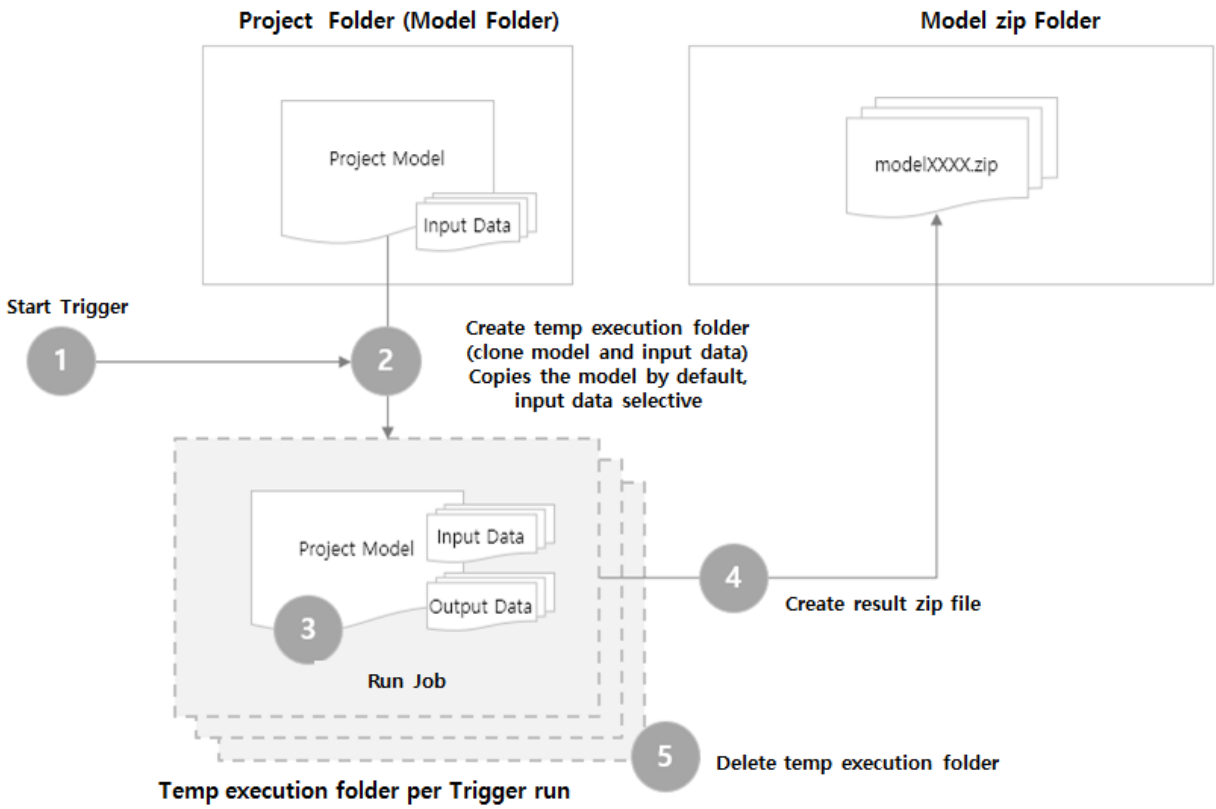
2. Double-click Triggers node to open trigger page.

3. Select a trigger to be deleted from list.

4. Click **[Remove]** menu from the top menu bar

Copy Model and Data to Temp Folder to Run Trigger

This is a guide to run the model of the trigger not from the `Project` folder, but from a temporary execution folder.



Argument to Create and Use Temporary Folder to Run Trigger

The following table lists the extended arguments in MMC to create a temporary folder containing the vModel file and input data copied from the `Project` folder, and run the Trigger.

Argument	DataType	Description

#use-run-dir	boolean	Indicates whether to use the temporary folder to run Trigger. (Default: false) The temporary folder creates under `WorkingDirectory` > Execution > *[Trigger Name]* > *[Executed Time]*
#max-run-dir	int	Specifies the number of temporary folders to create. (Default: 2) When the trigger run finishes the temporary folder is deleted. The value in the argument indicates the number of the folders to be kept. (i.e If the value is 2, then the two temporary folders of the recent executed trigger is left.
#use-parent-path	boolean	Indicates whether to use the temporary folder of the reference trigger. (For dependent trigger only)

When #use-run-dir = true, a temporary folder is created when the trigger executes. The location of the temporary folder is as follows.

Temp execution folder location: *WorkingDirectory\Execution\[Trigger Name]\Temp\[YYYYMMDD-HHmmss-random string]*

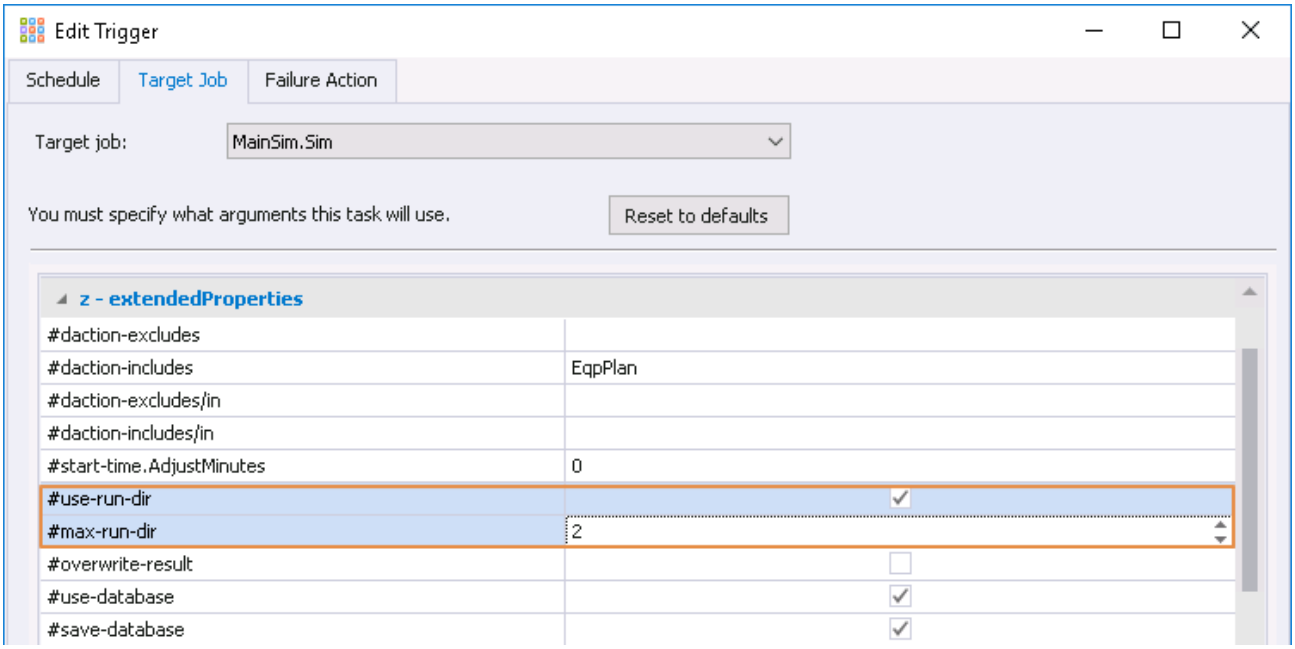
> **Note**:
> Please close all temporary folders when the trigger is executed. Leaving a folder opened may cause the folder not to be deleted especially when the folder to be deleted is opened.

How to Use

The arguments can be set from either Job or Trigger setting window of MMC.

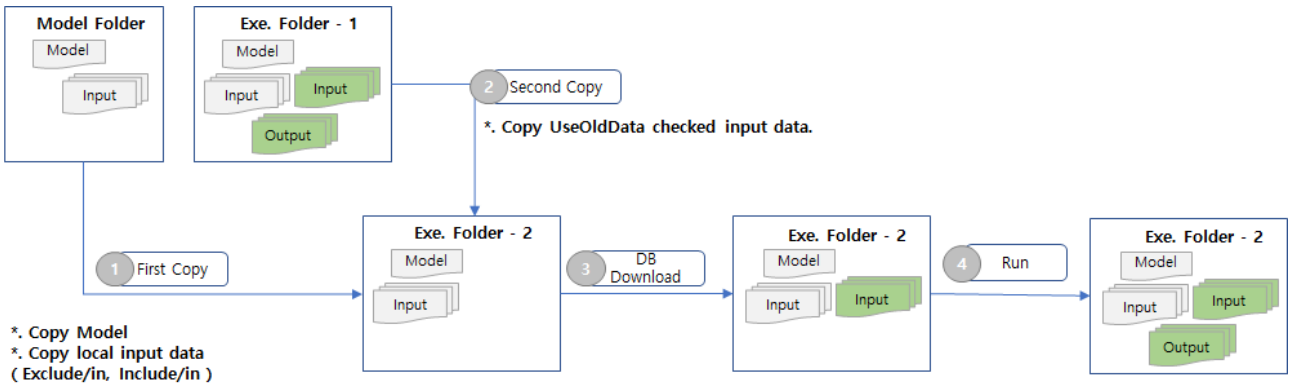
How to Set from Jobs/Triggers

1. Run **MozartManagementConsole2.exe**. The file is located in the path where MOZART client is installed (i.e *C:\Program Files (x86)\VMS\Mozart\Server\bin*).
2. Select a server node from Server Explorer. Then, right click and select **[Connect Server]**.
3. Type the log-in ID and password to **[User ID]** and **[Password]** box and then click **[OK]** button to connect to the server.
4. Right-click on **[Jobs]** or **[Triggers]** node and select **[Open]** or double-click **[Jobs]** or **[Triggers]** node from Server Explorer to open the window.
5. Select a job/trigger from the list and double-click to open **[Edit Job]/[Edit Trigger]** dialog.
6. Go to **[Parameters]** and scroll down until you see **[#use-run-dir]** and **[#max-run-dir]**.
7. Check **[#use-run-dir]** to create temporary folder and type a number to **[max-run-dir]** box to decide the number of temporary folders to maintain.



Operation Structure

When `#use-run-dir` is `true` the vModel and input vData files in the `Project` folder is copied to the temporary folder of the trigger when the trigger is executed. The vData files copied to the temporary folder are the files that are not to be donwloaded from the database. After the files are copied, the model executes from the created temporary folder according to the arguments set in trigger setting. The data to be downloaded from the database is stored in the input folder of the temporary folder.



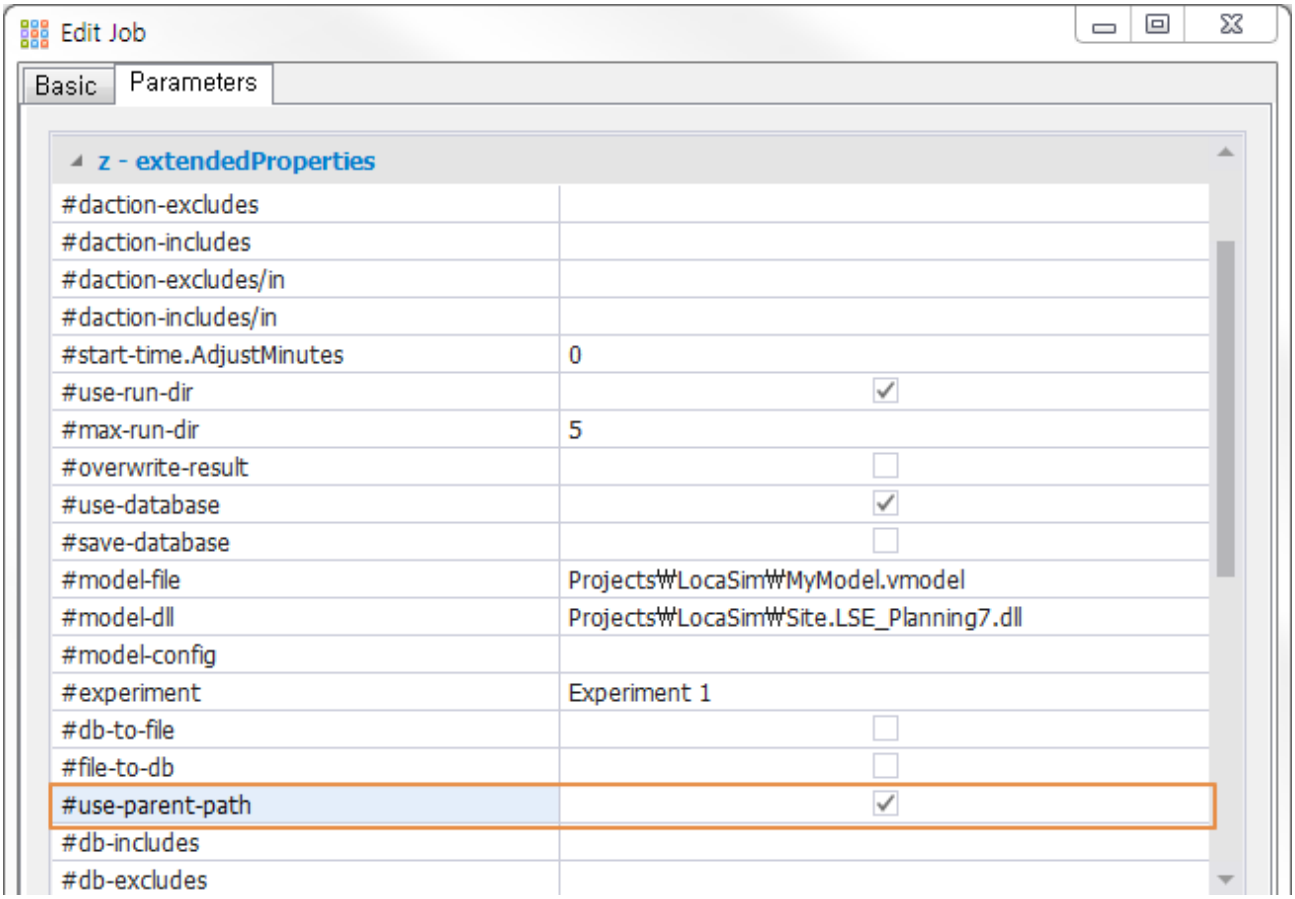
How to Set Dependent Trigger to Refer the Temporary Execution Folder of Parent Trigger

When the option to use the temporary execution folder for the trigger each trigger will have a dedicated temporary folder of its own. This is same for the dependent trigger as well. However, there are cases where the dependent trigger needs to refer the execution result from its reference (parent) trigger.

For instance, if the task of the dependent trigger is to save the result from its parent to the database, then the dependent task needs to get the data result from the parent.

How to Use & Example

Set `#use-parent-path` option to `true` from the trigger settings in MMC, in order to use the model and data in the temporary execution folder of the parent trigger. This option is only effective for dependent trigger only and will not work on independent triggers.



#file-to-db Argument Example

The following example is a dependent trigger using #file-to-db argument saving the experiment result of the parent trigger to the database.

Scenario

- Trigger A : Main Trigger
- Trigger B : The dependent trigger to save the essential result data of Trigger A to the database. (Starts task after Trigger A finishes)
- Trigger C : The dependent trigger to save the monitoring result data of Trigger A to the database. (Starts task after Trigger B finishes)

Trigger Settings

1. Set `#use-run-dir = true` to Trigger A,B, and C.
2. Set `#file-to-db` to Trigger B and C (`#save-database` should be set as true in advance.) 2. Set Trigger B dependent to Trigger A from Basic, then go to Parameters and set `#use-parent-path = true` from Trigger B.

3. Set Trigger C dependent to Trigger B from Basic, then go to Parameters and set `#use-parent-path = true` from Trigger C.
4. Run Trigger A and see if Trigger B and Trigger C saves the result from Trigger A to the database.

Other Remarks

The operation of `#use-run-dir` and `#max-run-dir` works differently depending on the following arguments set in the trigger setting.

- **db to file Job :** When set `true`, input data is downloaded to `Project` folder instead of the temporary folder.
- **file to db job :** When set 'true' the output data from `Project` folder is uploaded to database.
- **overwrite job :** This argument cannot be used with `#use-run-dir` argument. When set `true`, the model in the recent made temporary folder is executed and no additional temporary folder is created

Run Trigger from Another Domain/Execution DLL Versions

When the version of the MOZART Server is upgraded, the stability of the Job/Trigger execution from the latest version is not guaranteed. The stability issue may require to operate the Job/Trigger of the stabled version until the stability of the latest version is guaranteed.

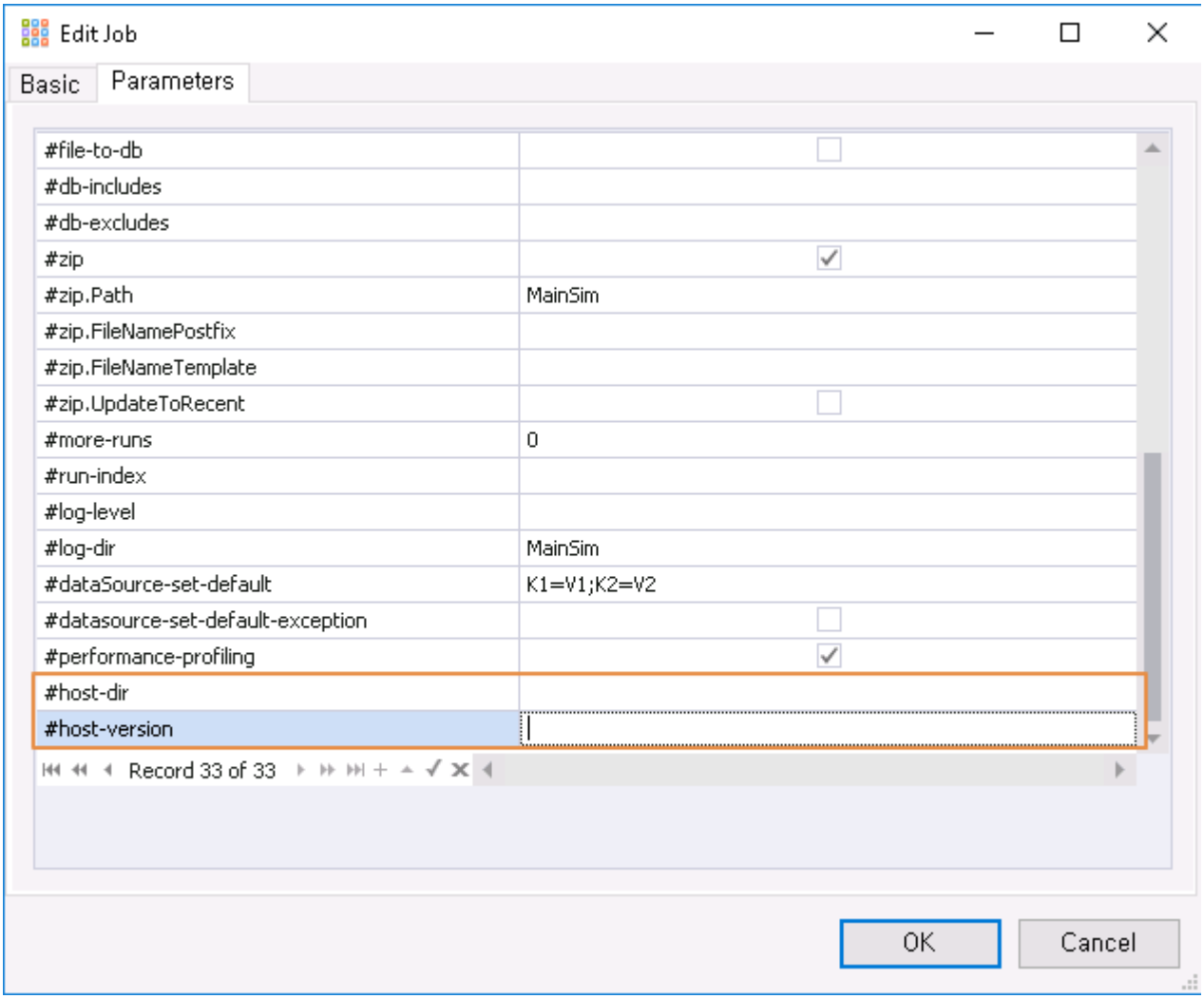
In MMC, users can set the Job/Trigger to run from different versions of domain library and execution DLL files other than from the latest installed version.

The following table shows the name of the extended arguments and descriptions that you can use from MMC to set the Job/Trigger to run from different MOZART versions.

Argument	DataType	Description
#host-version	string	The path of the domain library and execution dll files set from Execution Path in MOZART Configurator for Server. The input value is the version number of the assemblies for the Job/Trigger to refer.
#host-dir	string	**Relative Path**: The name of the folder in WorkingDirectory where the domain library and execution DLL files are stored. **Absolute(Full) Path**: Any location where the domain library and execution DLL files are stored. The full path must be typed in.

How to Use

This section explains on how to use #host-dir and #host-version arguments in MMC to set the version of domain/execution DLL files for the Job/Trigger to refer. You can use either one of them.



In order to use one of these functions, at least two different versions of domain/execution DLL files should exist in the machine where MOZART Server is installed.

#host-version

The steps to use #host-version are as follows:

1. Run **MozartManagementConsole2.exe**. The file is located in the path where MOZART client is installed (i.e *C:\Program Files (x86)\VMS\Mozart\Server\bin*).

2. Select a server node from Server Explorer. Then, right click and select **[Connect Server]**.

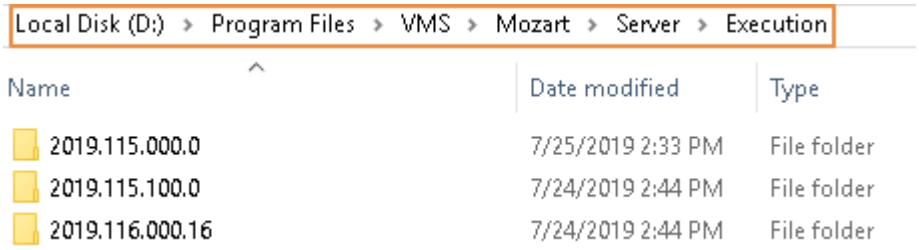
3. Type the log-in ID and password to **[User ID]** and **[Password]** box and then click **[OK]** button to connect to the server.

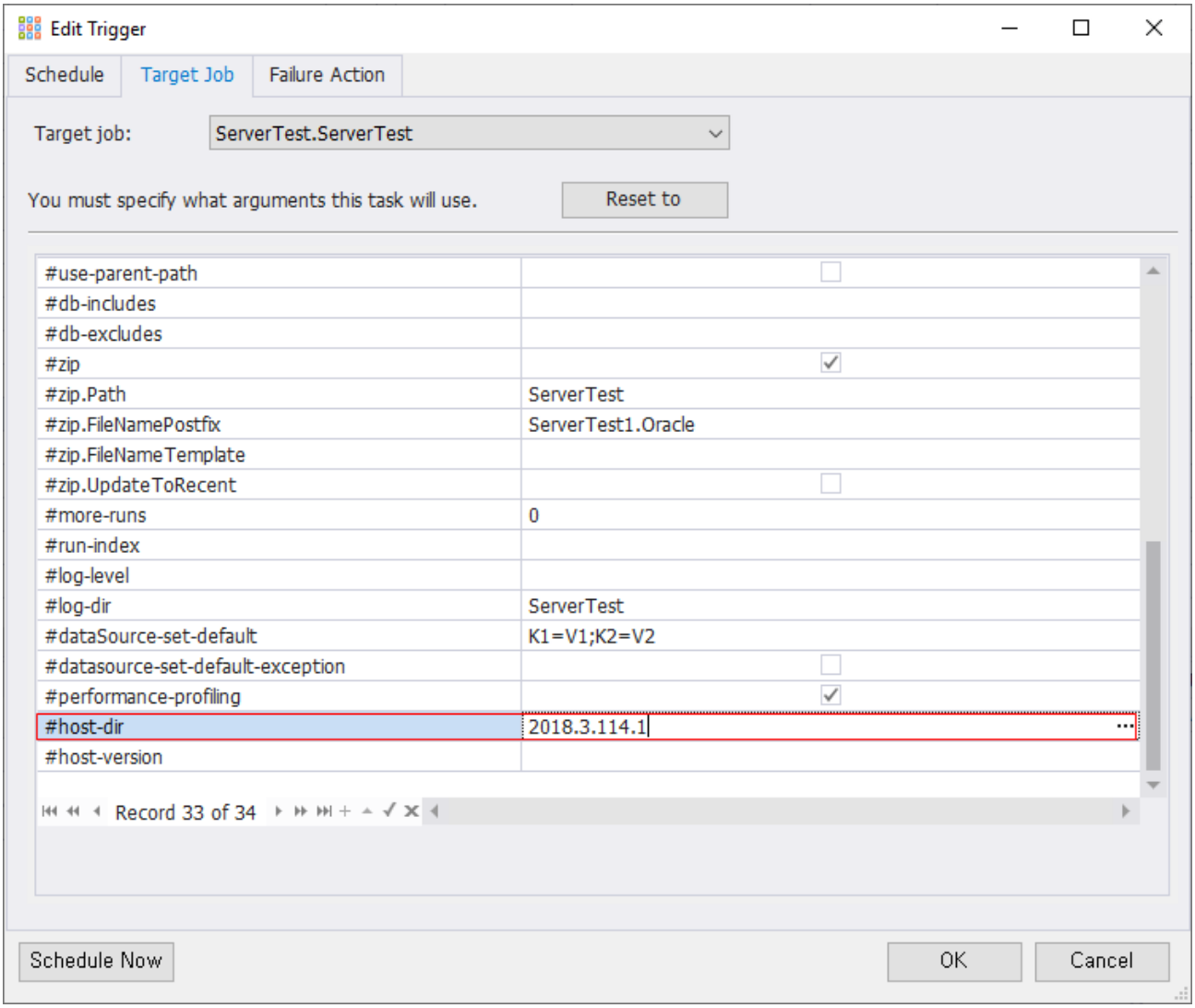
4. Right-click on **[Triggers]** node and select **[Open]** or double-click **[Triggers]** node from Server Explorer to open **[Triggers]** window.

5. Select a trigger from the list and double-click to open **[Edit Trigger]** dialog.

6. Go to **[Target Job]** and scroll down until you see **[#host-version]**.

7. Type the version number you want to run the trigger in **[#host-dir]** box. A folder having the version number as its name is created when domain library `mdz` file is installed.





8. Click **[OK]** button to save the changes and close the dialog.

#host-dir: Relative Path (WorkingDirectory)

The folder that contains the domain library and execution DLL files need to be placed in WorkingDirectory. Depending on the MOZART server versions, these files are located in different paths. The following lists the default path where the files are located.

- **2019.3.114.1 and below**: *%ProgramFiles% or %ProgramFiles(x86)%\VMS\Mozart\Server* or *\VMS\Mozart\Server* folder in the path assigned during MOZART server installation.
- **2019.115.000.0 ~ 2019.115.100.0**: *%ProgramFiles% or %ProgramFiles(x86)%\VMS\Mozart\Server\Execution\[Version]* or in the *\Execution\[Version]* folder in the path assigned during MOZART Configurator for Server installation.
- **2019.116.000.0 and above**: *%ProgramFiles% or %ProgramFiles(x86)%\VMS\Mozart\Server\Execution\[Version No]* or in the *\Execution\[Version]* folder assigned from Execution Path: in MOZART Configurator for Server.

The steps to use #host-dir are as follows:

1. Follow the steps 1~5 in #host-version.

2. Go to **[Target Job]** and scroll down until you see **[#host-dir]**.

3. Click **[...]** button in **[#host-dir]** box. Then, select the folder of the version to host the Job/Trigger execution in Browse For Folder dialog and click **[OK]** button.

4. Click **[OK]** button in Edit Trigger dialog to save the changes and close the dialog.

#host-dir: Absolute(Full) Path

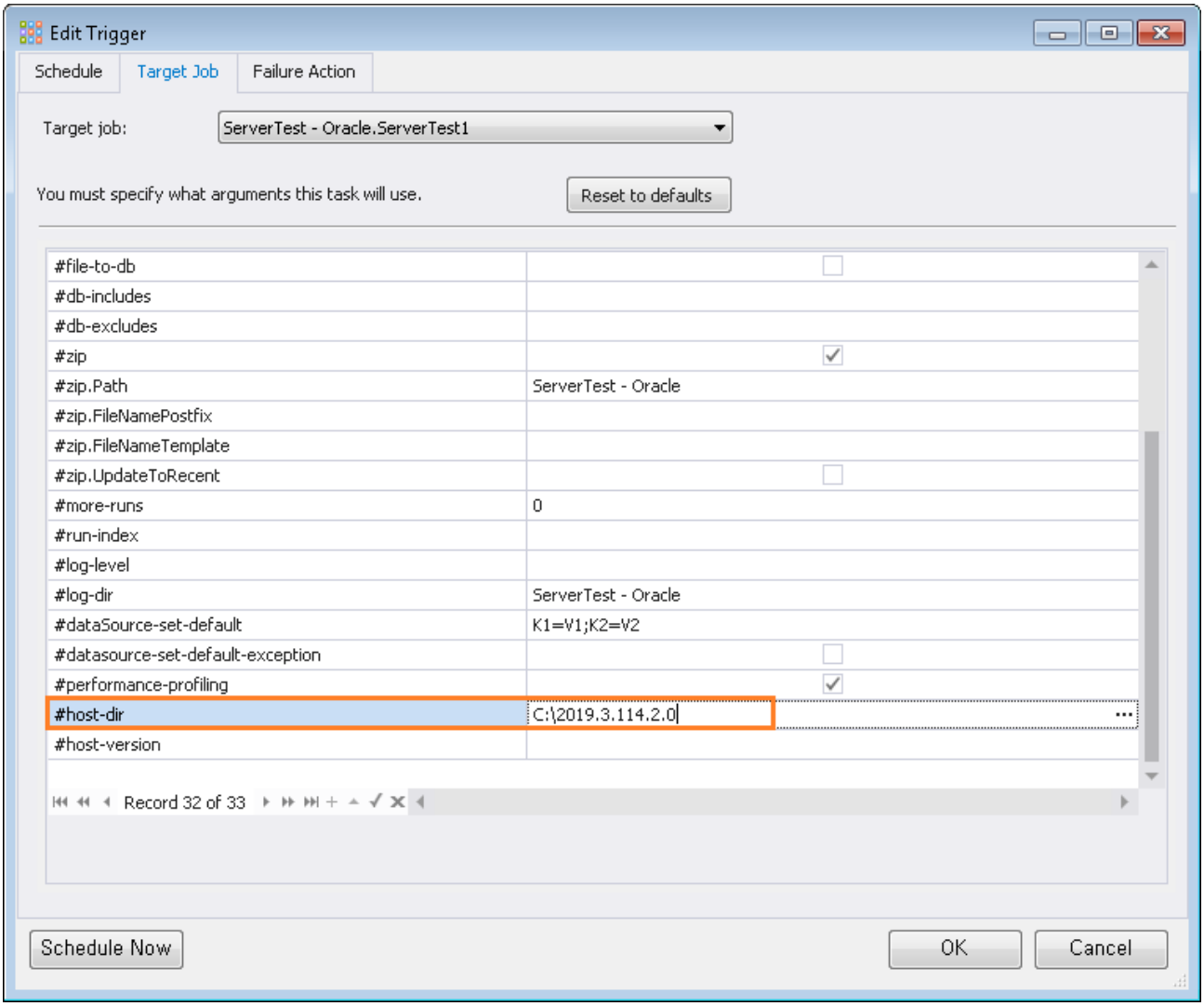
When using the absolute path to #host-dir, the specified folder is searched only in WorkingDirectory. However, when you set the full path to #host-dir, the folder containing the domain library and execution DLL files can be located anywhere that you prefer.

The steps to use #host-dir are as follows:

1. Follow the steps 1~5 in #host-version.

2. Go to **[Target Job]** and scroll down until you see **[#host-dir]**.

3. Type the full path of the folder where the domain library and execution DLL files are located in **[#host-dir]** box.



4.Click **[OK]** button in Edit Trigger dialog to save the changes and close the dialog.

Priority

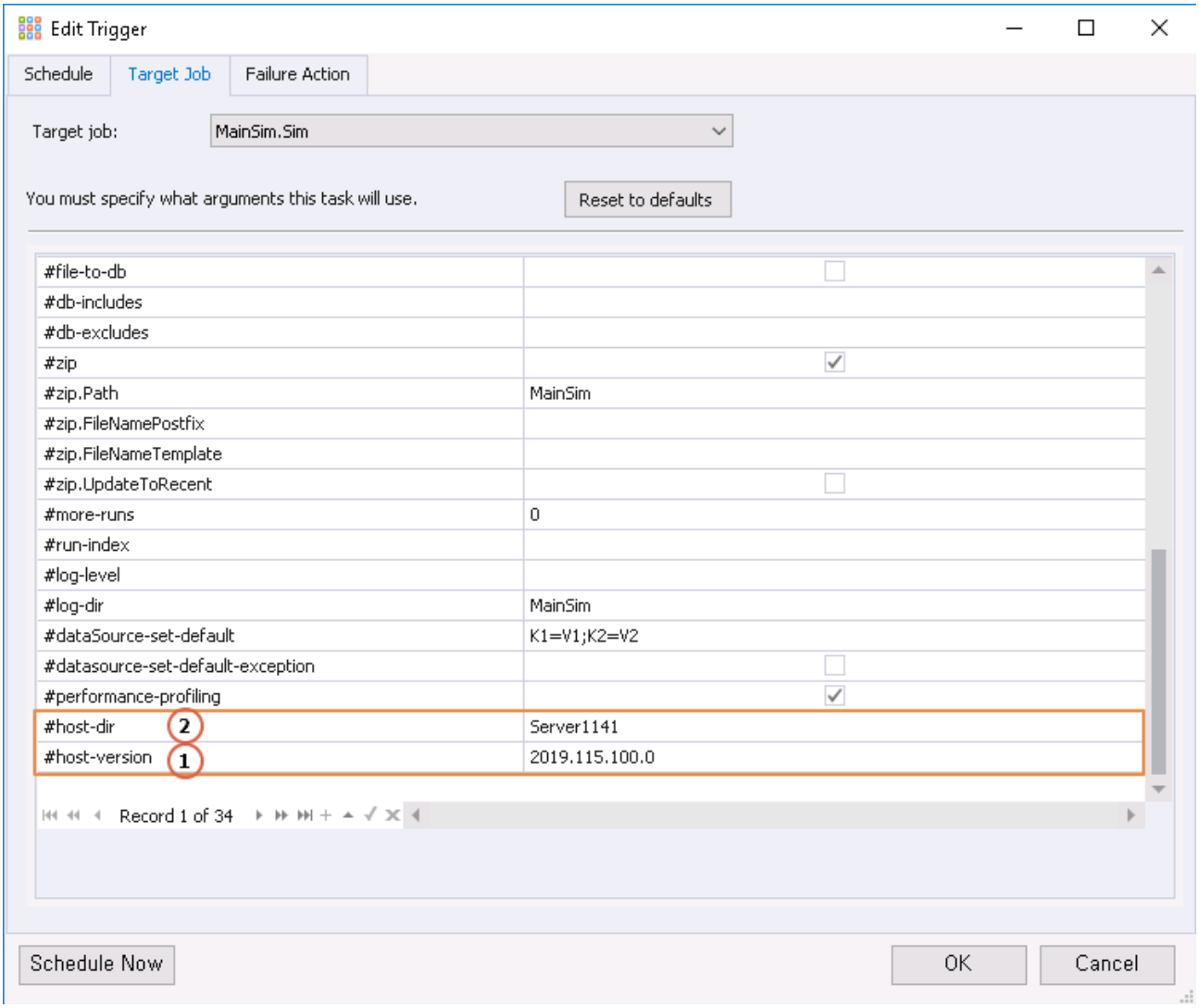
As mentioned above, to run Job/Trigger from a different version of domain library and execution assemblies, you only need to configure either #host-version or #host-dir.

When both #host-dir and #host-version have values, the server searches for the existence of the folder in the following order.

1. **host-version**
2. **host-dir (Absolute path > WorkingDirectory)**
3. The folder with the highest version number in *Execution* path.

Let us assume that both values are set in #host-version and #host-dir. If the folder with the name specified in #host-version exists, then the Job/Trigger runs hosted by #host-version. Otherwise, the Job/Trigger runs hosted by #host-dir. If both folders are not found or no values are set in both #host-version and #host-dir, then Job/Trigger runs hosted by the folder with the latest version name in *Execution* path.

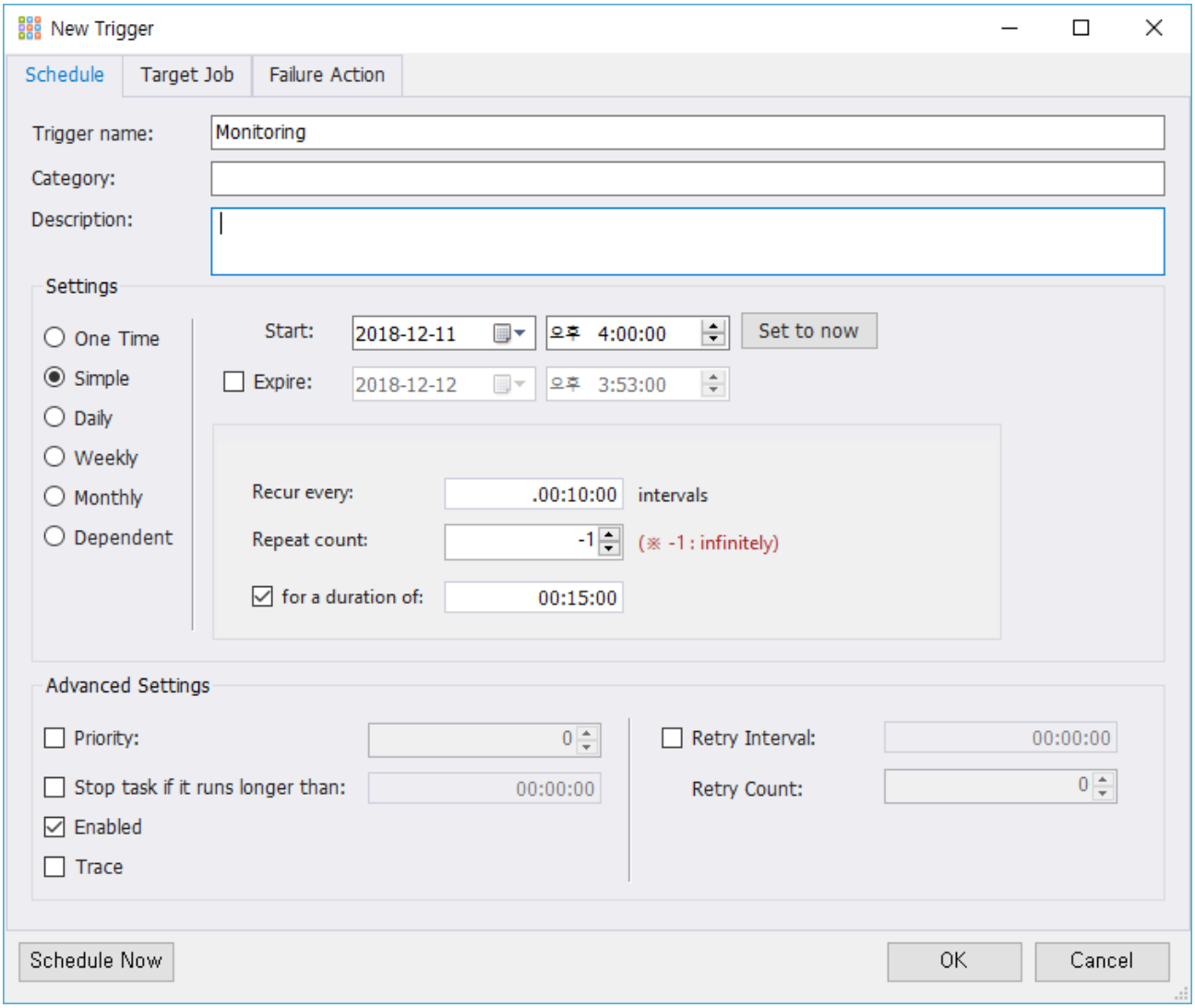
The following example has values set for both #host-version and #host-dir. The Job/Trigger will run using 2019.115.000.0 version DLLs if 2019.115.100.0 folder exists in *Execution* path.


Edit Trigger
Schedule
Target Job
Failure Action
Target job:
MainSim.Sim
You must specify what arguments this task will use.
Reset to defaults
#file-to-db
#db-includes
#db-excludes
#zip
#zip.Path
MainSim
#zip.FileNamePostfix
#zip.FileNameTemplate
#zip.UpdateToRecent
#more-runs
0
#run-index
#log-level
#log-dir
MainSim
#dataSource-set-default
K1=V1;K2=V2
#datasource-set-default-exception
#performance-profiling
#host-dir
2
Server1141
#host-version
1
2019.115.100.0
Record 1 of 34
Schedule Now
OK
Cancel

Dependent Trigger Sample

Dependent Trigger 의 한 유형으로 대상이 되는 Trigger 에서 수행한 Job 의 결과값을 확인하여 실행 여부를 설정하는 예를 통해 설정 방식을 설명합니다. 타 시스템과의 연계 실행을 위해 특정 DB 의 Table 을 사용하는 경우가 이러한 Dependent Trigger 를 사용할 수 있는 대표적인 예라고 할 수 있습니다.

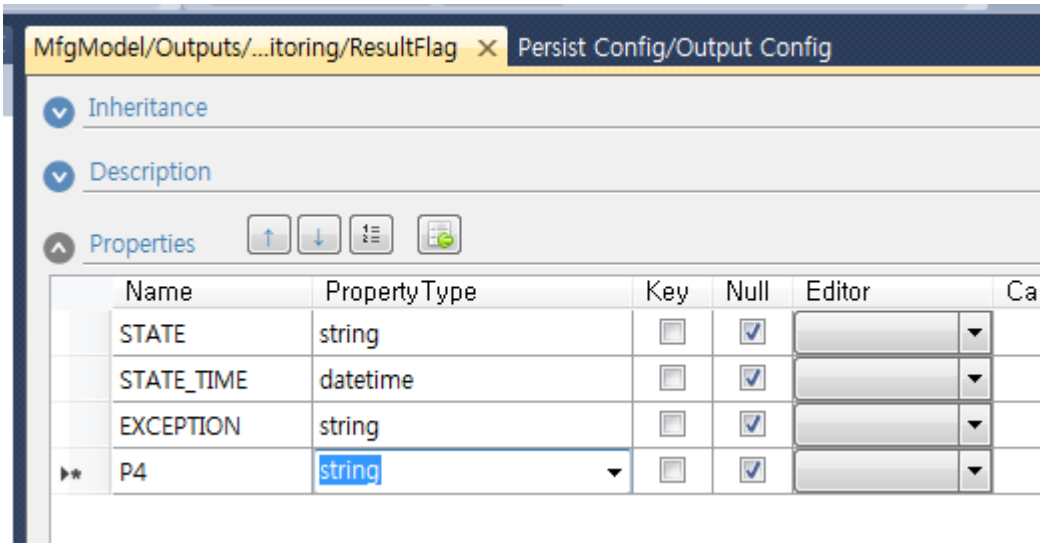
아래 그림과 같이 특정 시스템에서 작업을 실행 한 후, 결과를 확인하고 우리가 수행할 Job 을 실행해야 하는 경우 **1) 결과 확인을 위한 Job 을 생성** 하고 타 시스템의 결과를 **2) 모니터링하도록 Trigger 를 설정** 합니다. 그리고 마지막으로 과정 2에서 설정한 Trigger 를 대상으로 **3) Dependent Trigger 를 만듭니다.**



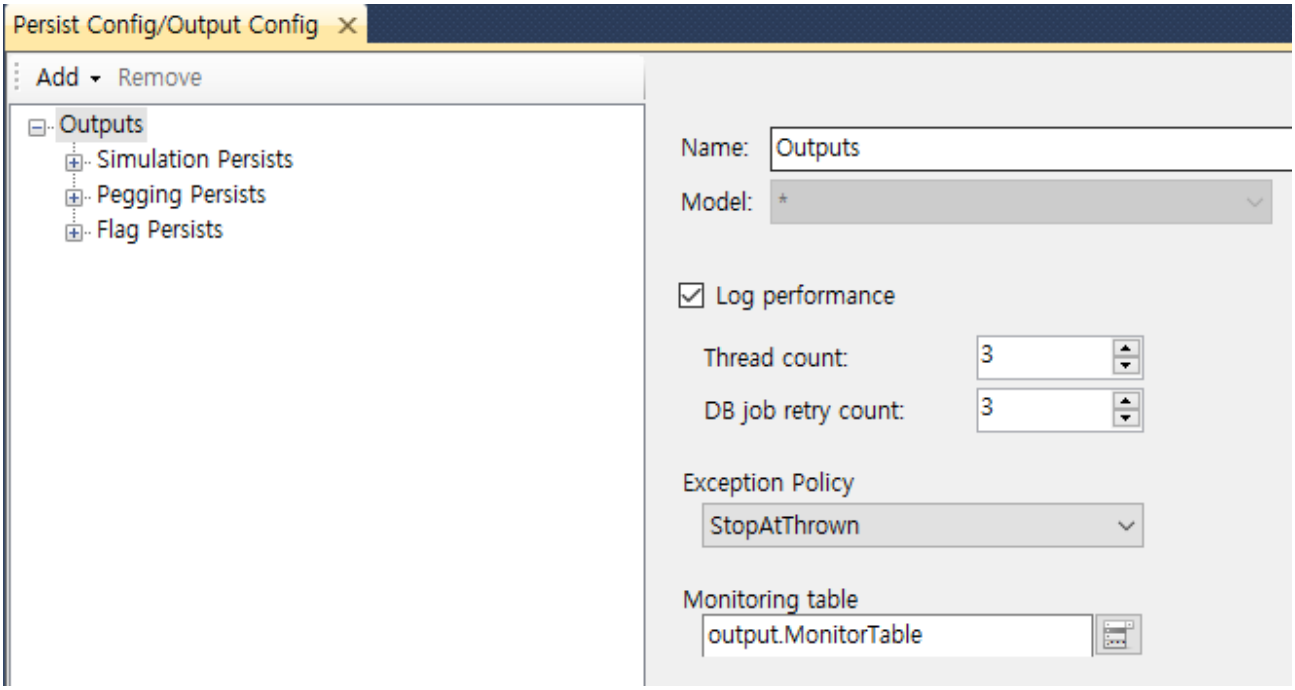
실행결과를 DB에 업데이트 하는 로직 추가(Monitoring 대상 Model)

Mozart Project 의 실행이 정상적으로 종료되었는지 여부를 외부의 DB 로 기록하여 Monitoring 하기 위해 아래의 절차를 사용합니다.

1 . Output 결과 기록을 위한 Output Data Item을 정의합니다.



2. Output Persist Config에 Monitoring table 속성에 결과 기록을 위한 Output data item(1번 항목에서 정의한 item)을 지정합니다.

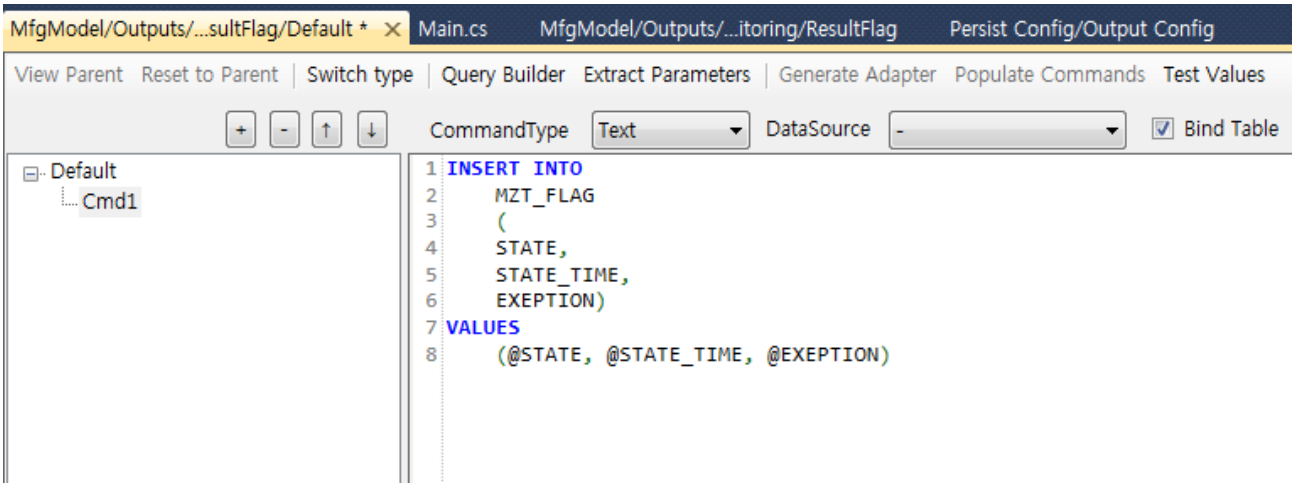


3. MainControl의 `ShutDown` FEAction을 구현합니다. 아래는 예제 소스입니다.

```
// Main > Shutdown 함수 구현
public void SHUTDOWN_0(ModelTask task, ref bool handled)
{
    //Job 수행 결과 기록
    MonitorTable mt = new MonitorTable;
    mt.VERSION_NO = task.Context.VersionNo;
    mt.STATE_TIME = DateTime.Now;
    mt.STATE = task.Context.HasErrors ? : "FAIL" : "SUCCESS";
    mt.EXCEPTION = task.EXCEPTION.Message;

    OutputMart.Instance.MonitorTable.Add(mt);
}
```

4. 결과 Ouput에 대한 DataAction 설정을 통해 Monitoring용 DB에 결과를 기록합니다.



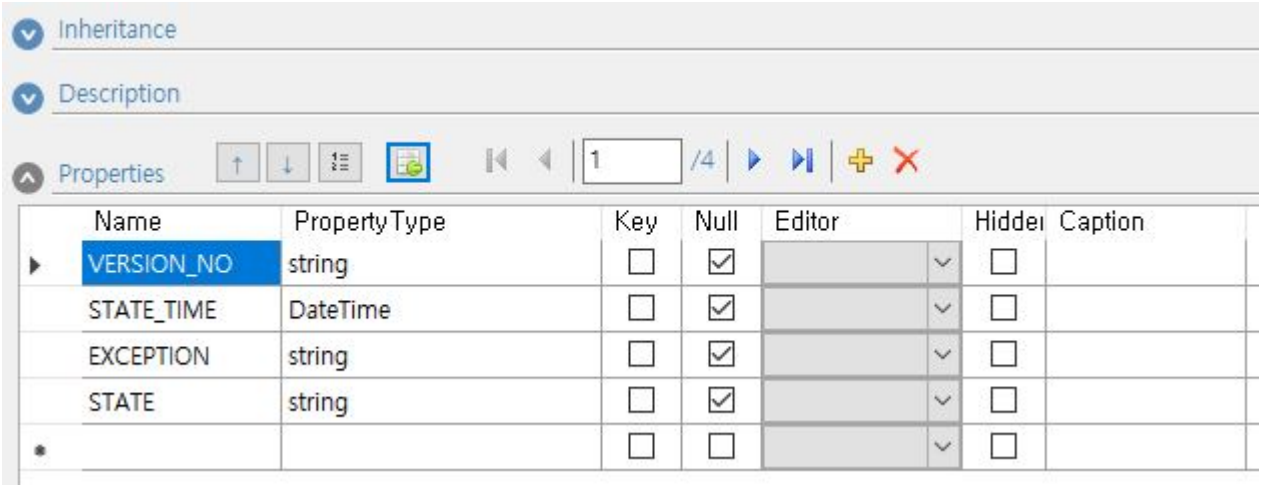
실행결과 확인을 위한 Job 생성(Monitoring Task)

실행결과 확인을 위한 Job 은 기본적으로 **1) 특정 DB로 부터 데이터를 조회하여 결과를 확인**하고 **2) Target Job 실행여부를 기록**한 후 **3) 확인된 결과를 반환**하도록 작성합니다.

> Dependent Trigger 의 **ReturnIfTrue** Type Job 의 return 값은 Main 함수의 **ModelContext**의 **Result** 값에 설정합니다. 설정값은 **boolean** 형식입니다.

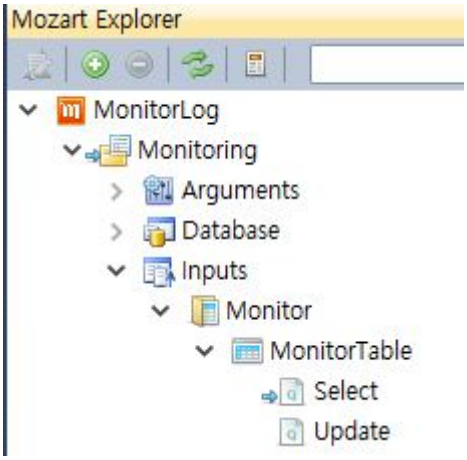
대상 Job의 실행 결과를 확인하기 위한 Monitoring Task 준비는 아래의 절차를 사용합니다.

1. Monitoring 대상 Job의 실행 결과 Output을 기록하는 DB table를 조회하기 위한 Input Data Item을 정의합니다.

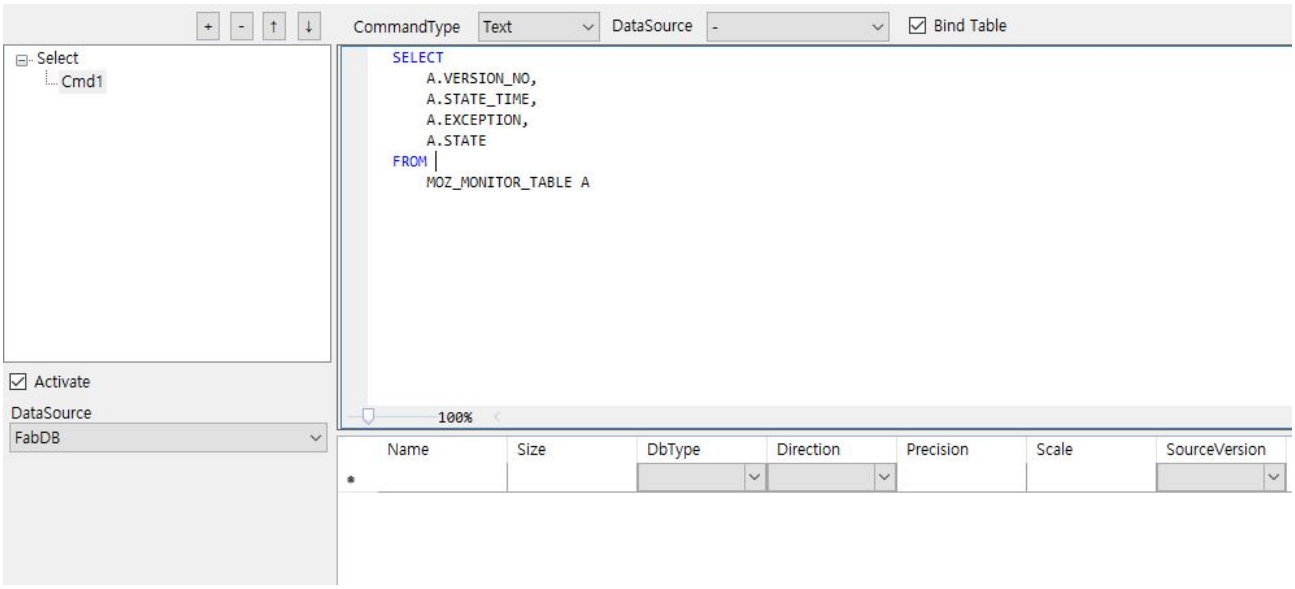


2. 새롭게 정의한 Data Item에 Default Data Action에는 DB로부터 Job 실행 결과를 조회 할 수 있는 쿼리, 그리고 Data Action을 하나 더 추가하여 Monitoring Task에서 업데이트 할 쿼리를 정의 합니다. 이때 Select문을 실행 할 DataAction `Active` 로 설정합니다.

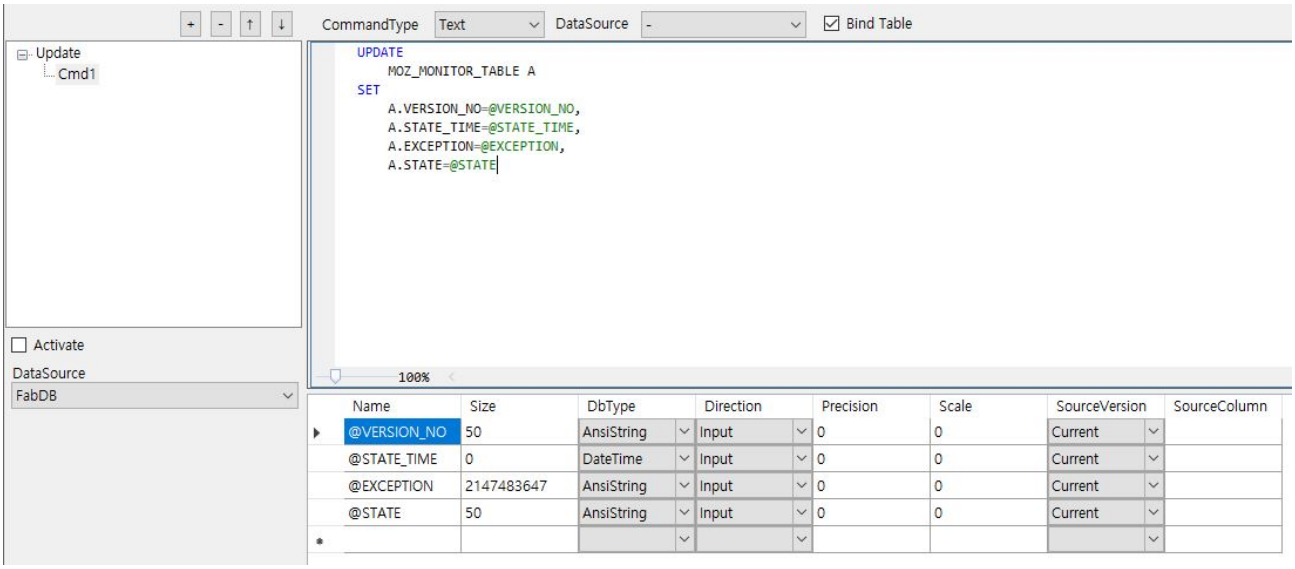
DataAction 추가 화



Select문 구현 예제



Update문 구현 예제



3. 대상 Job 수행 결과에 따라 결과값을 true를 반환하는 로직을 구현합니다. 아래는 **[Main > Run]** 함수의 구현 예입니다.

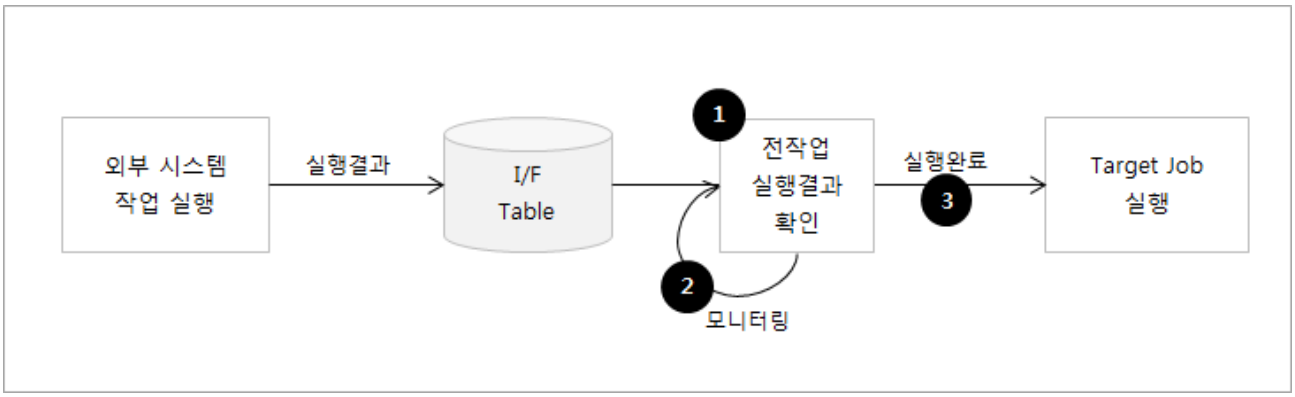
```
//Main > Run 함수 구현
public void MONITORING(ModelContext context, ref bool handled)
{
    var info = InputMart.Instance.MonitorTable.Rows.FirstOrDefault();

    //이전 Job 결과가 기록 그리고 정상 수행된 경우 Monitor Table 업데이트
    if(info!= null && info.STATE == "SUCCESS")
    {
        //다음 Job 성공까지 본 조건문에 들어오지 않게 STATE 값 변경.
```

```
10          info.STATE = string.Empty;
11          info.STATE_TIME = DateTime.Now;
12
13          //MonitorTable 이력을 DB에 업데이트 하기 위한 쿼리 실행 및 실행 여부를 
14          var dir = Path.Combine(context.ModelDirectory, context.VModelName
15          var model = Mozart.Task.Model.ModelEngine.Load(dir);
16
17          var source = new MonitorTable[] {info};
18          var result = InputAccessHelper.Save<MonitorTable>(model, "MonitorT
19
20          //쿼리가 정상 실행된 경우 result = 1이며, 그 경우 결과값을 true 반환
21          //ReturnIfTrue 타 Dependent Trigger를 실행함.
22          if(result > 0)
23              context.Result = true;
24      }
25  }
```

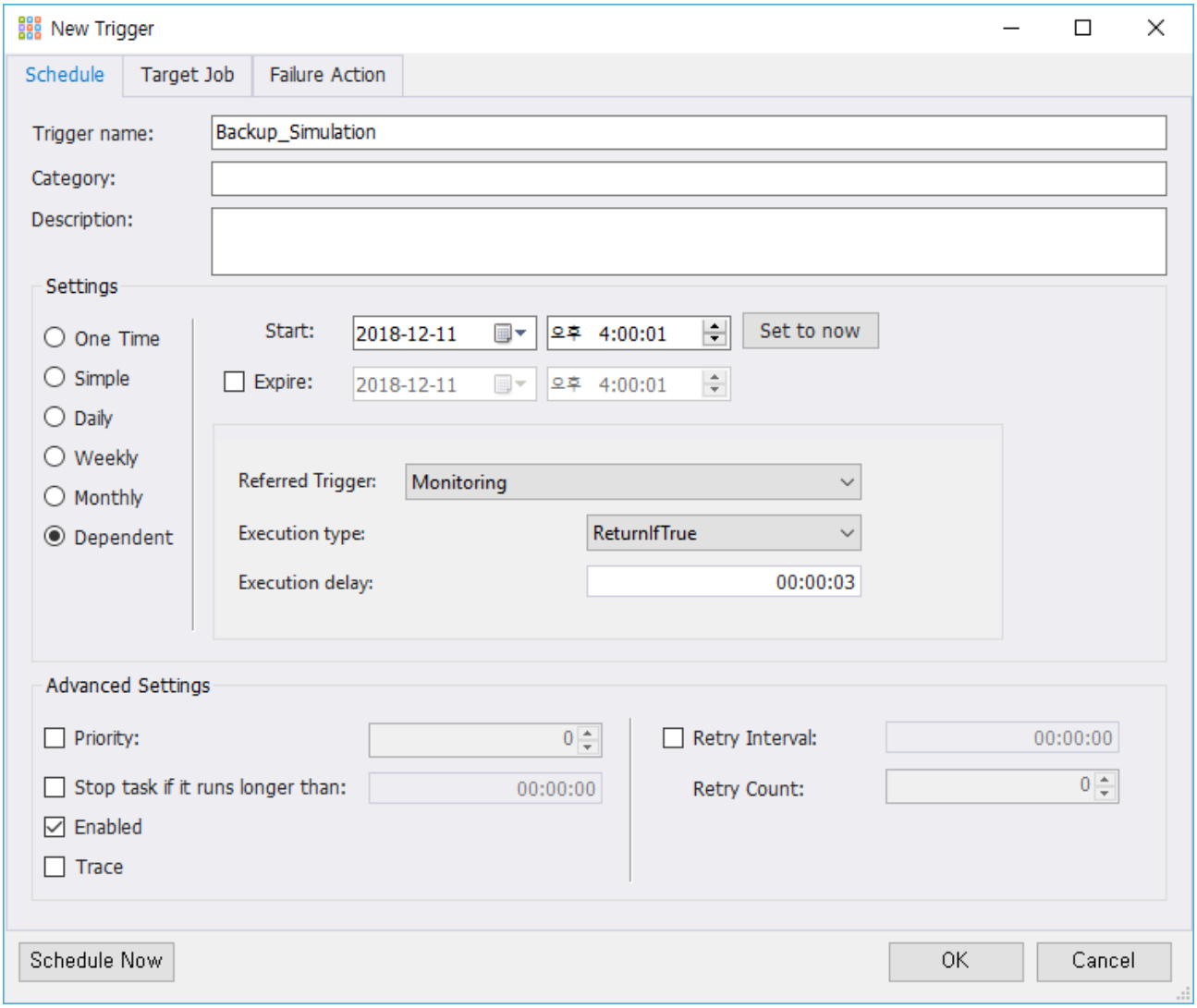
모니터링 Trigger 설정

2번에서 개발한 Monitoring을 위한 vModel 및 dll를 MMC를 통해 Upload하고, Job을 생성하여 해당 Job을 실행하기 위한 Trigger를 설정합니다. 통상적으로는 Monitoring Job을 수행하기 위한 Trigger는 **Simple Trigger**로 설정을 하며 주기는 경우에 따라 최소 1분, 최대 1시간으로 설정을 합니다. 아래의 예제는 매일 오후 4시부터 10분 단위로 15분간 Trigger가 수행됩니다. `for a duration of` 옵션은 Repeat count가 무제한(-1)으로 설정된 경우에만 활성화 됩니다. 만약 상시로 10분마다 상태를 체크해야 하는 경우에는 `for a duration of` 옵션을 설정하지 않습니다.



Dependent Trigger 설정

Monitoring Trigger가까지 등록을 하였다면, 마지막으로 Target Job 실행을 위한 Depdendent Trigger를 생성하여 설정합니다. 이때 `Referred Trigger` 는 Monitoring을 수행하는 Trigger를 선택하고, Monitoring 결과값이 true인 경우에만 실행이 필요한 관계로 `Execution type` 을 **ReturnIfTrue**으로 선택합니다. 아래는 설정 예제입니다.



Extended Arguments

The actual execution of Task and Model that are created through MOZART Project is done through ModelTask of MOZART execution engine. The preset arguments to adjust the execution options of Model Task are System Arguments. Developers can assign these System Arguments to Input Arguments of the Model to use the preset execution options. The followings are the descriptions of System Arguments.

Basic Arguments

Argument Name	Argument Description	Data Type
#experiment	Name of experiment that Model execution's output is created. Default is "Experiment 1"	string
version-no	Model's version name (Default format : {model-name}-{yyyyMMdd-HHmmss})	string
model-name	Default name for versionNo when there is no versionNo entered	string
start-time	Task starting time (Simulation clock)	DateTime
end-time	Task completion time (Simulation clock)	DateTime
period	Plan&Schedule period	float
period-unit	period configuration unit (default : day)	string
#start-time.AdjustMinutes	Input variable to adjust starty time tp job execution time	int
#model-file	Full path of the vModel file	string
#model-dll	Full path of the model dll file	string

#model-config	Full path of the model configuration file	string

Data Download/Upload Arguments

Argument Name	Argument Description	Data Type
#overwrite_result	Option whether to overwrite result or not.	boolean
#use-database	Option whether to use database or not. (Input data download)	boolean
#save-database	Option whether to save output data to DB or not.	boolean
#db-to-file	Option whether to synchronize database without running simulation. (default value = false) : Input data download	boolean
#file-to-db	Option whether to synchronize database without running simulation. (default value = false) : Output data save to DB	boolean
#db-includes	File name containing the list of tables to synchronize input data to the database. The tables not listed will not be synchronized.	string
#db-excludes	File name containing the list of tables not to synchronize input data. All tables except for the target tables will synchronize and if there is same table entered in #daction_includes, the following table will not be excluded.	string

#daction_excludes	List of tables that doesn't execute DataAction after simulation is completed. Comma is used as separator.	string
#daction_includes	List of tables that execute DataAction after simulation is completed. Comma is used as separator.	string
#daction_excludes/in	List of actions not to be executed from Input Schema's DataAction during simulation. Comma is used as delimiter.	string
#daction_includes/in	List of actions to be executed from Input Schema's DataAction during simulation. Comma is used as delimiter.	string
#dataSource-set-default	Sets the connection string to use as default from the model. The key is the name of the data source and the value is the name of the connection string. In case multiple connection strings need to be set the delimiter is semicolon (;).	Dictionary<string,string>
#datasource-set-default-exception	The option whether to occur an exception in case the connection string specified in #dataSource-set-default could not be found.	boolean

Logging/Performance Arguments

Argument Name	Argument Description	Data Type
#log-dir	The relative path (Working Directory\Logs) to save the trigger execution log files.	string
#log-level	Sets the log level. (Verbose~Fatal)	string
#performance-profiling	Option whether to collect performance data of model execution (default = false). Trigger Execution Log information will not appear from Triggers and Monitoring if the option is set to false.	boolean

Run Arguments

Argument Name	Argument Description	Data Type
#more-runs	Repeat count of Model execution.	int
#more-config-[runindex]	This variable is used to configure argument's value for each repeated execution. If not designated, argument value of the previous occasion is used. This is automatically created by MMC	string
#run-index	The current repetition's index. This is automatically created by MMC.	int

Temp Folder Run Arguments

Argument Name	Argument Description	Data Type

#use-run-dir	The argument to indicate whether to create a temporary folder to execute the trigger. For more details see here.	boolean
#max-run-dir	Sets the maximum number of temporary folders to maintain. The oldest folder will be deleted when the number of folders created exceeds the number set in this argument.	int
#use-parent-path	Indicates whether to use the most recently created temporary folder of the reference trigger when dependent trigger executes. This argument is valid when #file-to-db is set as true.	boolean

Zip Model Arguments

This Arguments are used to configure rules for making a compressed file(like ZIP file) from an executed Model.

Argument Name	Argument Description	Data Type
#zip	Option whether Model is compressed after simulation is completed.	bool
#zip.FileNameTempate	Template to save the name for compressed file. Default template is "${Model_name}_${zip_now}${zip_postfix}" The followings are the allowed keywords to be used. • ${Model_name} : Name of Model • ${now} : Time when compression begins (DateTime) • ${zip_now} : Time string (format : yyyyMMddHHmmss) • ${zip_postfix} : postfix used for compressed file name • ${version_no} : Model's execution version name	string

#zip.FileNamePostfix	Postfix for compressed file name	string
#zip.Path	The path to create compressed file. If not set, the file is saved where Model files are located. The folder is created as a relative path to Working Directory or else Working Directory itself will be used.	string
#zip.UpdateToRecent	Option whether the recently compressed file is updated or not. If true, the most recently-compressed file is overwritten with the same name. If #zip.FileNameTemplate begins with yyyyMM format, new compressed file is created with the name of the most recently-compressed file that has the same year and month.	bool

Hosting Arguments

The arguments listed below relates to the setting for hosting job/trigger from different Mozart server versions.

Argument Name	Argument Description	Data Type
#host-dir	This argument is to set the relative path of the mozart server located in the WorkingDirectory to execute the trigger from a different version from the mozart server installed currently. For more details, see here.	string

#host-version	This argument is to set the version of the moart server to execute the trigger from a different version from the mozart server installed currently. This argument works as same as #host-dir but instead of locating the mozart server DLL files to the working directory, this argument finds the DLL files of the specified version from the Execution folder. When #host-dir and #host-version is set at the same time, the trigger will be hosted from #host-version. For more details, see here.	string

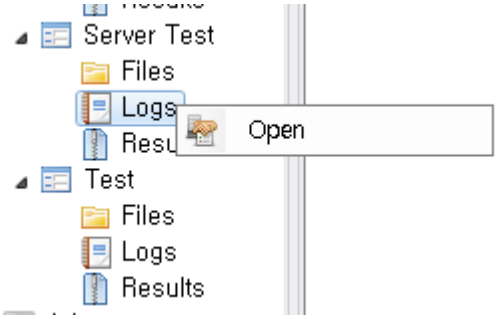
Checking Logs

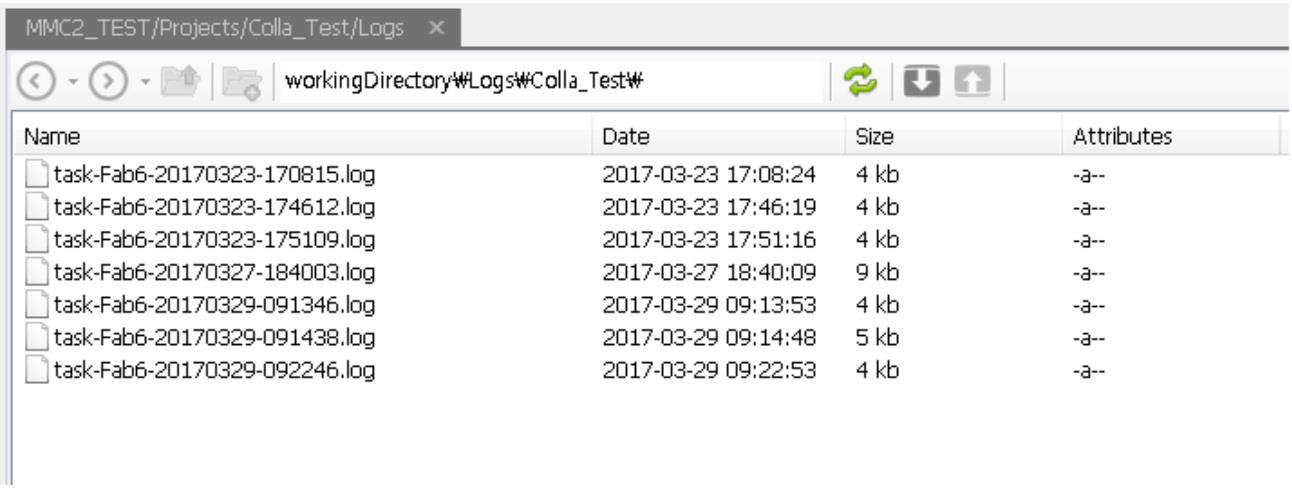
The logs for executed Triggers could be found through Logs folder located in Projects node of the mapped Job/Trigger. In addition, execution logs could be acquired from Monitoring node as well.

Viewing Logs in Projects

1. In Server Explorer -> Projects node, open the Project node linked with the Trigger.

2. Double-click the Logs folder or click **[Open]** through right-clicking.





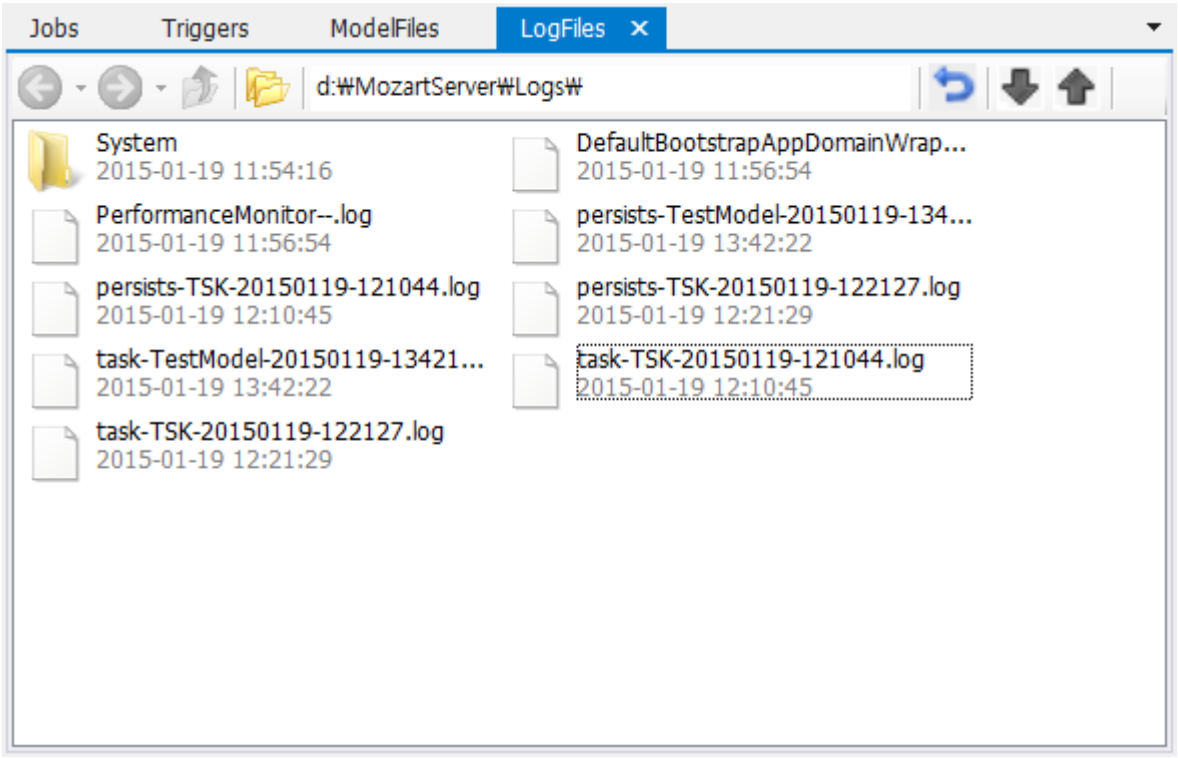
> Task logs are normally stored in WorkingDirectory\Logs. System folder in Logs folder stores the logs for the entire events occurred in the server. If the Model

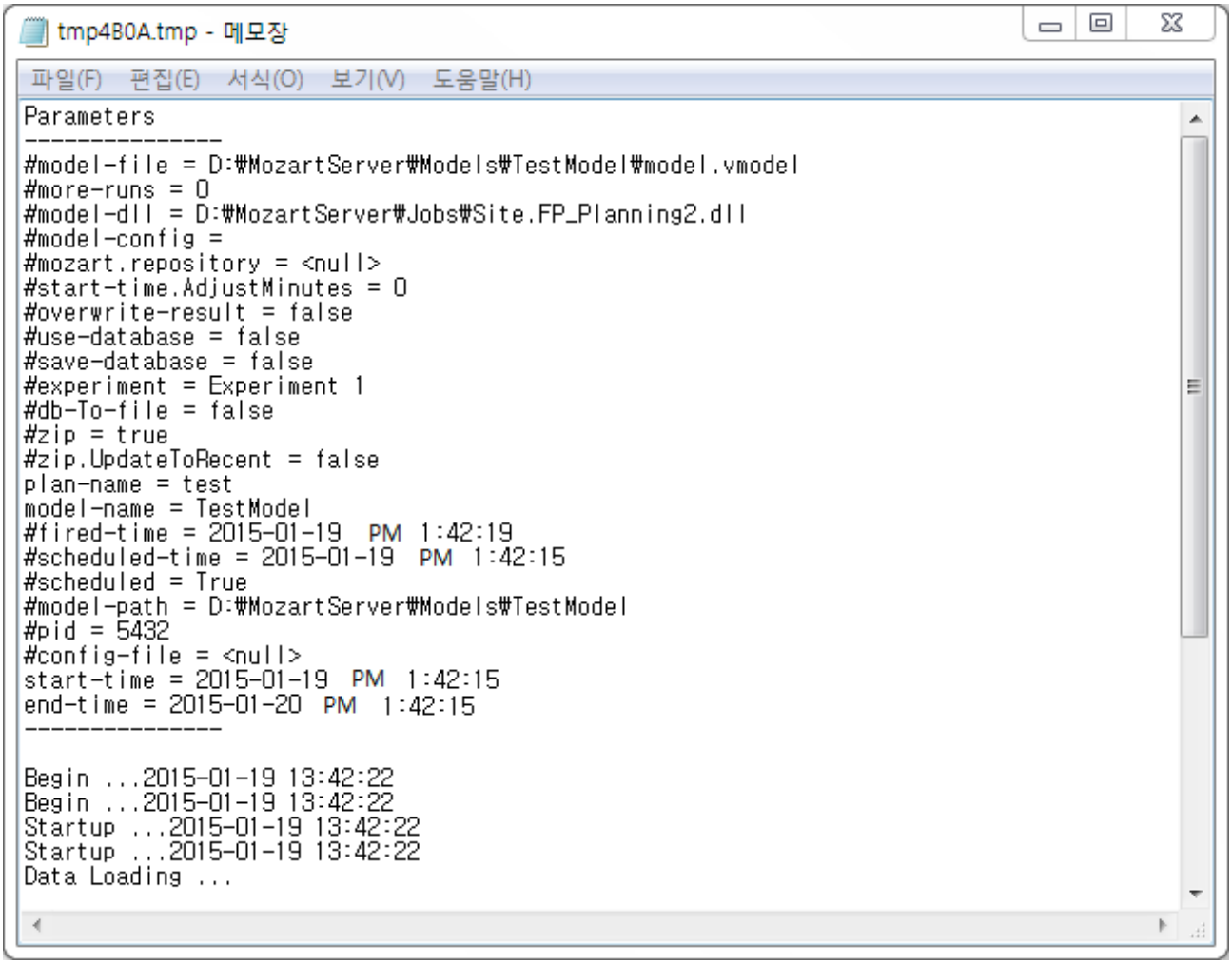
> does not have a config file(check **How to designate a Log**) to designate a folder to store the logs, the logs will be stored in Logs folder.

3. Double-click the task log file in the target time to view the execution log results.

> Two trigger logs are generated when job is triggered. The file name of the log is the parameter value model-name which is one of the arguments in Model.

- **task log** : this log file records a log result when task module is executed. If model-name is registered and configured in Input Argument, a log file including the corresponding model-name is created. (Ex. When model-name is configured as 'SAMPLE', log file name is created like task-SAMPLE-20150101-125648.log.) If file is double-clicked, the corresponding log file is opened by the default editor(i.e. Notepad) and can be checked.





- **persists log** : Log file that records persist results before/after task module is executed

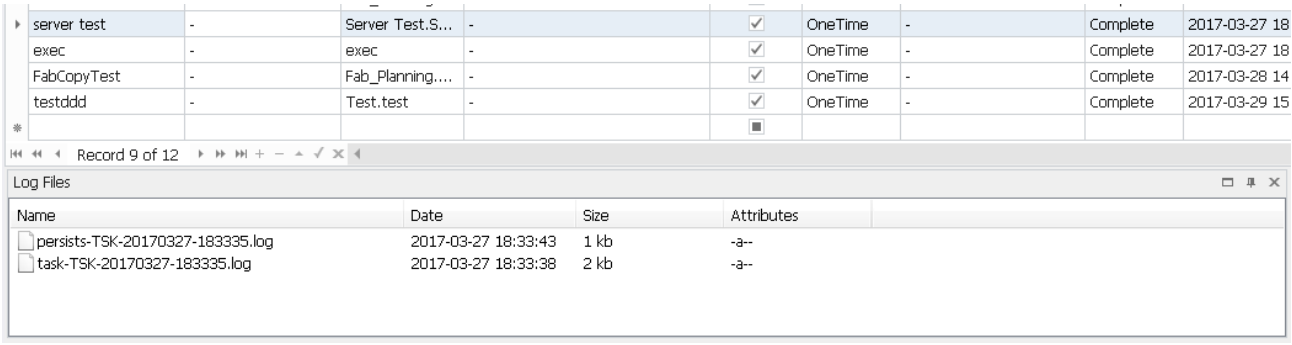
---

Viewing Logs in Triggers

1. Double-click the Server Explorer -> Triggers node or click the right mouse button menu **[Open]**.

2. When the Triggers tab is open, select the Trigger you want to view the Log.

3. The Log Files area at the bottom of the Triggers Tab window displays a list of the Task and Persist log for the Trigger. It will open in Notepad when you double-click on the log file you want to view.



Viewing logs in Monitoring

1. In Server Explorer -> Monitors node, double-click the View that you want to check the Log or click the right mouse button **[Open]**.

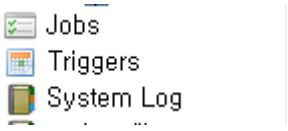
2. Like in the "Triggers Tab", if you select the Trigger that you want to check the Log, you can see a list of the Task and Persist log of the Trigger selected in the Log Files area at the bottom of the Monitor / [View Name] tab window.

3. Double-click the log file to open it through Notepad.

Check logs via System Log

1. Double click on **[System Log]** node or check WorkingFolder/Logs/system folder through Trigger log folder.

2. 'app.log' is the log of the current day. The logs from the past will have dates behind the names. (i.e appYYYYMMDD-xxxxx.xx.log). app.log file is backed up as appYYYYMMDD.xxxxx.xx.log format after one day is pasted. Double-click on the log file to check the logs.



Monitoring

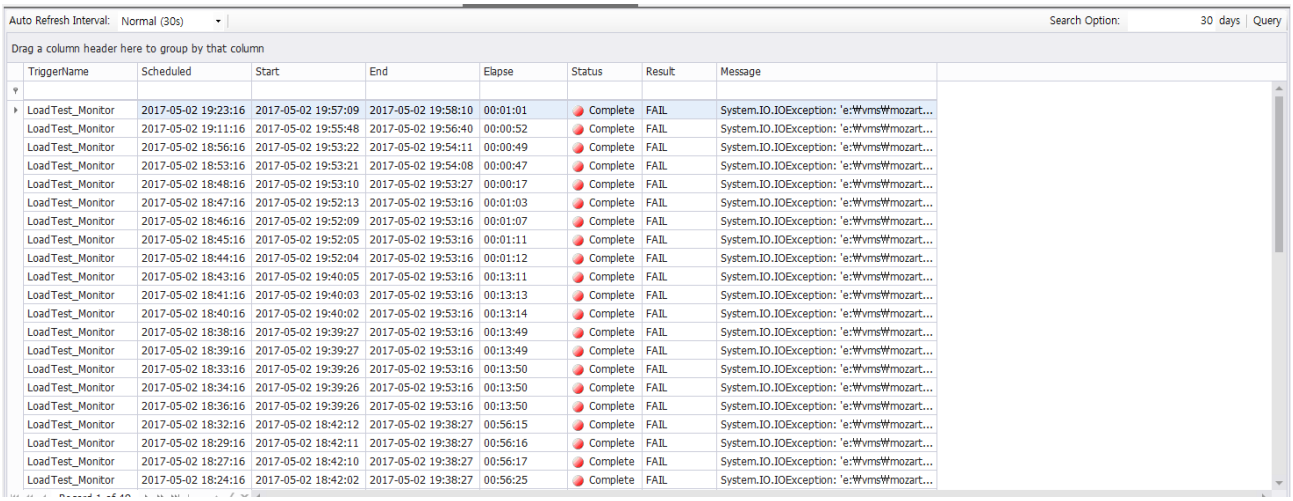
In Monitoring, you can check the execution status of Trigger in MMC, CPU / Memory usage status of target server that performs Trigger, log for each Trigger, and execution time per Task.

When MOZART Server is installed, Monitor node is created on the target server in Server Explorer. By default, all view that can check the latest status of triggers and error view to check error history is created as Child node. In order to monitor specific triggers, users can add views directly and check the monitoring history of each triggers. The following descriptions are about the basic interfaces and functions of Monitoring.

Nodes

A description of Monitor node in Server Explorer.

- **All** : If the registered Trigger is executed once, a history will be saved in All View. In All, the most recent information of the target trigger is only displayed. When Trigger A is executed once at 9 o'clock and executed at 10 o'clock again, All displays information about Trigger A performed at 10 o'clock.
- **Errors** : It displays the execution error history information of Trigger. While the most recent information of the target trigger is only shown in All, the target trigger is recorded every time an error occurs in Errors. Errors allows you to set the period that users want to view records.



- **User-Defined View** : A View users added. A user-defined view can check the monitoring status of all the execution history of the trigger selected by the user. The list is output in 10 units according to the activated window size by paging method. Please refer to **Monitoring View Registration / Modification** for adding user View.

Monitoring Table

The following describes the status information table of Trigger which is the main section in Monitoring UI.

- **TriggerName** : The name of the target trigger registered in Trigger.
- **Scheduled** : Displays the time when the target trigger is scheduled to run.
- **Start** : Displays the time at which the target Trigger actually started to run.
- **End** : Displays the target trigger is ended.
- **Elapse** : Displays the total elapsed time of trigger execution.
- **Status** : Displays the current status of the target trigger.
  - Run : Target Trigger is currently running.
  - Complete : Target Trigger has successfully completed with no errors.
  - Aborted : Target Trigger has been stopped by force.
  - Complete : Target Trigger has ended abnormally due to an error.
- **Result** : Displays the result of Trigger execution. Only when the status of target Trigger is Complete, it is recorded.
  - SUCCESS : SUCCESS is recorded when the target trigger finishes normally.

    - FAIL : If the target trigger is abnormally terminated due to an error during execution, FAIL is recorded. (The aborted status is not recorded in Result when Trigger is forcibly terminated by the user.)
- **Message** : This column records an error message when an error occurs during execution of the Trigger.

Top Menu Bar

- **Stop Trigger** : This button is used to forcibly terminate the running Trigger. If the trigger is terminated by “Stop Trigger” in Monitoring or “Stop task if it runs longer” in Trigger, Trigger is terminated and the status turns into Aborted .

  ⚠ If the trigger is terminated by “Stop Trigger” in Monitoring or “Stop task if it runs longer” in Trigger(**Trigger Setting**), it can also be changed as Complete (FAIL) according to the thread.
- **Auto Refresh Interval** : Set the time interval for updating the status of Monitoring Table.
    - Normal : This is the default setting. Updates Monitoring table information every 30 seconds.
    - High : Updates Monitoring table information every 10 seconds.
    - Low : Updates Monitoring table information every 60 seconds.
    - Pause : Does not update Monitoring table information as long as there is no user intervention.
- **Query** : It is used in Errors or a custom view. Enter the period to be searched in Search Option, then it displays the monitoring information for the period when executing the query.

Performance Trend

In Performance Trend, only the $model and $cola Job Types are analyzed. In the Performance Trend graph of the Monitor, the base line is drawn based on the start of the selected target triggers and the CPU / Memory usage of the Server Machine within 10 minutes before and after the base line and the number of running triggers at that time are

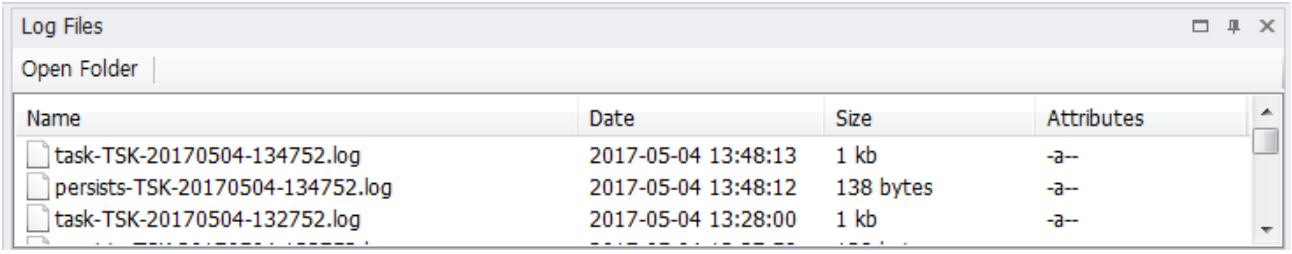
displayed. The Trigger's Count is calculated by not only the Trigger you selected but it also includes other triggers that were running at that time. If a user views the Trigger A and Trigger B is running at the time, the Trigger's count will be 2 in the Performance Trend.

CPU / Memory usage shown in Performance Trend includes CPU / Memory used by other processes besides Trigger (Mozart Agent). The dashed lines in the Performance Trend graph means the average CPU / Memory usage of the Server Machine and the solid lines indicate the actual usage of the time. The gray area means the number of triggers at that time. You can check the detailed information in Performance Trend graph when mouse over the graph.

> Performance Trend aggregate performances every 1 minute. If Trigger runs shorter than 1 minute, it could bypass the aggregation interval and may not be shown in the trend. (Performance, Count)

Log Files

In Log Files, you can look up the history log file of selected Trigger in Monitoring. You can open the log file directly by double-clicking the mouse or download the log file locally through the right-click menu. You can open the folder tab where the log file is located through the Open Folder button. Like Triggers, Log Files in Monitoring displays the log files for the last 10 Triggers.



Trigger Execution Log

Trigger Execution Log allows you to check the time consumption of the selected trigger. To record the Trigger Execution Log in Monitoring, the #performance-profiling option of Trigger Argument should be True.

- **DOWNLOAD** : Displays the total time downloading data from DB during task execution.
- **PERSIST_IN** : Displays the total time loading input data during task execution.
- **ENGINE_RUN** : Displays the consumption time of module (Pegging, Simulation, CBS, etc) during task execution.
- **PERSIST_OUT** : Displays the total time storing engine results during task execution.
- **SAVE_DB** : Displays the total time uploading the result data to the target DB during task execution.

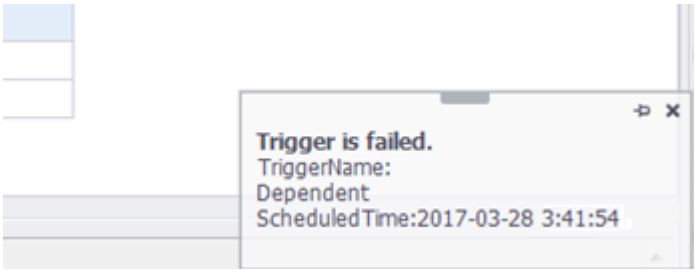
Error Message

The Error Message records a detailed message about an error when the executed Trigger is abnormally terminated due to an error.

You can copy Error Message to the clipboard or check detailed error message.

Error Notification

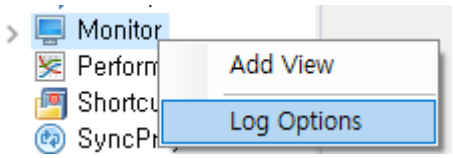
Mozart Management Console (2.0) not only records logs in the Errors view when an error occurs during the execution of the engine but also notifies the user that an error has occurred.



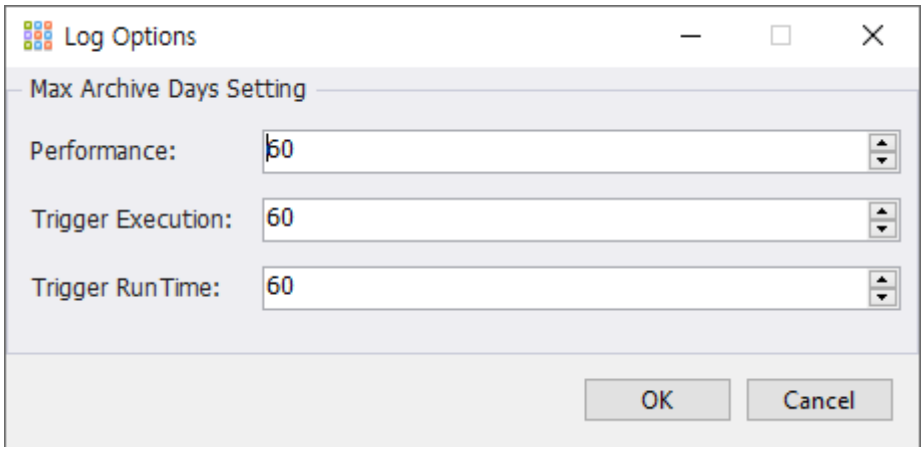
In order to see the error notification, Errors View tab should be opened. Error notification pops-up only when the trigger error occurs during RUN state. No notification will be shown when the trigger has encountered an error as soon as its been executed.

Setting Log Preservation Period

The period to preserve the trigger execution, trigger run time and performance logs from MMC. Log Options menu appears by clicking the right-button of the mouse from the Monitor node.



60 days are set as default. The period can be modified in days.



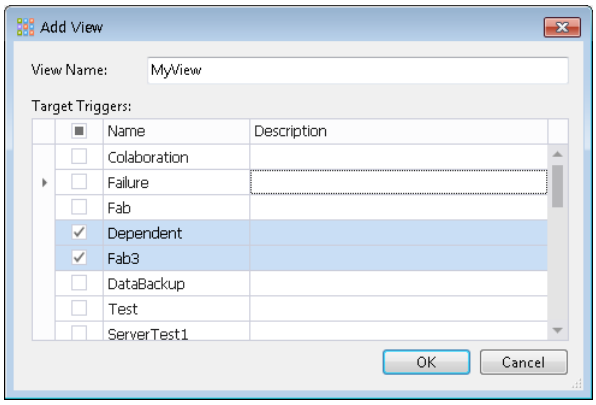
How to Use Monitoring View

Monitoring View Registration

1. In Server Explorer, select a sever to register Monitoring View.

2. Right-click on the Monitoring node of the target server and select **[Add View]**.

3. In the Add View Dialog, enter a name for the View Name and check the checkbox for the trigger you want to monitor. (Multiple choices available)



4. Click **[OK]** button to complete.

Monitoring View Modification

1. In Server Explorer, select the target server for the View modification.

2. Select the view you want to modify from the Monitoring node of the target server, right-click and select **[Edit View]**. (Predefined views such as All and Errors cannot be modified.)

3. Click **[OK]** button to complete.

Delete Monitoring View

1. In Server Explorer, select the target server to delete a view.

2. Select the view you want to delete from the Monitoring node of the target server, right-click and select **[Delete View]**.

3. Click the **[Yes (Y)]** button in the pop-up window to remove the target view.

Trigger Performance

The Performance node of MMC2 monitors the overall performance of triggers registered in the target server. By analyzing the performance through the Trigger Performance, you can check which stage of the task is delayed when the engine is started. Especially, developers and administrators can compare the performance of vmodel or dll before and after through the data. The following sections describe how to view the performance of the Trigger registered in the target server and the functions of the Performance window.

How to Collect Trigger Performance Information

To check the performance of the trigger registered in the target server's in Performance, one of the options in Trigger Argument needs to be enabled. The argument can be configured as follows.

1. In Server Explorer, select the target server for which you want to set the Trigger Argument.

2. Double-click the Trigger node of the target server to activate the Trigger window.

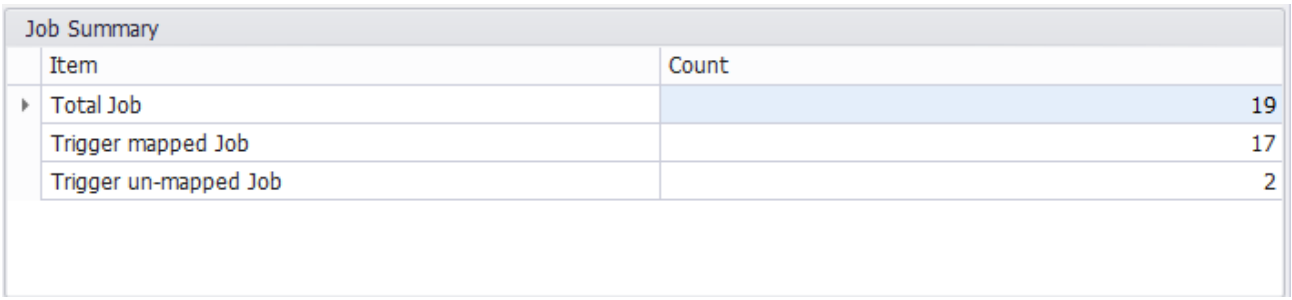
3. Select the target Trigger to record in the Performance and double-click or click the **[Edit]** button in the upper menu bar.

4. In the Edit Trigger Dialog, go to the Target Job tab.

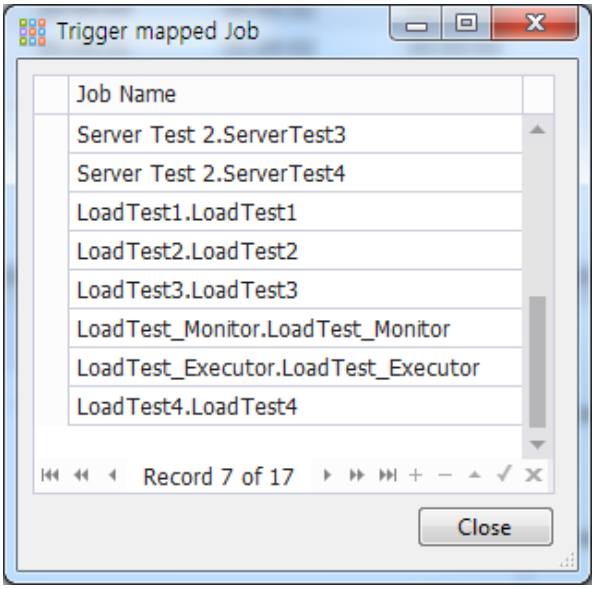
5. Go down to the bottom of the list shown in the Target Job, check the #performance-profiling check box and click the **[OK]** button to save the settings.

Performance UI

- **Job Summary**
  In Job Summary, you can check the total number of jobs registered in target server and the number of triggers mapped to job.



- - **Total Job** : The total number of jobs registered in the target server.
  - **Trigger mapped Job** : The number of jobs registered in the target server, which is registered by Trigger. When you double-click Trigger mapped Job Row, you can check the list of jobs for which the trigger has been registered. Even if multiple triggers are mapped to one job, the count of job is one in the trigger mapped job. (EX: Job A {Trigger A, Trigger B, Trigger C} -> Trigger mapped Job = 1)



  - **Trigger un-mapped Job** : The number of jobs that triggers are registered in the target server. When you double-click Trigger un-mapped Job Row, you can see a list of jobs for which triggers are not registered.
- **Trigger Summary**
  Trigger Summary is a summary of the triggers registered in the target server. You can check the detailed information of triggers in the target server such as the number of triggers that are activated / deactivated among the registered triggers and the number of triggers to be recorded in the performance.

Trigger Summary

Item	Count
Total Trigger	20
Active Trigger	14
Profiling Trigger	6

- **Total Trigger** : The total number of triggers registered in the target server.
- **Active Trigger** : The number of Triggers registered in the target server with activated **[Enabled]** option.
- **Profiling Trigger** : The number of Triggers that #performance-profiling is true and registered in the target server.

- **Trigger Performance**
  Trigger Performance is the area where the performance aggregation information of Profiling Triggers is expressed numerically and graphically. When the task is executed, you can check the detailed information such as the consumption time of each step, the total execution count of the trigger, and the success / failure ratio. The following definitions are about the terms used in the Trigger Performance area.
  - **Period** : Sets the period during which users want to check Trigger Performance. The period can be selected from Days / Hours and calls the aggregated history before Days / Hours set based on the current time.
  - **Trigger**
    - Name : The name of the registered Trigger Node.
    - Description : A description of the registered Trigger of the target server. Description can be entered at the Triggers node.

Trigger

Name	Description
Colaboration	
Dependent	Dependent Trigger Test
Fab	
Fab3	
FabCopyTest	

  - **RunTime(sec)**
    - DOWNLOAD : Displays the average elapsed time downloading data from DB of the total number of times the task has been executed.
    - PERSIST_IN : Displays the average elapsed loading Input Data of the total number of tasks performed.

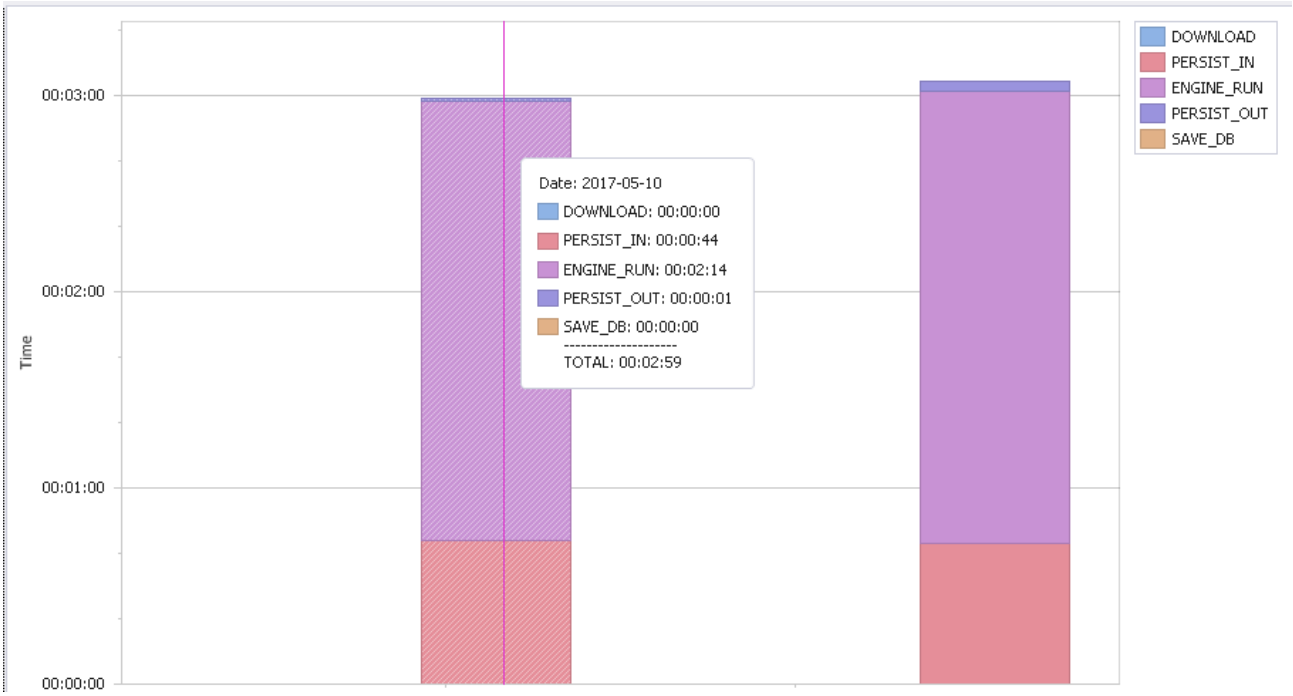
- ENGINE_RUN : Displays the average run-time of the execution of module (Pegging, Simulation, CBS, etc) of the total number of tasks performed.
- PERSIST_OUT : Displays the average elapsed time storing the engine results.
- SAVE_DB : Displays the average elapsed time writing the result data to DB.
- TOTAL_RUN : The average time of cumulative time from DOWNLOAD to SAVE_DB Action.

RunTime(sec)					
DOWNLOAD	PERSIST_IN	ENGINE_RUN	PERSIST_OUT	SAVE_DB	TOTAL_RUN
00:00:00	00:01:18	00:04:04	00:00:02	00:00:00	00:06:07
00:00:00	00:03:26	00:00:21	00:00:02	00:00:00	00:04:09
00:00:00	00:00:49	00:02:25	00:00:02	00:00:00	00:03:37
00:00:00	00:00:41	00:00:20	00:00:00	00:00:00	00:01:04
00:00:00	00:00:00	00:00:00	00:00:00	00:00:00	00:00:02

In RunTime (sec) of Trigger Perfomance, the average consumption time of each target Trigger by action is displayed. In the bottom left grid of the Trigger Performance area, you can check the maximum / minimum consumption time of each trigger.

Action	Min	Average	Max
PERSIST_IN	00:00:49	00:01:18	00:01:32
ENGINE_RUN	00:02:29	00:04:04	00:04:32
PERSIST_OUT	00:00:01	00:00:02	00:00:03

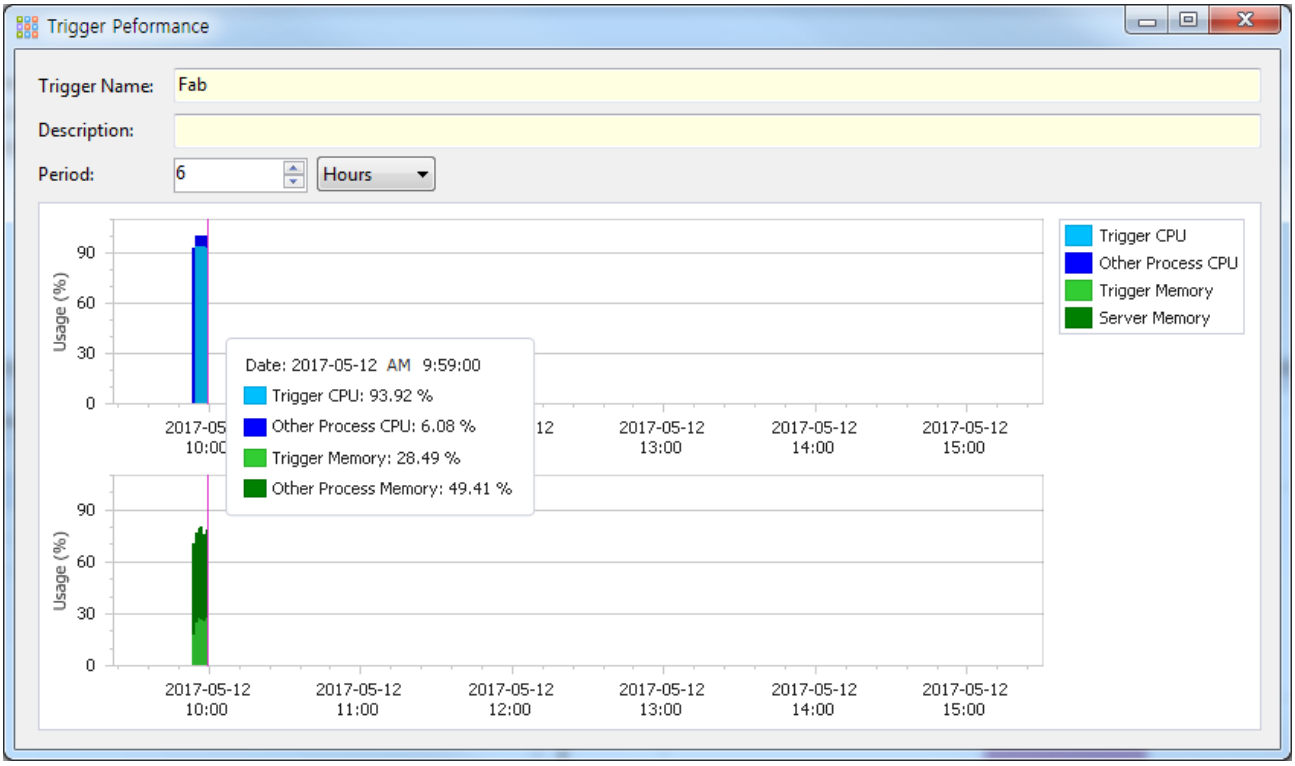
Furthermore, in the lower right part of the Trigger Performance area, you can see the ratio of the number of execution of the target Trigger for the user-defined period in the TOTAL_RUN time for each action in graph form. For checking the detailed information, you can check the execution time of Trigger action executed at the relevant time by mouse over the bar graph.



- **Reliability**
  - MIN : The shortest execution time among the entire execution of target trigger.
  - MAX : The shortest execution time among the entire execution of target trigger.
  - LIMIT : The execution time limit of the target trigger. You can set it in Triggers -> Target Trigger -> Schedule Tab -> Stop task if its longer than.
  - RUN_COUNT : The total number of times that the target trigger has been executed. The count does not increase in case trigger has been stopped by users stop the task with Stop task in Monitoring or the Stop Trigger if it runs longer than option.
  - FAIL_COUNT : The number of times the target Trigger has been abnormally stopped due to an error.
  - FAIL_RATE : The percentage of FAIL_COUNT out of RUN_COUNT of the target Trigger.
  - SUCCES_RATE : The ratio of RUN_COUNT of the target Trigger that was performed normally.

Reliability						
MIN	MAX	LIMIT	RUN_COUNT	FAIL_COUNT	FAIL_RATE	SUCCESS_RATE
00:03:35	00:07:06	00:00:00	5	0	0 %	100 %
00:04:09	00:04:09	00:00:00	1	0	0 %	100 %
00:00:01	00:07:37	00:02:00	22	6	27 %	73 %
00:01:04	00:01:04	00:00:00	1	1	100 %	0 %
00:00:02	00:00:02	00:00:00	1	1	100 %	0 %

- **Trigger Performance Dialog (Server Machine Resource Utilization Check)**
  In the Performance window, the CPU / Memory usage and percentage of the target Server Machine is displayed in a graph form in the Trigger Performance Dialog during execution of trigger. In the Trigger Performance Grid of the Performance window, the Trigger Performance Dialog window pops up if you double-click the Row of the target Trigger.



The Period is based on the user-specified period in Trigger Performance and users can set the period again through In the Trigger Performance Dialog window. In the Trigger Performance Dialog window, the execution time is counted at the time when the target trigger is executed. The total CPU / Memory usage of Server Machine during Trigger execution time is displayed as a graph. The upper blue graph shows the CPU usage and the lower green graph shows the memory usage. The light blue bar graph is the percentage of total CPU usage that the target Trigger (MozartAgent process) occupies and the light green bar is the percentage of Memory total usage that the target Trigger occupies. Darker colors represent the percentage of processes other than triggers. like most graphs in MMC2, you can check the detailed information in Trigger Performance Dialog if you mouseover over the graph bar.

How to Use Backup

When registering multiple Jobs/Triggers in the Server and executing Tasks, the HDD capacity can be fill in WorkingDirectory because of the accumulated engine result and log file. The results of engine run cannot be saved because due to insufficient HDD capacity when the administrator doesn't care about the HDD capacity. MMC2 provides the function to allow users to schedule backups for managing HDD capacity efficiently. The following settings are possible through the MMC2 Backup function.

- **Backup Cycle** : Users can set a backup cycle just the way you set Task Trigger.
- **Back Path** : You can designate the path for backup to different drive other than where WorkingDirectory is located.
- **Delete Source file(s) after backup** : You can decide whether to keep or delete the source file after performing the backup operation.
- **File Filtering** : You can set whether only certain extensions in the source path are backup progress or exception.
- **Set Min/Max File Size** : You can set the maximum and minimum size of files to be backed up.
- **Back for File(s)/Folder(s) with Specific Date** : You can set files to back up that are created within specific time or date.

Please check **Backup Registration/Modification** to see more details on how to configure backup schedule through MMC2.

Creating Backup Schedule

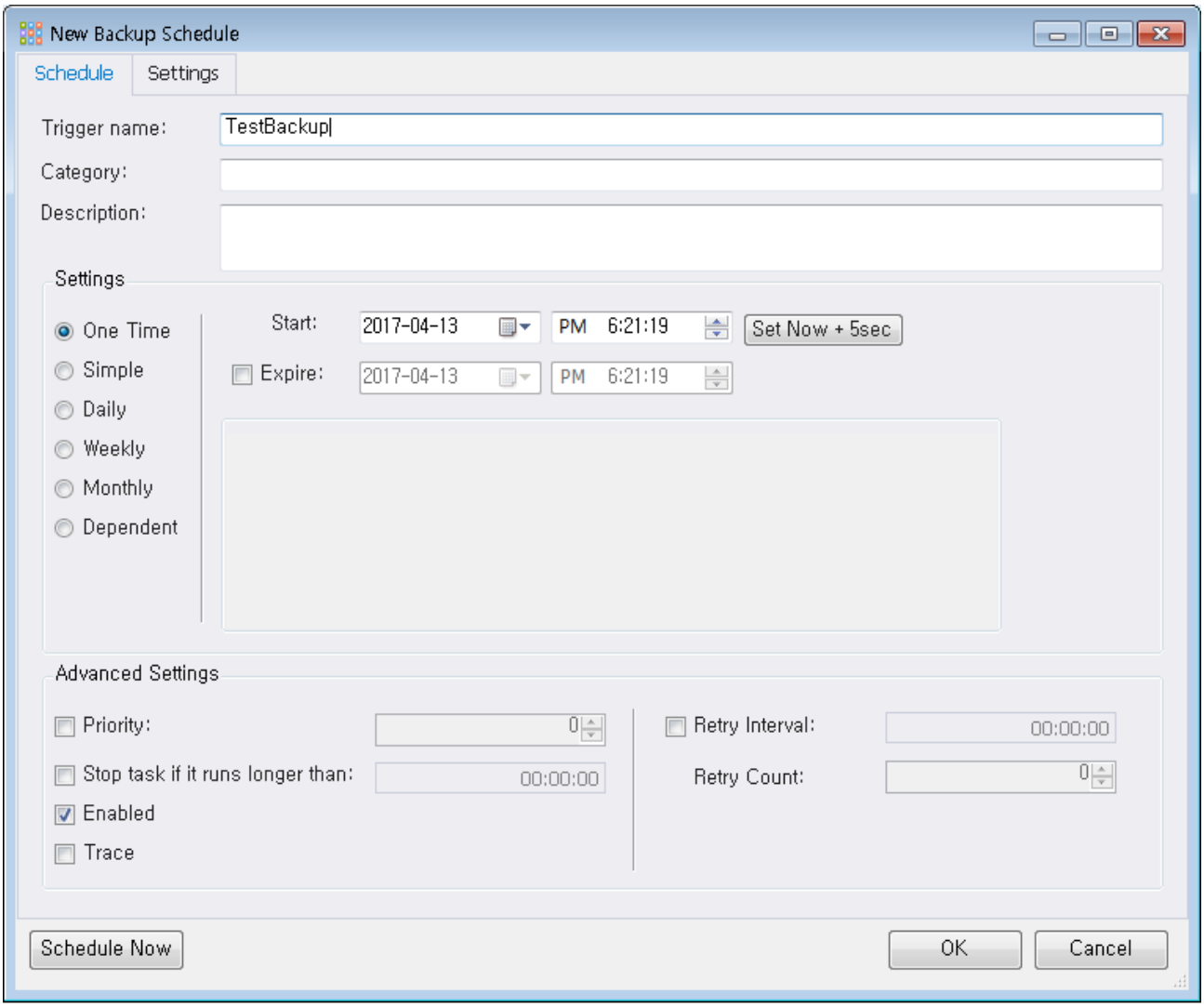
1. In the Server Explorer, select the target server for Backup registration.

2. Double-click the target server's Backup to activate the Backup query screen.

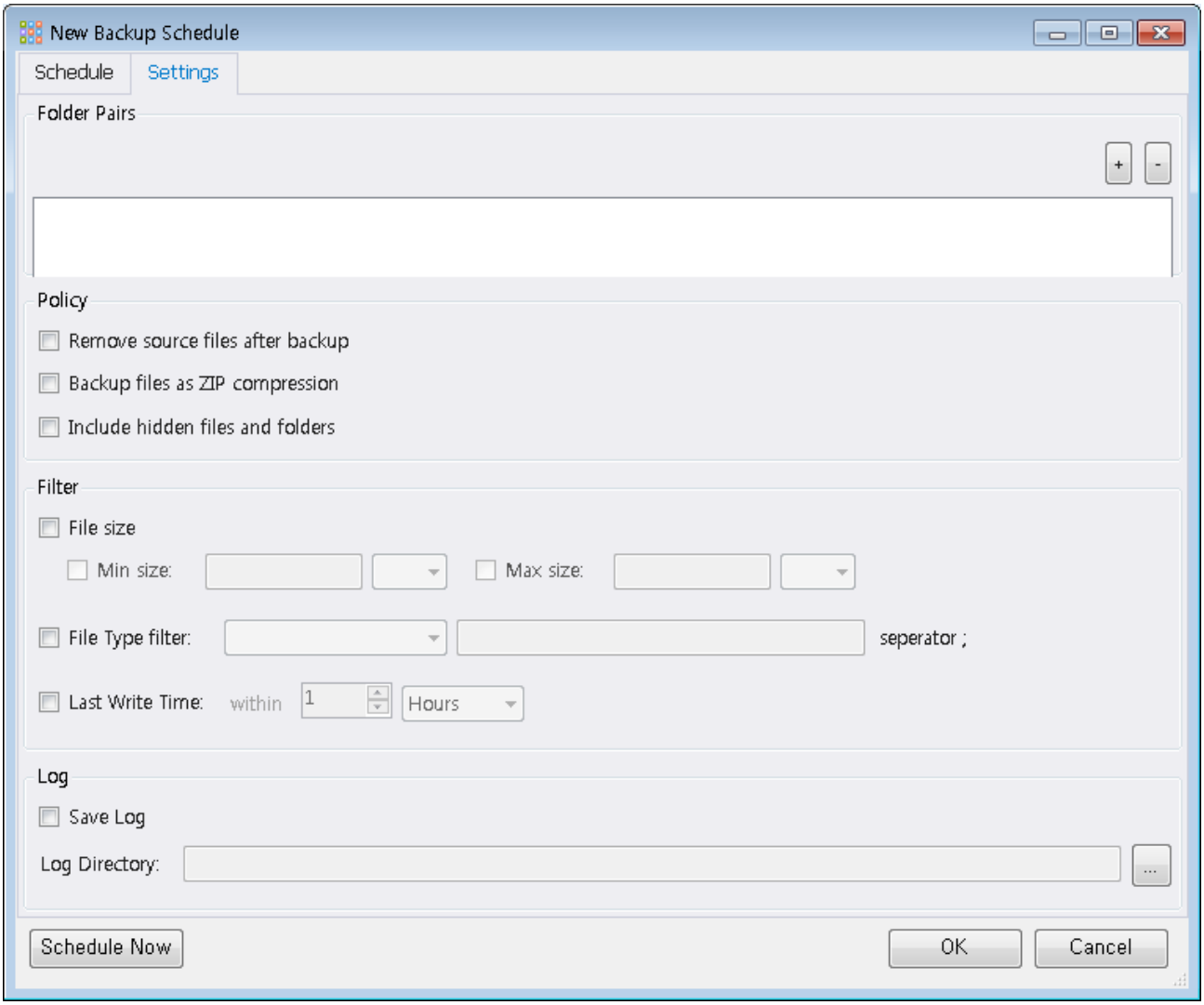
3. Click the **[Add]** button on the top menu bar.

4. In the New Backup Dialog, enter information for Schedule, Settings Tab.

5. Enter the information in Schedule Tab. The basic setting is the same as the **Trigger registration** method.

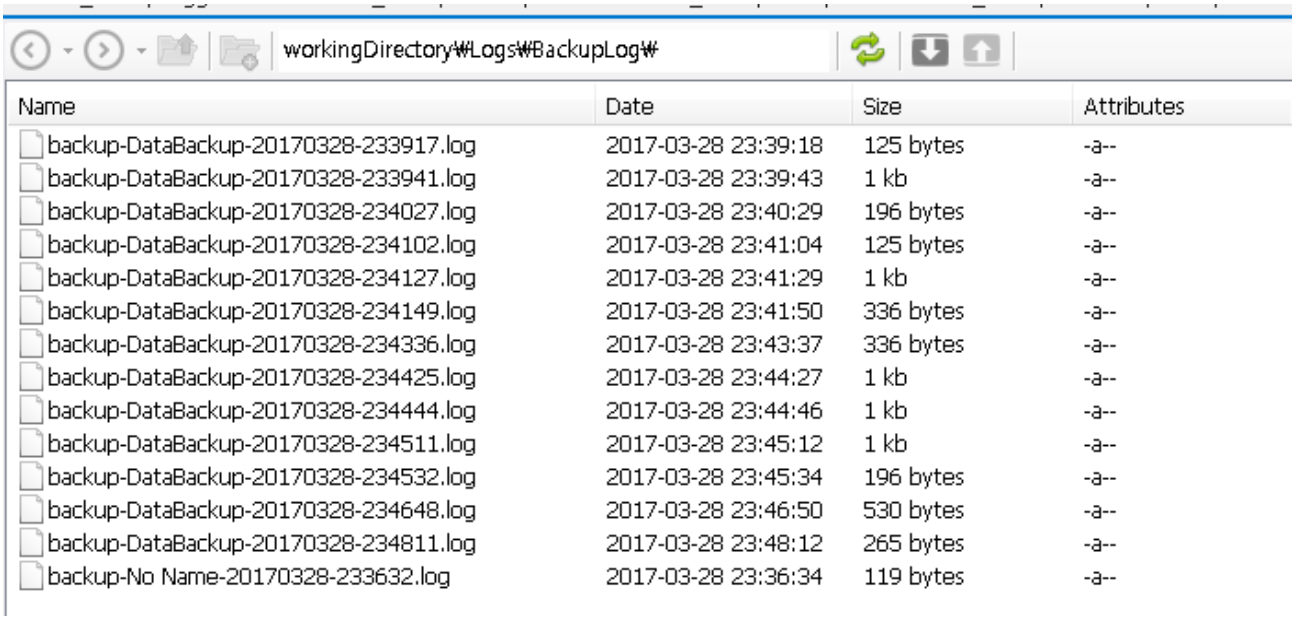


6. Enter the information in Settings Tab.



- **Folder Pairs** : This section is to set the source/destination folder for the backup. Source/Destination folder pairs are be added by clicking the + button and to delete the pair click the - button.
- **Policy** : Sets up the backup policy. Multiple selections are possible.
  - **Remove source files after backup** : Option whether to delete the source files/folders after backup.
  - **Backup files as ZIP compression** : Option whether to compress source files and save it to the destination.
  - **Include hidden files and folders** : Option whether to back up hidden files and folder from the source.
- **Filter** : Sets the filter conditions of Backup. Multiple settings are possible.
  - **File size** : You can limit the maximum/minimum size of the file to be backed up.
    - **Min size** : Any file size below Min size value will be excluded for backup (units: KB/MB/GB)

        - **Max size** : Any files size above Max size value will be excluded for backup (units: KB/MB/GB)
    - **File Type filter** : Backup files only or exclude from backup with the configured file extensions.
        - **including** : When this option is selected and the file type is entered, only the files with the corresponding type in the source path will be backed up. Separator is used as ';' when inputting multiple files.
        - **excluding** :When this option is selected and the file type is entered, the files with the corresponding type in the source path won't be backed up. Separator is used as ';' when inputting multiple files.
    - **Last Write Time** : Option whether to backup files that exceed the specified period. The basic unit is hour / day. (i.e. If the period is set as 1 Hour, the last modification date of the file that has passed one hour from the current date will be backed up.)
- **Log** : Option whether to leave logs for Backup
    - **Save Log** : Logs for backup are saved when the checkbox is enabled.
    - **Log Directory** : Set the log file path of Backup. The backup log file is saved as a text document in .log format and the file name format is **backup- [Backup Name] - yyyymmdd-hhmmss.log**.



> The backup history log path does not need to be in the WorkingDirectory. A node to check logs like Projects are not added to Server Explorer and users can view the history by registering a shortcut or accessing the path where the backup logs are saved.

Checking Backup

1. In Server Explorer, select the Backup Check Se.
2. Double-click the target server's backup file to activate the Backup screen.
3. In the Backup list, click the Backup Trigger that you want to view the Backup files / folders.
4. In the Destination Folders Tab, you can see the files and folders of the selected Backup.

Editing Backup Schedule

1. In Server Explorer, select the server to modify the backup schedule.
2. Double-click the target server's Backup node to activate the Backup screen.
3. Double-click Backup Trigger in the Backup.
4. Edit the Schedule / Settings Tab information and click the **[OK]** button.

Deleting Backup Schedule

1. In Server Explorer, select the server to delete the backup schedule .
2. Double-click the target server's Backup to activate the Backup screen.
3. Select the Backup Trigger to be deleted from the Backup.
4. Click **[Remove]** button on the top menu bar to delete Backup.

User Account and Authorization

User Account and Authorization

Credential information is required in order to access to the server via MMC. User with administration power can create user account and grant authority to restrict the usage of MMC functions according to user roles. The reason for separating authority is to avoid unnecessary incidents that could be caused by users rather than by the system. In this case, it is difficult to track the problem and sometimes leading to more serious ones. Default administrator account is created during MOZART Server installation (refer to **Server Management**). Other authorities should be granted manually and the following describes the authorities provided by MMC and its limitations.

- **Administrator** : This authority has full access to all the functions in MMC. Only administrator power can create/modify credentials and grant authority to user accounts.
- **Developer** : This authority has access to most of the functions in MMC except for creating/editing credentials and granting authority. Developer power is authorized to manage Projects and to distribute files.
- **Operator** : This authority is used mainly for Job/Trigger management and execution. Operator does not have the authority to do any task from Projects node.
- **Viewer** : This authority has the least access to MMC functions. Job/Triggers cannot be created or executed. This authority is mainly used when only information from Monitoring/Performance is required.

The following table shows which of the functions from MMC are permitted to use depending on the authority power.

Function Name	Viewer	Operator	Developer	Administrator
Connect Server	O	O	O	O
Edit Server Connection	O	O	O	O
Delete Server Connection	O	O	O	O
Projects node visiblity	O	O	O	O
Jobs node visibility	O	O	O	O
Triggers node visibility	O	O	O	O
System node visibility	O	O	O	O
Backup node visibility		O	O	O
Monitor node visibility	O	O	O	O
Performance node visibility	O	O	O	O
Shortcut node visibility	O	O	O	O
User node visibility		O	O	O
Deployable Folder node visibility	O	O	O	O

Refresh Job Schdeuler Status	O	O	O	O
Pause Job Scheduler		O	O	O
Resume Job Scheduler		O	O	O
License Information	O	O	O	O
Server Information	O	O	O	O
Add Project			O	O
View Project	O	O	O	O
Edit Project			O	O
Remove Project			O	O
Commit & Deploy			O	O
View History (Projects)	O	O	O	O
View Jobs list	O	O	O	O
View Job	O	O	O	O
Add/Remove/Edit Job		O	O	O

View History (Jobs)	O	O	O	O
Manage Job Types		O	O	O
Add/Delete/Edit Job Type		O	O	O
View Trigger list	O	O	O	O
View Trigger	O	O	O	O
Add/Remove/Edit Trigger		O	O	O
View History (Triggers)	O	O	O	O
View Trigger Execution log	O	O	O	O
View Backup trigger list		O	O	O
View Backup trigger		O	O	O
Add/Delete/Edit Backup trigger		O	O	O
View Destination folder		O	O	O
Download from Destination folder		O	O	O
Add/Remove/Edit Monitor view	O	O	O	O

See Monitor view	O	O	O	O
Stop Trigger		O	O	O
View Performance	O	O	O	O
View Shortcut	O	O	O	O
Add/Remove/Edit Shortcut	O	O	O	O
Download file from Shortchut	O	O	O	O
View Users list		O	O	O
Add/Remove/Edit Users				O
View History (Users)		O	O	O
View Deployable folder (mozart.dir)	O	O	O	O
Download file from Deployable folder	O	O	O	O
Upload file from Deployable folder	O	O	O	O
Remove file from Deployable folder		O	O	O
End MMC	O	O	O	O

Please refer to **How to Register/Modify Users** for more details on how to create user account and grant authorization to the account.See Also

---

How to Register/Modify Users

Create User Credential

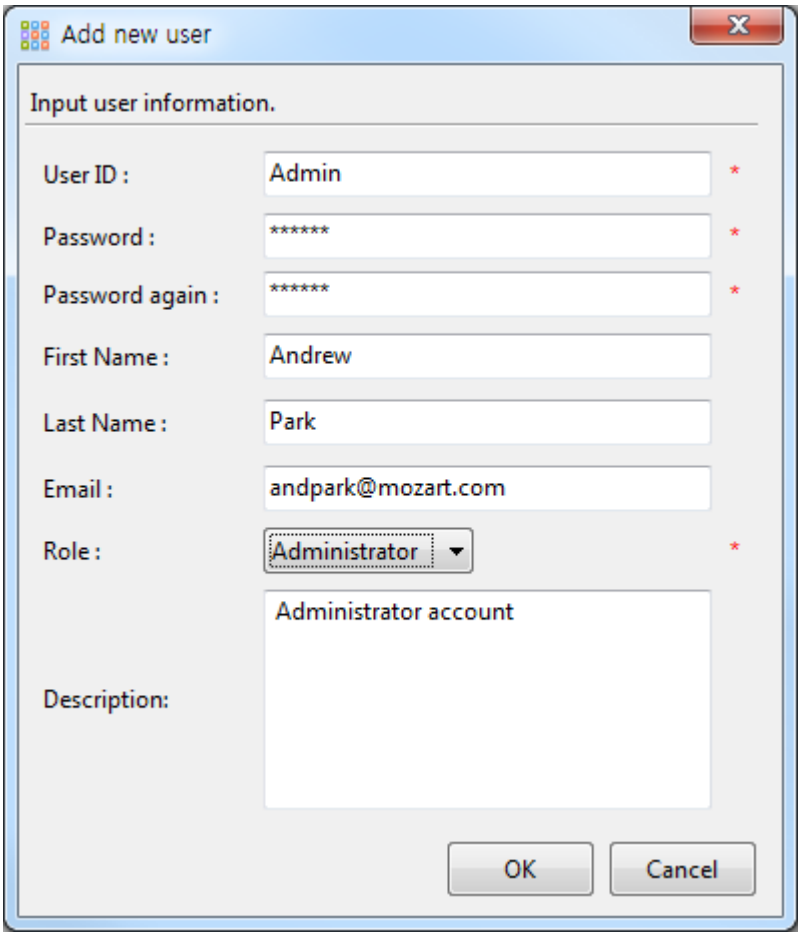
1. Select the server from Server Explorer to add user credential.

2. Connect to the server with administrator account.

3. Double click on Users to activate Users tab.

4. Click **[Add]** button from the top side menu bar.

5. Enter required information through Add new user Dialog.



- **User ID** : Enter the ID to be used to access to the server. *(Mandatory)*
- **Password** : Enter the password to be used to access to the server. *(Mandatory)*
- **Password again** : Re-enter the password entered in Password for validation. *(Mandatory)*
- **First Name** : Enter the first name of the owner of the account. *(Optional)*
- **Last Name** : Enter the last(sir) name of the owner of the account. *(Optional)*
- **Email** : Enter the e-mail address of the owner of the account. *(Optional)*

- **Role** : Select the user role from the list. For more details regarding authorization please refer to **User Account and Authorization**. *(Mandatory)*
- **Description** : Enter any additional information for the user account.

6. Click **[OK]** after entering all information.

Edit User Information

1. Select the server from Server Explorer to edit user credential.

2. Connect to the server with administrator account.

3. Double click on Users to activate Users tab.

4. Select the user to edit from the list, double click or click **[Edit]** button from the topside menu bar.

- UserID cannot be modified. In addition, the Role for default account 'sa' cannot be modified.
- If the Password textbox is empty during modification, the previously set password will be maintained.

Deleting Users

1. Select the server from Server Explorer to delete user credential.

2. Connect to the server with administrator account.

3. Double click on Users to activate Users tab.

4. Select the user account from the list and click **[Remove]** button to delete the user.

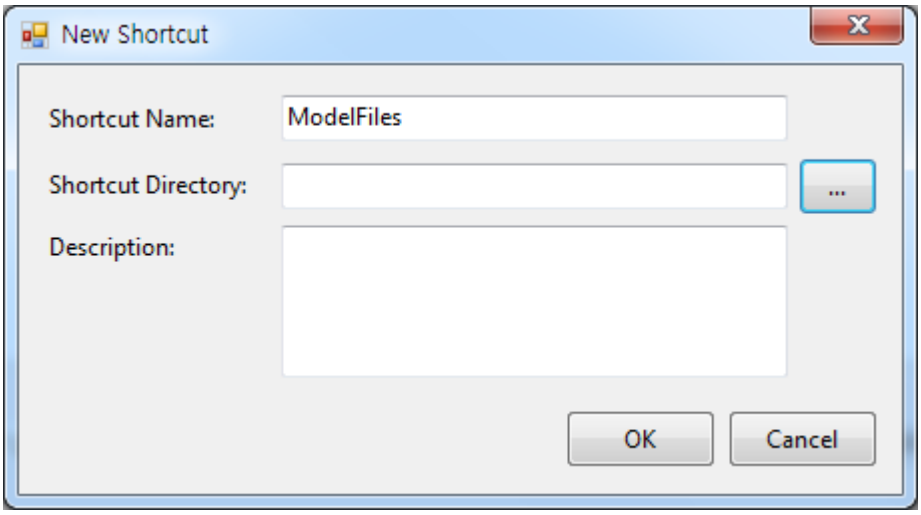
> Default account **'sa'** cannot be removed.

Managing Shortcut

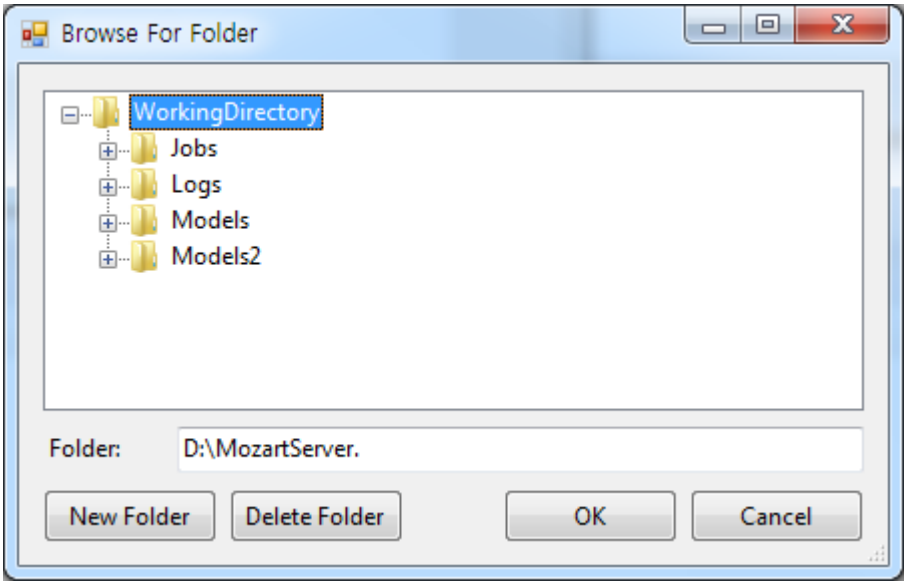
Shortcut a method for MMC users to access server folder (**Shortcut concept**). The Shortcuts are managed through Server Explorer.

Adding a Shortcut

1. In order to add a Shortcut, select a server node as the target to access.
2. Add a Shortcut by clicking **[Add Shortcut]** of pop-up menu or clicking icon at the top of Server Explorer.
3. Input each item in New Shorcut dialog.

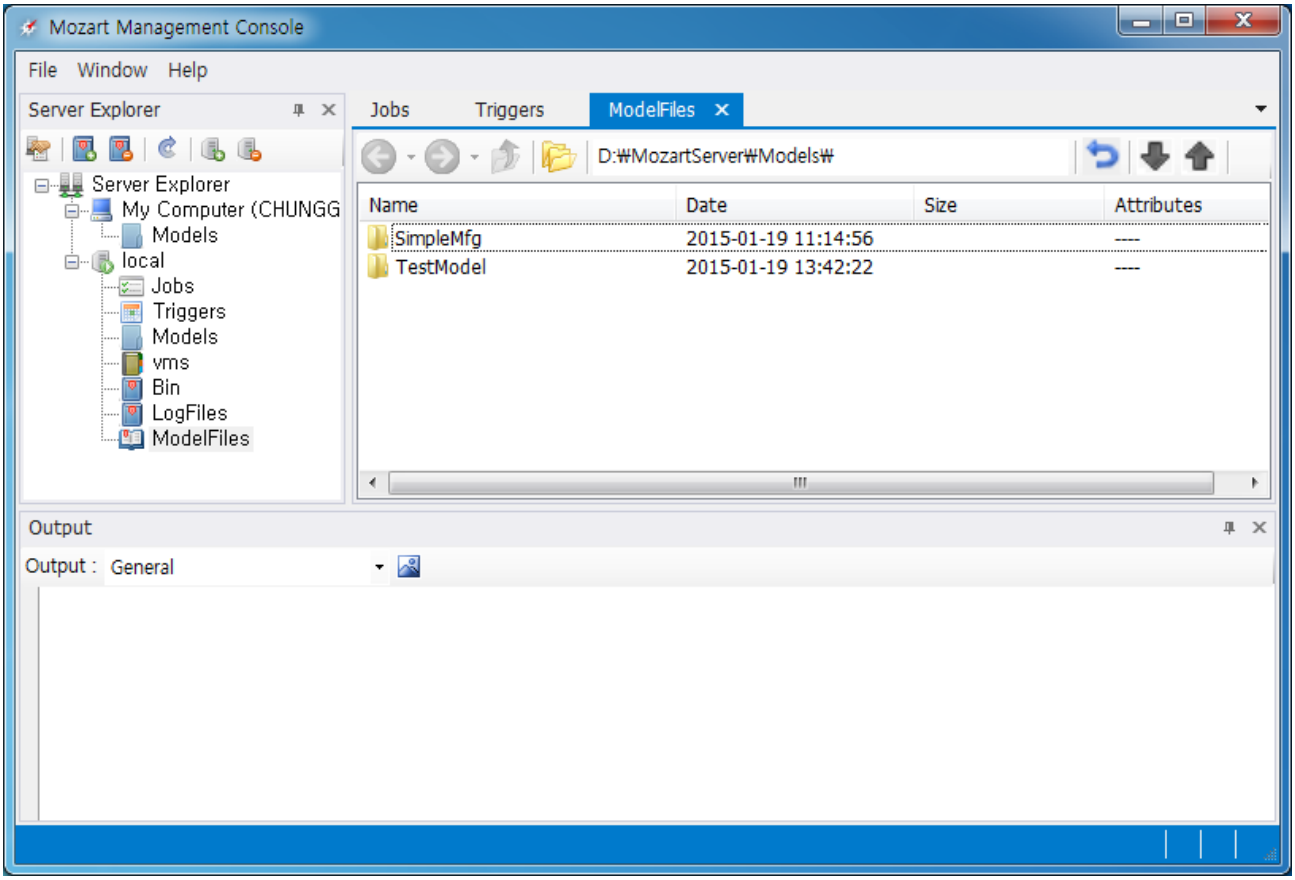


- **Shortcut name** : Shortcut ID. This is the name of displayed Shorcut.
- **Shortcut Directory** : Server folder that Shortcut is mapped on. Select a Server folder by using **[...]** button at the right side. WorkingDirectory is the designated folder when MOZART server is installed. Shourcut can be created only in Working Directory.



- **Description** : Description of Shortcut.

4. The created Shortcut can be accessed by executing **[Open]** command in pop-up menu or double-clicking the shortcut. This is similar to Windows Explorer and the folders subordinate to the correspond Shortcut can be explored.



Modifying a Shortcut

1. Select a target Shortcut that will be modified and execute **[Edit Shortcut]** in pop-up menu.

2. Modify input item when the above Shortcut is added. Modify the name and Directory, etc.

Deleting Shortcut

1. Select a target Shortcut that will be deleted and execute **[Delete Shortcut]** in pop-up menu or click Shortcut delete icon  in Server Explorer. Then, the Shortcut will be deleted.

Configuring Server default Shortcut

The shortcut information is designed to be set separately by each user and use PC . However, if a Shortcut is shared in Server for common use by all MMC User, this can be configured in Server. Refer to **Installing MOZART Server/Configure Default Shortcut**.