

# Automatic deployment toolkit for an SOA project environment, Part 3: IBM DB2 and IBM DB2 Content Manager automatic installation scripts

Skill Level: Intermediate

[HeQing \(Hawking\) Guan \(guanheq@cn.ibm.com\)](mailto:guanheq@cn.ibm.com)

Staff Software Engineer

IBM

[Qiang Bai \(baiqiang@cn.ibm.com\)](mailto:baiqiang@cn.ibm.com)

Software Engineer

IBM

[YuLin Chen \(chenyul@cn.ibm.com\)](mailto:chenyul@cn.ibm.com)

Software Engineer

IBM

[YaoFei \(Richard\) Zhu \(zhuyaof@cn.ibm.com\)](mailto:zhuyaof@cn.ibm.com)

Staff Software Engineer

IBM

17 Oct 2008

This article [series](#) introduces an automatic deployment toolkit (Automatic-DT), which helps infrastructure architects install and configure deployment nodes with a list of IBM software installed and configured automatically. It also helps testers and developers refresh builds in their daily tests and integration life cycle. In this article, Part 3 in the series, build automatic installation and configuration scripts on IBM® DB2® Universal Database and IBM DB2 Content Manager.

## Introduction

This article shows you how to automatically set up DB2 Universal Database and DB2 Content Manager. The automatic installation scripts are written in Python, which enables these two products to be installed without human intervention.

The setup of each product is divided into three main steps:

- **Pre-installation** steps are performed to check whether the requirements of the product are met.
- **Installation** starts.
- **Post-installation** steps are executed after installation is complete to verify a successful installation and to allow for some extra tasks to be done based on project requirements; for example, it might check the installation logs to confirm that no error occurred in the installation or execute some DB2 SQL scripts to set up a DB2 database for development.

## Response file

To set up a product like DB2 automatically, you need a silent installation. In a silent installation, you use response files to specify the configuration parameters that you'd normally type in the installer's GUI. A *response file* is a text file with one line for each input field that you'd find in the regular interactive installation.

In general, there are two ways to prepare a response file:

- **Saving:** When performing manual installation, you can have the installer generate one, recording the options that you've selected during the installation, and then make some small modifications as needed. This is quite useful if you're installing the exact same components on a large number of machines.
- **Editing:** Modify a response file from a common template. A template response file displays all the possible options that you can specify, but it's more difficult because you must be an expert on the response file template.

In the automatic installation scripts, several response files are already provided. You can use them directly, make some modifications, or users can overwrite the response file. An example of a DB2 response file is shown in Listing 1.

### Listing 1. Example of a DB2 response file

```
PROD=ENTERPRISE_SERVER_EDITION
LIC_AGREEMENT=ACCEPT
# installation destination dir
FILE=C:\IBM\SQLLIB\

INSTALL_TYPE=CUSTOM

COMP=SPATIAL_EXTENDER_SAMPLES
COMP=INFORMATION_CATALOG_SAMPLES
```

```

COMP=JDBC_SUPPORT
COMP=IBM_JDK
COMP=IBM_JRE
COMP=LDAP_EXPLOITATION
... ..
COMP=TCPIP_DB2_LISTENER_SUPPORT

LANG=EN
DAS_CONTACT_LIST=LOCAL

INSTANCE=DB2
DB2.NAME=DB2
DEFAULT_INSTANCE=DB2
DB2.AUTOSTART=YES
#db2 user name and password
DB2.USERNAME=db2admin
DB2.PASSWORD=244259359291351148346332
... ..
DB2_ADMINGROUP_NAME=DB2ADMNS

```

## DB2 setup

As discussed in the Introduction, complete the following three main steps—pre-installation, installation, and post-installation—to finish DB2 setup.

### DB2 pre-installation

1. Make sure there's a DB2 product installed in the destination directory, as shown in Listing 2.

#### Listing 2. DB2 pre-installation

```

#1. Make sure there's a DB2 product installed in the destination
dir.
def checkDB2Installed():
    filename = "db2level_output.txt";
    db2level = os.path.join(db2BinDir, "db2level.exe")
    executeFile = db2level + " > " + filename
    os.system(executeFile)
    if(os.path.exists(filename)):
        f = file(filename)
        for line in f:
            if(line.startswith("Product is installed at \"" + db2BinDir +
"\\")):
                return True
        return False

```

### DB2 installation

2. Make sure the DB2 response file is already prepared.
3. Execute the DB2 silent installation, as shown in Listing 3.

#### Listing 3. DB2 installation

```

#execute DB2 installation
def executeDB2Installation() :
    #1. Make sure the response file is prepared.
    if os.path.exists(responseFile):
        executeFile = executeFile + " -f -l " + db2_log + " -u " +
responseFile
    #2. execute silent installation
    os.system(executeFile)
    return True
    else :
        log(responseFile + " is invalid!")
        return False

```

## DB2 post-installation

4. Make sure the installation is successful. Here you use the operation `checkDB2Installed`, shown in [Listing 2](#).
5. Try to find and catalog existing databases on the local hard disk, as shown in Listing 4. This task is useful to recover the existing data in the scenarios that a DB2 server has removed and then reinstalled.

### Listing 4. Find and catalog existing databases on the local hard disk

```

#2. Find and catalog already existed databases on the local hard
disk
def addExistedDatabase():
    self.executeDB2Command("list database directory on " +
db2InstalledBaseDir)
    if(os.path.exists(db2_cmd_log)) :
        f = file(db2_cmd_log)
        for line in f:
            #find the database name in the line, denoted by name1
            databaseList.append(name)
            self.executeDB2Command("catalog database " + name1)
        return True
# execute "db2cmd db2 command"
def executeDB2Command() :
    if( os.path.exists(db2cmdFile())) :
        #run db2cmd db2 -z log command
        cmd = "start /D" + db2BinDir + " /I /WAIT " + db2cmdFile
+ " /w " + "db2 " + "-z" + db2_cmd_log + " " + command
        os.system(cmd)
        return self.checkCommandExeResult(db2_cmd_log)
    return False
# See if the result of "db2cmd db2 command" is successful
def checkCommandExeResult(self, logfile = ""):
    if(os.path.exists(logfile)):
        f = file(logfile)
        for line in f:
            if line.find("DB20000I") != -1 :
                return True
        return False
    return False

```

6. There are also some extra tasks based on the project requirements (see Listing 5), such as creating tables, preparing needed data, or creating a

DB2 instance.

**Listing 5. Extra tasks based on the project requirements**

```
def doMoreTasks():
    db2.createDatabase('SampleDB')
    db2.executeSQLScripts('Sample.sql')

#create database databaseName
def createDatabase():
    command = "create database " + databaseName
    return self.executeDB2Command(command)

#execute sql scripts
def executeSQLScripts(self, filename) :
    #run db2cmd /w db2 -tf filename -z logfile
    cmd = "start /D" + db2BinDir
        + " /I /WAIT " + db2cmdFile + " /w " + "db2 -tf "
        + filename + " -z " + logfile
    os.system(cmd)
    if(os.path.exists(logfile)):
        logfilef = file(logfile)
        for line in logfilef:
            #Find an error in the sql execution
            if( not line.startswith("DB20000I")):
                return False
        return True
    return False
```

Table 1 shows detailed information about how to use Python and DB2 commands in your DB2 automatic installation scripts.

**Table 1. Detailed information about Python and DB2 commands**

Steps	Task	Python functions	DB2 commands
Pre-installation	Make sure there's a DB2 product installed in the destination directory.	os.system(...)	db2level
Installation	Confirm that the response file has already been prepared.	os.path.exists(...)-	
	Execute the DB2 silent installation.	os.system(...)	db2setup -f -l db2.log -u responseFile.txt

<b>Post-installation</b>	Make sure the installation was successful.	<code>os.system(...)</code>	<code>db2level</code>
	Find and catalog the existing databases on the local hard disk.	<code>os.system(...)</code>	(1) <code>db2cmd db2</code> (2) <code>LIST DATABASE DIRECTORY [ON drive]</code> (3) <code>CATALOG DATABASE database-name [AS alias] [ON drive]</code>
	Perform extra tasks based on the project requirements, such as creating tables, preparing needed data, or creating a DB2 instance.	<code>os.system(...)</code>	<code>db2 -tf filename -z logfile</code> <code>db2 CREATE DATABASE database-name</code>

## WebSphere Application Server V5.1.1 setup

The WebSphere Application Server V5.1.1 setup isn't the focus of this article, but we're introducing it here because DB2 Content Manager needs WebSphere Application Server V5.1.1. So the following section covers just enough of the WebSphere Application Server installation and configuration to show you how it supports the DB2 Content Manager installation. (For more information about WebSphere Application Server V5.1.1 installation and configuration, visit the [WebSphere Application Server information center](#).)

### WebSphere Application Server pre-installation

1. Make sure there's a WebSphere Application Server V5.1.1 product already installed in the destination directory, as shown in Listing 6.

**Listing 6. Make sure there's a WebSphere Application Server V5.1.1 product already installed in the destination directory**

```
#Make sure there's a WebSphere Application Server V5.1.1
product already installed in the destination dir
def checkWASInstalled():
    versionInfo = WASBinDir + "\\versionInfo.bat"
    if(not os.path.exists(versionInfo)):
        return False
    versionLog = "versionInfo.log"
    os.system(versionInfo + " > " + versionLog)
    installed = False;
    if os.path.exists(versionLog):
        f = file(versionLog, 'r')
        #Read all content in versionLog, and try to find version in the
        versionLog
        for line in f:
```

```

    if line.find(version) != -1:
        installed = True
    f.close()
    return installed

```

## WebSphere Application Server installation

2. Make sure that the WebSphere Application Server response file is prepared.
3. Execute the WebSphere Application Server V5.1.0 silent installation.

### Listing 7. Execute WebSphere Application Server V5.1.0 silent installation

```

def installWAS(self, executeFile, responseFile) :
    #1. Make sure the WebSphere Application Server response file is
    prepared.
    if os.path.exists(responseFile):
        executeFile = executeFile + " -options " + responseFile
    #2. Execute WebSphere Application Server 5.1.0 silent
    installation.
    os.system(executeFile)
    return True
    else :
    log( responseFile + " is invalid!" )
    return False

```

4. Make sure WebSphere Application Server V5.1.0 is installed successfully. Here you use the operation `checkWASInstalled`, shown in [Listing 6](#).
5. Execute the WebSphere Application Server V5.1.1 fix pack silent update.

### Listing 8. Execute WebSphere Application Server V5.1.1 fix pack

```

#4. Install WebSphere Application Server 5.1.1 fix pack silent
update.
def installWASFixPack() :
    log( "start to install the fixpack of WAS ...." )
    logfile = "fixpack.log";
    command = FixpackExecutableFile
    + " -fixpack -install -installDir " + "\""
    + WASInstalledDir + "\"\"\"\"
    + " -fixpackDir " + "\"" + FixpackDir
    + "\"\" + " -fixpackID " + FixpackID
    + " -ihsInstallDir " + IHSInstalledDir
    + " -skipMQ " + "> " + logfile
    os.system(command)
    log( "The installation is ended!" )
    return True

```

## WebSphere Application Server post-installation

6. Make sure that WebSphere Application Server V5.1.1 has been installed successfully. Here you use the operation `checkWASInstalled`, shown in [Listing 6](#).

Table 2 shows detailed information about how you use Python and WebSphere Application Server commands in your WebSphere Application Server automatic installation scripts.

**Table 2. Detailed information about Python and WebSphere Application Server commands**

Steps	Task	Python functions	WebSphere Application Server commands
<b>Pre-installation</b>	Make sure a WebSphere Application Server V5.1.1 product is installed in the destination directory.	<code>os.system(...)</code>	<code>versionInfo.bat</code>
<b>Installation</b>	Confirm that the response file has already been prepared.	<code>os.path.exists(...)</code>	-
	Execute the WebSphere Application Server V5.1.0 silent installation.	<code>os.system(...)</code>	<code>win\Install.exe -options responseFile.txt</code>
	Execute the WebSphere Application Server V5.1.1 fix pack silent update.	<code>os.system(...)</code>	<code>fixpackDir\updateSilent.bat -fixpack -install -installDir WAS511InstalledDir -fixpackDir fixpackDir\fixpacks -fixpackID "was51_fp1_win" -ihsInstallDir IHSInstalledDir -skipMQ &gt; logfile</code>
<b>Post installation</b>	Check to see if the installation was successful.	<code>os.system(...)</code>	<code>versionInfo.bat</code>

## DB2 Content Manager setup

This section walks you through the steps involved in DB2 Content Manager setup. DB2 Content Manager depends on a DB2 server and WebSphere Application Server V5.1.1. So in the pre-installation of DB2 Content Manager, you also complete steps

for DB2 setup and WebSphere Application Server setup.

## DB2 Content Manager pre-installation

1. Make sure DB2 Content Manager is installed in the destination directory.
2. Make sure that DB2 and WebSphere Application Server V5.1.1 have been installed. If they have not been installed, please go perform the installation steps discussed in the [DB2 setup](#) and [WebSphere Application Server V5.1.1 setup](#) sections.

### Listing 9. DB2 Content Manager pre-installation

```
#1. Make sure DB2 Content Manager is installed in the destination
dir.
def checkDB2CMIInstallation():
    foundedString = "IBM DB2 Content Manager Enterprise Edition"
    if(code == "IIC"):
        foundedString = "IBM DB2 Information Integrator for Content"
    elif(code == "EC"):
        foundedString = "IBM DB2 Content Manager eClient"
    db2cmlevel = CMBinDir + "\\cmlevel.bat" > filename
    if(os.path.exists(db2cmlevel)) :
        os.system(db2cmlevel)
        if(os.path.exists(filename)):
            f = file(filename)
            for line in f:
                if(line.startswith(foundedString)):
                    log("The " + code + " has been installed")
                    return True
            log("The " + code + " has not been installed")
            return False

#2. Make sure DB2 and WebSphere Application Server V5.1.1 are
installed
def checkPreRequisit(self):
    if(not checkWASInstalled()):
        log("WAS 511 has not been installed")
        return False
    if(not checkDB2Installed()):
        log("DB2 has not been installed")
        return False
    return True
```

## DB2 Content Manager installation

3. Start the WebSphere Application Server V5.1.1 server called server1.  
**Listing 10. Start the WebSphere Application Server V5.1.1 server called serverName**

```
def startServer(serverName):
    if(not self.serverStatus(serverName)):
        log("try to start " + serverName)
        logfile = "startServer.log"
        startfile = WASBinDir + "\\startServer.bat"
        os.system(startfile)
```

```
return True
```

4. Make sure that the DB2 Content Manager response file is prepared.
5. Execute the DB2 Content Manager silent installation.
6. Make sure that DB2 Content Manager is installed successfully.

#### Listing 11. DB2 Content Manager installation

```
def installCM() :
    executeDir = CMSrcInstallDir
    executeFile = CMSrcInstallDir + "\\setup-cm-8.3.00.exe"
    responseFile = CM_response_file
    logfile = 'cminstall-rc.txt'
    successString = "0"
    self.installDB2CMPProduct(executeDir, executeFile, responseFile)
    # Check that the DB2 Content Manager is installed successfully.
    if(os.path.exists(logfile)) :
        f = file(logfile)
        for line in f:
            if(line.startswith(successString)):
                return True
    return False

def installDB2CMPProduct(executeDir, executeFile, responseFile) :
    if os.path.exists(responseFile)
        and os.path.exists(executeFile):
        executeFile = executeFile + " -is:javahome "
            + executePath + "java\\jre"
            + " -options " + responseFile\
            + " -silent"
        #execute db2cm product silent instalation
        os.system(executeFile)
        return True
    else :
        log( responseFile + " or "
            + executeFile + " is invalid!" )
        return False
```

7. Start the WebSphere Application Server V5.1.1 server named icrm, which is added by the DB2 Content Manager installation.
8. Make sure that the IBM DB2 Information Integrator for Content response file is prepared.
9. Execute the DB2 Information Integrator for Content silent installation.
10. Make sure that DB2 Information Integrator for Content is installed successfully.

#### Listing 12. DB2 Information Integrator for Content installation

```
def installIIC() :
    executeDir = IICSrcInstallDir
    executeFile = IICSrcInstallDir + "\\setup-ii4c-8.3.00.exe"
    responseFile = IIC_response_file
```

```

logfile = 'ii4cinstall-rc.txt'
successString = "0"
if(not startServer("icrm")): #5. Start WebSphere Application
Server
    V5.1.1 server named icrm.
    log("Fail to start icrm, please take a look at the WebSphere
server")
    return False
self.installDB2CMPProduct(executeDir, executeFile, responseFile)
# Check that the DB2 Information Integrator for
Content is installed successfully.
if(os.path.exists(logfile)) :
    f = file(logfile)
    for line in f:
        if(line.startswith(successString)):
            return True
    return False

```

11. Make sure that the DB2 Content Manager eClient response file has been prepared.
12. Execute the DB2 Content Manager eClient silent installation.
13. Make sure that the DB2 Content Manager eClient is installed successfully.

#### Listing 13. DB2 Content Manager eClient installation

```

def installEC() :
    executeDir = ECSrcInstallDir
    executeFile = ECSrcInstallDir + "\\setup-ec-8.3.00.exe"
    responseFile = EC_response_file
    logfile = 'ecinstall-rc.txt'
    successString = "0"
    if(not startServer("icrm")): #5. start was 511 server named icrm.
        log("Fail to start icrm, please take a look at the WebSphere
server")
        return False
    self.installDB2CMPProduct(executeDir, executeFile, responseFile)
    # Make sure the DB2 Content Manager eClient is installed
    successfully.
    if(os.path.exists(logfile)) :
        f = file(logfile)
        for line in f:
            if(line.startswith(successString)):
                return True
    return False

```

## DB2 Content Manager post-installation

Because you've already checked the installation result in the [DB2 Content Manager installation](#) section, there are no more tasks in this step.

Table 3 shows detailed information about how you use Python, WebSphere Application Server, and DB2 Content Manager commands in your DB2 Content Manager automatic installation scripts.

**Table 3. Detailed information about Python, WebSphere Application Server, and DB2 Content Manager commands**

Step	Task	Python function	DB2 Content Manager command
Pre-installation	Make sure there's a DB2 Content Manager product installed in the destination directory.	<code>os.path.exists(...)</code>	<code>cmlevel.bat</code>
	Make sure WebSphere Application Server is started.	<code>os.system(...)</code>	<code>startServer [serverName]</code> [serverName] can be <code>server1</code> or <code>icmm</code> . This is a WebSphere Application Server command.
	Make sure DB2 is installed.	This depends on other Python modules that were developed earlier in the article. See the operation <code>checkDB2Installed</code> in <a href="#">Listing 2</a> .	
Installation	Execute DB2 Content Manager silent installation.	<code>os.system(...)</code>	<code>[executeFile]</code> <code>-is:javahome</code> <code>executePath</code> <code>java\jre</code> <code>-options</code> <code>responseFile.txt</code> <code>-silent</code> [executeFile] can be <code>setup-cm-8.3.00.exe</code> , <code>setup-ii4c-8.3.00.exe</code> , or <code>setup-ec-8.3.00.exe</code> due to the code (CM, IIC, or EC).
	Post-installation	Check the installation logs to confirm that the DB2 Content Manager installation is finished with no errors.	<code>file(...)</code> -

## Conclusion

This article introduced how to automatically install DB2 and DB2 Content Manager, and how to make use of Python. After finishing your design and implementation, you applied those scripts to prepare development and test environments for other team members. You've seen that it saves time and that it's easy to use. Plus, you were able to do all of this installation using only one command!

# Resources

## Learn

- Get more information about [Python](#).
- Visit the [DB2 Information Center](#), [DB2 Content Management Information Center](#), and [WebSphere Application Server V5.1.1 Information Center](#).
- The [SOA and Web services zone](#) on IBM developerWorks hosts hundreds of informative articles and introductory, intermediate, and advanced tutorials on how to develop Web services applications.
- Play in the [IBM SOA Sandbox!](#) Increase your SOA skills through practical, hands-on experience with the IBM SOA entry points.
- The [IBM SOA Web site](#) offers an overview of SOA and how IBM can help you get there.
- Stay current with [developerWorks technical events and webcasts](#).
- Browse for books on these and other technical topics at the [Safari bookstore](#).
- Check out a quick [Web services on demand demo](#).
- Get an [RSS feed for this series](#). (Find out more about [RSS](#).)

## Get products and technologies

- Download [IBM product evaluation versions](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

## Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

## About the authors

HeQing (Hawking) Guan

HeQing Guan (Hawking) is a senior software engineer on the IBM Global Business Solution Center team. He has a doctorate from the Chinese Academy of Sciences and has worked in the SOA field for almost 5 years.

## Qiang Bai

Qiang Bai has worked as a software engineer with the Global Business Solution Center in CSDL. In addition to focusing on software configuration management (SCM) products, he is responsible for developing the solution assets built on IBM's Service-Oriented Architecture (SOA) methodology.

---

## YuLin Chen

YuLin Chen is a software engineer in the IBM Global Business Solution Center. He is responsible for developing the solution assets built on IBM's Service-Oriented Architecture (SOA) methodology.

---

## YaoFei (Richard) Zhu

YaoFei Zhu (Richard) joined the IBM China Development Lab in 2005 and has worked in Linux on POWER and on Global Business Solution Center (GBSC) teams as a system developer, application developer, and infrastructure architect. He has more than 6 years of experience in AIX, Linux, System p, System x, storage, and SOA.

## Trademarks

IBM, the IBM logo, ibm.com, DB2, developerWorks, Lotus, Rational, Tivoli, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. See the current list of [IBM trademarks](#).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.