|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Home_logo2 | **文件名称** | DBMOVE流程及规范 | **文件编号** | N/A |
| **当前版本号** | V0.1 | **安全级别** | 限公司内部使用 |
| **新蛋中国有限公司** | | | |

DBMove流程规范

Contents

[1 DBmove 4](#_Toc356573320)

[1.1 什么是DBMOVE? 4](#_Toc356573321)

[1.2 DBMOVE作用 4](#_Toc356573322)

[1.3 DBMOVE 流程 5](#_Toc356573323)

[2 数据库环境说明 5](#_Toc356573324)

[2.1 测试环境（QC） 5](#_Toc356573325)

[2.2 开发环境（DEV） 6](#_Toc356573326)

[2.3 成都本地环境（CD） 6](#_Toc356573327)

[2.4 产品环境（PRD） 6](#_Toc356573328)

[2.5 产品环境(在QC下模拟) 6](#_Toc356573329)

[3 DBMOVE 的填写 7](#_Toc356573330)

[3.1 Database Move to QC &Production。 7](#_Toc356573331)

[3.2 MOVE IN QC和MOVE IN PRD 请求详情 8](#_Toc356573332)

[3.3 database request 8](#_Toc356573333)

[3.3.1 server 的填写 8](#_Toc356573334)

[3.3.2 alter table/view 11](#_Toc356573335)

[3.3.3 create index 12](#_Toc356573336)

[3.3.4一个脚本对应多个对象情况 12](#_Toc356573337)

[3.3.5 create job 13](#_Toc356573338)

[3.3.6 alter job 13](#_Toc356573339)

[3.4 replication request 14](#_Toc356573341)

[3.5 Others 14](#_Toc356573342)

[3.6 其他注意事项 15](#_Toc356573343)

[4 DBCC的填写 15](#_Toc356573344)

[4.1 DBCC No 15](#_Toc356573345)

[4.2 basic info of project 16](#_Toc356573346)

[4.3 detailed info of Change 16](#_Toc356573347)

[4.4 Evaluation and Analysis the Change 17](#_Toc356573348)

[4.5 change Confirm 18](#_Toc356573349)

[4.6 其他注意事项 18](#_Toc356573350)

[5 DBMove ADD in的使用 19](#_Toc356573351)

[5.1 Code Checker 19](#_Toc356573352)

[5.2 Replication 20](#_Toc356573353)

[5.3 object viewer 22](#_Toc356573354)

[5.4 Formatter 23](#_Toc356573355)

[5.5 Guide 23](#_Toc356573356)

# DBmove

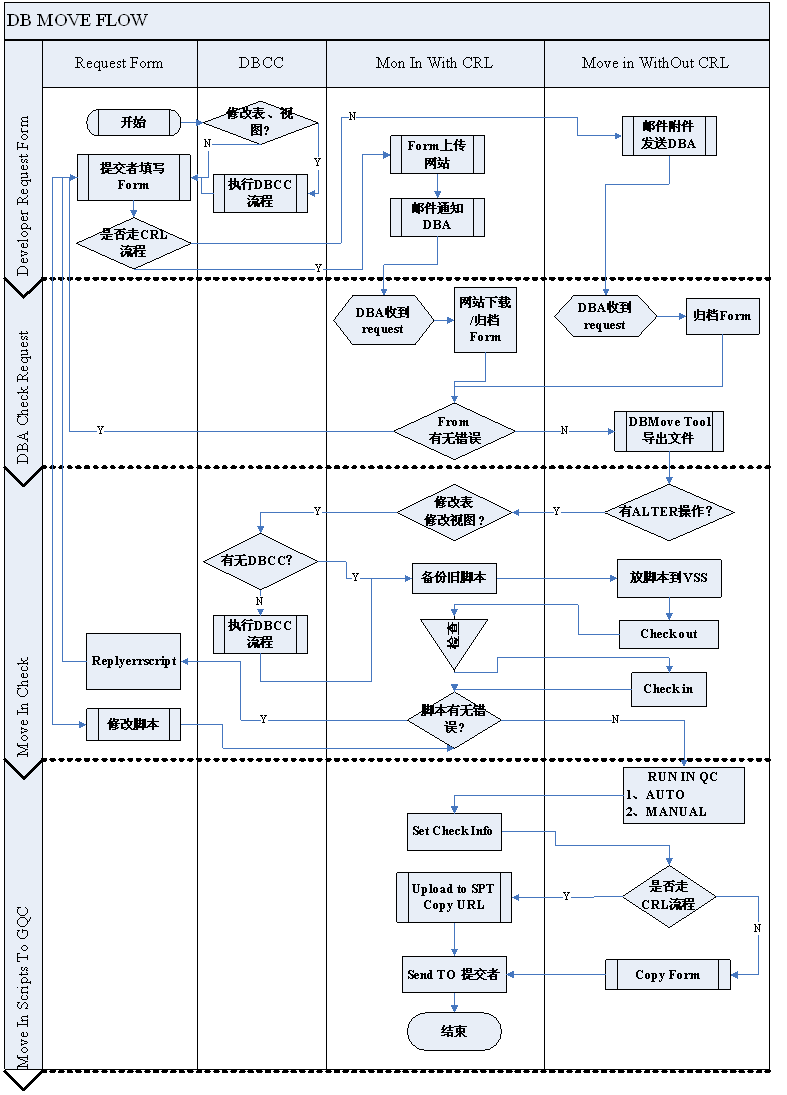
## 什么是DBMOVE?

DB MOVE工作是将开发团队编写的脚本提交到测试环境进行测试，测试环境通过后提交到产品环境，这样的一个工作流程叫DBMOVE

## DBMOVE作用

1. 确保提交到产品环境脚本的规范化
2. 确保提交到产品环境脚本的高效性

## DBMOVE 流程



# 数据库环境说明

## 测试环境（QC）

测试环境是对产品环境的一个模拟，且在这个环境下执行的任何动作基本都要经过DBA审核然后由DBA来执行。开发组提交来的脚本首先在他们本地或开发环境测试好，然后进一步提交form到DBA，DBA首先在QC环境下对他们的脚本进行审核，审核通过后由DBA执行，这些脚本在测试环境下运行一段时间开发组觉得没有问题，最后才申请move in 产品环境，然后由DBA真正move in 到产品环境。以下大概的列出QC环境下的服务器和实例的一些信息，一般QC环境中一个机器上会安装多个实例，所以QC的servername命名格式为：机器名称\实例名称，且QC的机器名一般都是以S1Q开头的，这个也是我们根据servername辨别服务器是哪个环境的好方法。如下所附文档为：QC环境下服务器和实例的一些信息。



## 开发环境（DEV）

开发环境是提供给开发人员进行测试的平台。如下所附文档为：DEV环境下的服务器和实例的一些信息。一般DEV环境中一个机器上会安装多个实例，所以DEV的servername命名格式为：机器名称\实例名称，且DEV的机器名一般都是以S1D开头的。



## 成都本地环境（CD/XA/SH）

成都本地也有自己的实例供开发组在工作时使用，如下所附文档为：CD环境下服务器和实例的一些信息。一般CD环境中一个机器上会安装多个实例，所以CD的servername命名格式为：机器名称\实例名称，且CD的机器名一般都是以SCD开头的



## 产品环境（PRD）

产品环境指的是我们线上正在运行直接面向客户的环境，如下所附文档为：PRD环境下的服务器和实例信息。一般PRD环境中一个机器上只安装一个实例，而且是默认实例。所以PRD的servername命名直接是机器名，也就是我们一般用的实例名称。比如NEWSQL，S7OVSDB04



## 产品环境(在QC下模拟)

测试环境是对产品环境的一个模拟，但是测试环境的服务器实例和产品环境的服务器实例命名不是完全相同的，且产品环境和测试环境的服务器数量也是不同的，但是我们产品环境的实例和测试环境实例基本上都有对应的关系，如下所附文档为：PRD环境下的实例与QC环境下实例的对应关系。



# DBMOVE 的填写

Form的内容区域又分为6个小的区域，以下做详细介绍。

## Database Move to QC &Production。

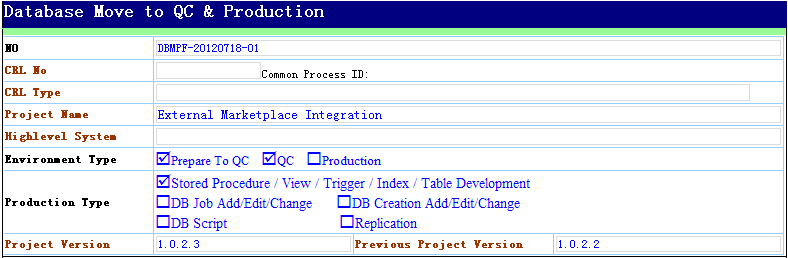


图3.4 database move to QC&Production列表详情

如图3.4所示：

**<NO>:**Form编号，组成：DBMPF-Submit time-Submitted order，例DBMPF-20120718-01；

**<CRL NO>:**如果走CRL流程，该项就需要填,CRL NO组成CRF-DB-ProjectName-Submit time +Submitted order，例如：CRF-DB-Update ComboSO check script prov-2012071705；

**<Common Process ID>:**进程ID；

**<CRL Type>:**走CRL流程时需要填写，填写CRL类型；

**<Project Name>:**此次Move脚本所属的Project名称；

**<Highlevel System>:**此Form所属的项目的紧急程度；

**<Environment Type>:**Move in的类型，包括以下三种类型：

**<Prepare To QC>:**做脚本检查，不做MOVE-IN操作；

**<QC>:**做脚本检查并做move in QC的操作；

**<Production>:**只做脚本检查和move到Production的预检查（Move到Production具体的操作由US DBA来完成）；

**<Production Type>:**脚本所包含的对象类型；

**<Project Version>:**项目当前版本；

**<Previous Project Version>:**项目之前的版本。

## MOVE IN QC和MOVE IN PRD 需求详情

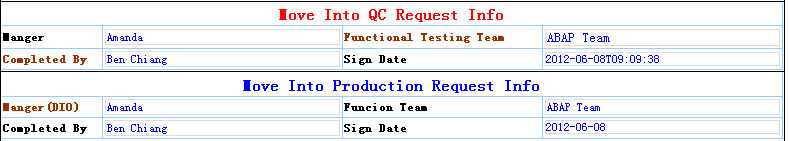


图3.5 move in QC 和move in PRD 请求列表详情

如图3.5所示，此区域主要是提交者所在项目组信息及这个脚本的编写者和完成时间的一些信息，包括请求move in的项目组名称，项目组的manager，完成者，签名时间等。

## database request

这一区域主要填写的是关于请求move in脚本的具体信息，包括在哪个服务器，哪个数据库，脚本位置，具体sql文件，脚本对象详情等。我们检查的主要也是这一区域。

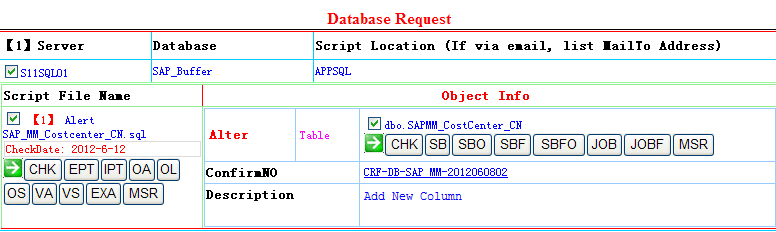


图3.6 Database request列表详情

### 3.3.1 server 的填写

**<Server>:**实例名称（以最终MOVE 到产品环境的实例名为准），或者与DBA约定好的服务器组名或者别名。切忌不要在实例名前面加机器名。

我们目前所使用的一些服务器组及对应组内服务器名和服务器别名及对应服务器名的一些信息如下所附文档。



如果相同的脚本需要提交到多个实例或者整个组的实例上，这样可以在form中只附一个脚本，在Server一栏填多个实例名或者一组实例的组名称。

以下图3.7是一个需要将脚本提交到WareHouse组内所有实例时，Form的填写方法。

这里的warehouse代表这个组内的所有的实例

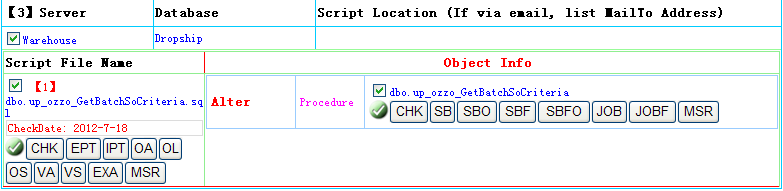


图3.7 Move in一组服务器

如果Server是WareHouse组内除S7SQL01以外的所有Server，填法如图3.8，使用逗号将服务名隔开，使用“-S7SQL01”来排除这个Server。

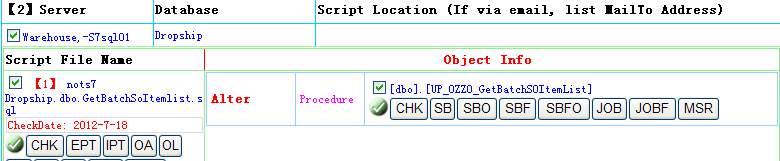


图3.8 Move in除某个实例以外的组内其他实例

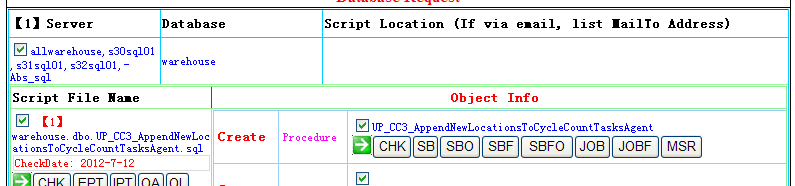
如果是多个SERVER，它们不是包含在一个组内，必须都写出来，我们的填写方法如图3.9所示，将所有的实例名用逗号隔开。

图3.9 move in非同一组的多个实例

**<DataBase>:**为脚本中所对应的数据库名；

**<Sript Location>:**为脚本的一些注释，如果出现产品和GQC的实例名不一致的情况，请在此注明GQC下面所对应的实例名及数据库。如图3.10所示。

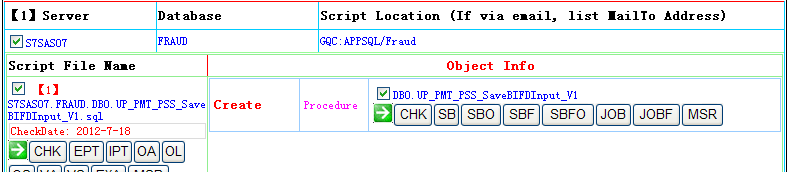


图3.10 move in的产品环境与QC环境实例不对应

**<Sript File Name>:**为附加的脚本；

**<Object Info>:**为左边对应的脚本中的对应操作类型及其对应的对象名。以上图都是Form在DBMOVE TOOL中呈现的样子，以下图3.11是Form本身打开填写时的样子。

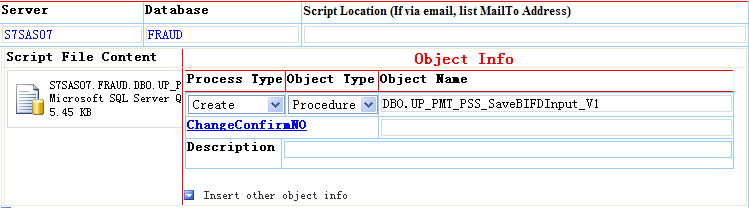


图3.11 插入脚本时，对脚本详情的填写

这里对Object Info作介绍:

**<Process Type>:**为脚本的操作类型；

**<Object Type>:**为脚本中操作的对象类型；

**<Object Name>:**为脚本中操作对象的名称，可以包括架构者和对象名称,如上图。但是要注意，切忌填写其他内容，特别是脚本文件名。

### 3.3.2 alter table/view

如果是Alter View/Table按我们的流程需要走DBCC流程，Form中就需要填ConfirmNO，Form填写如下图3.14所示。

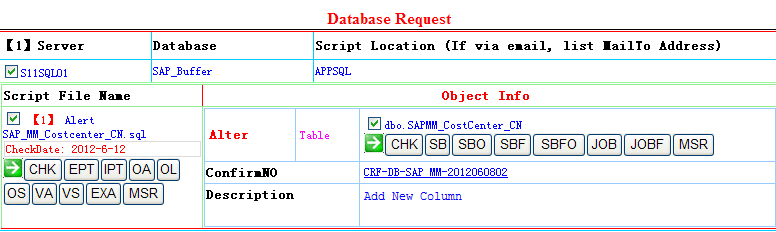


图3.14 Alter Table/View时Form的填写方法

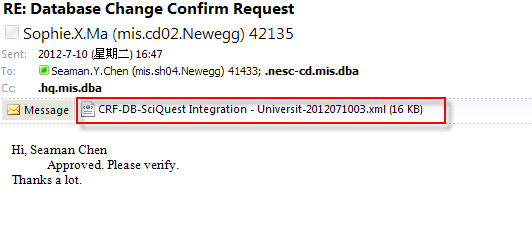


图3.15 DBA返给提交DBCC者的DBCC

**<ConfirmNO>:**填写DBCC确认编号，如上图3.15 是DBA通过DBCC确认最后返回的确认DBCC的Form，这个Form名字的构成CRF-DB-PrjName-YYYYMMDDnn.xml，开发人员在填写ConfirmNO时直接将这个名称去掉.xml后缀然后填进去保存就好了，切记要去掉.xml后缀，这样DBA在用DBMOVE Tools打开该表时，ConfirmNO就会显示成链接的形式，DBA点击一下就可以链接到相应的DBCC form,这样可以确认该DBCCform是否有效，是否与上面的脚本对应（有些开发人员会认为我们看到脚本已经填写了ConfirmNO就会直接通过，直接去看脚本了，所以他们有的人填写的ConfirmNO根本不是针对上面脚本的），最终决定这个操作是否可以运行。

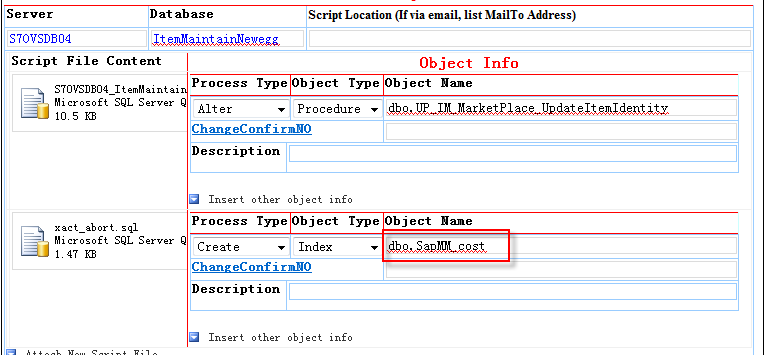
**<Description>：**脚本的一些备注信息。例如如果一些脚本不符合我们DBA的规范是因为他们特殊的业务需求，或者其他特殊情况需要说明，这时就可以让开发人员在该脚本下加description，这样我们DBA看到他们的描述就直接了解他们脚本的一些情况了，就免得再去找开发人员询问和确认。

### 3.3.3 create index

如果操作类型为create index，这个需要注意的是：

第一点：不需要走dbcc流程

第二点：Form中object name必须填写为schema.tablename，例如：dbo.tablename，如下图



### 3.3.4一个脚本对应多个对象情况

如果是一个脚本里面操作了几个对象，如一个脚本里面初始化几个表中的数据，FORM填写可以如图3.12。

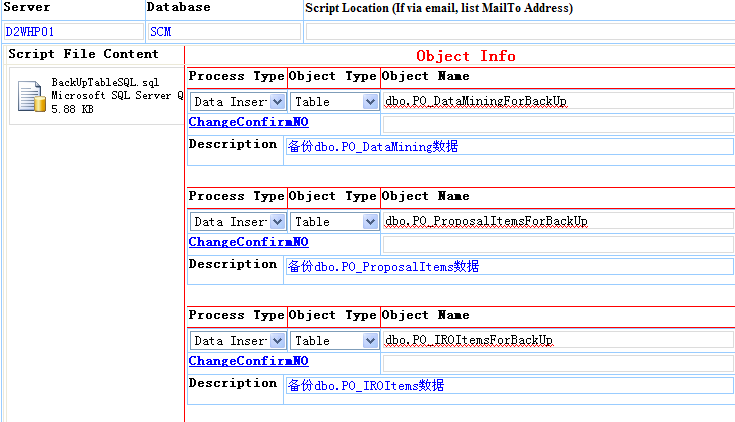


图3.12 一个脚本中操作多个对象

### 3.3.5 create job

如果是创建procedure的同时也需要创建一个JOB，我们的FORM填写如图3.13所示。

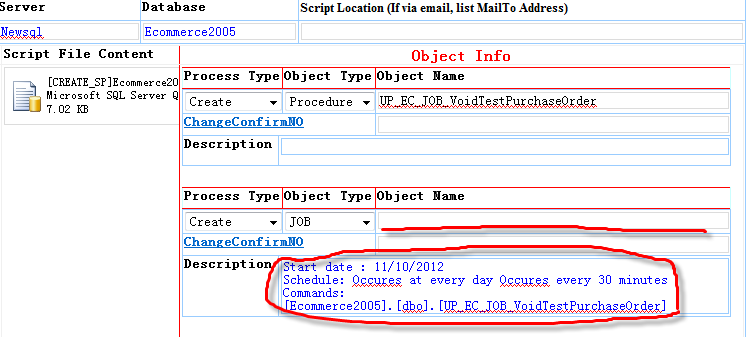


图3.13 create job时Form的填写方法

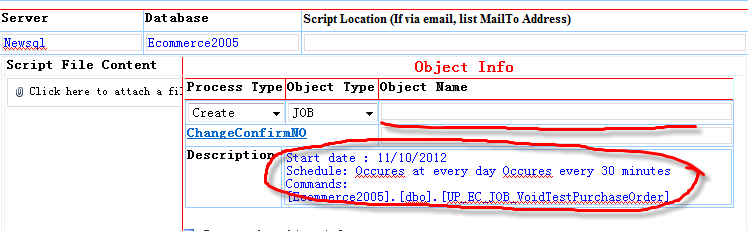
通常我们在建job的时候，都会创建一个Procedure，以供其调用；所以首先是准备好

sp的脚本，选择Create Procedure，填写sp的名称。然后点击下面的Insert other object info添加一行，选择Create job 并填写Job name(jobname的命名为JOB\_DBName\_ProcedureName),注意这里不再需要添加任何脚本；在description中填写两个重要的信息Job command和调度job的Schedule。

注：对于job的名称，如无特别要求，请勿指定，一般job的名称是DBA在创建时由DBA来命名，DBA创建了该job之后会将该job的名称返给之前提交Form者。

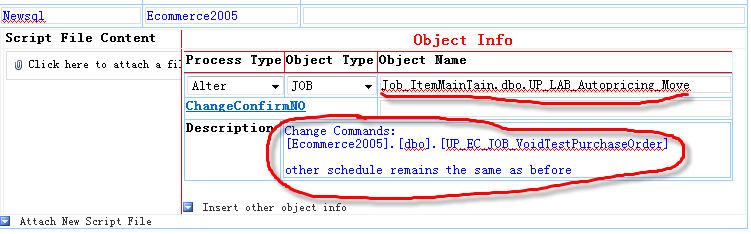
对于Schedule，如果不是周期性Job（每多少分钟（小时）执行一次的），请给定一个大致的时间范围，以便DBA根据服务器的Job情况设置具体的时间。

当创建job中调用的procedure已经存在，现在只是简单的想要创建一个job，那Form的填写如下图：



### **3.3.6 alter job**

如果想要修改一个线上存在的job，Form填写如下图，job的名称需要填写出来，除此之外一定要说清楚，你现在需要修改的什么，其他部分是否需要修改



## replication request

如果有创建同步链的需求，在这里可以填写请求同步链的详情，如同步链类型，同步的源头服务器，所需同步的表，目标服务器，执行时间和频率。如图3.21所示。

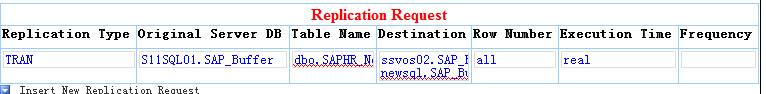


图3.21replication request 列表详情

**<Replication Type>：**同步的类型，常用的有TRA事务复制，SNP快照复制。目前在产品环境只允许建立事务型复制；

**<Original Server DB>:**源头实例名.数据库名；

**<Table Name>:**源头的表名，填写如[Schema].[TableName]；

**<Destination>:**目的地的实例名.数据库.表名（表名如果和源头一样可以省略），如果destination是多个server，如上图，那我们可以在这一列填写多个servername.dbname，以换行的形式隔开

**<Row Number>:**过滤条件，如果是全部同步，即填写ALL；

**<Execution Time>:**同步的时间，常用在快照型的同步，如果不填默认为实时同步；

**<Frequency>:**同步的频率。

## Others

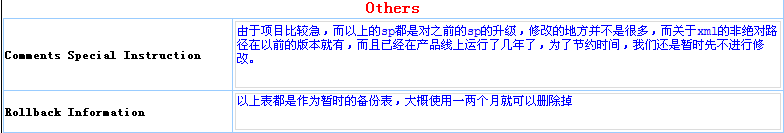
Others包括特殊说明信息和反馈信息。对于每个脚本，可以在ObjectInfo的对象名称后面的Description中填写相关的注意事项，而对于整个Form，如果需要让DBA在检查或者执行当中需要特别注意的地方，可以在Others里面特别说明一下情况，如图3.24所示。

图3.24 Others列表详情

## 其他注意事项

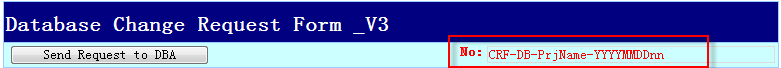
* 1. create/alter table和insert/update date table脚本不能放在同一个Form中
  2. 有特殊需求的需要在脚本中和Form中添加备注说明。Form中针对一个脚本的可以添加description中，针对整个Form可以添加others一栏
  3. 当你的脚本存在问题，DBA回复邮件给你，请你在DBA返回给你 的Form上做修改
  4. 创建log表的Form中应该同时出现定期删除该log表的job
  5. 关于insert/update/delete 一次性脚本，都必须加top限制，如果数据量多的话，就需要按照后面章节中讲到的批处理规范进行循环处理

# DBCC的填写

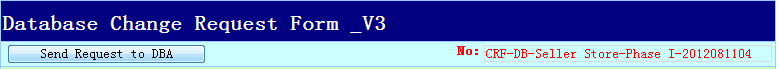
DBCC 主要有五个部分

## DBCC No

没有被DBA审批过的DBCC，NO一栏如下图：

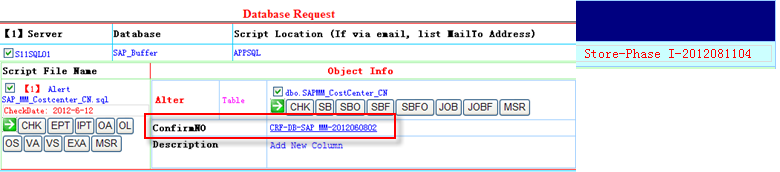


被DBA 审批过后的DBCC，No一栏如下图：



DBCC No的组成：CRF-DB-ProjectName-SignDate，这个是在DBA 给你们签名过后，由系统自动生成的。

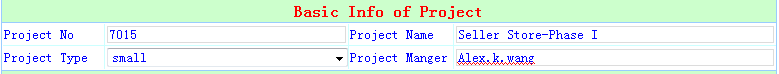
以上DBCC No就是我们DBForm中需要的ChangeconfirmNo，如下图：



在填写这个ChangeconfirmNo的时候，我们尽量采取粘贴复制的方式，否则即使后面多了一个空格，那我们的工具也是会检测出这个DBCC No都是无效的。

## basic info of project

这一部分就主要是填写项目及项目组的信息，这个就是开发人员根据自己的实际情况来填写。



## detailed info of Change

这一部分是DBA检查的重点，这里需要特殊说明的是：

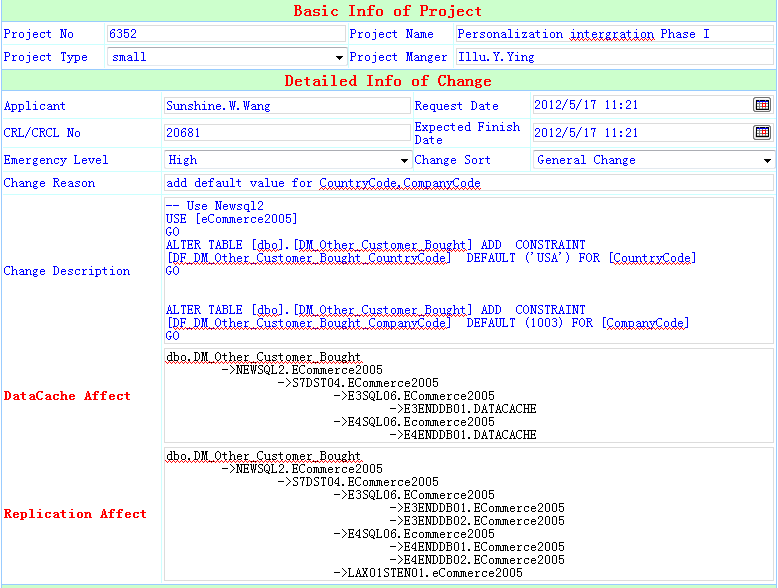
Change Reason：填写你们做出修改的原因

Change Description：填写你们修改的具体代码，这里注意的是：一定要记得将你操作所在server，db写清楚。格式如下图所示

DataCache Affect：查询该表是否有DataCache，如果有，请填写，格式如下图所示。如果没有就填写N/A

Replication Affect：查询该表是否存在replication，如果有，请填写，格式如下图所示。如果没有就填写N/A

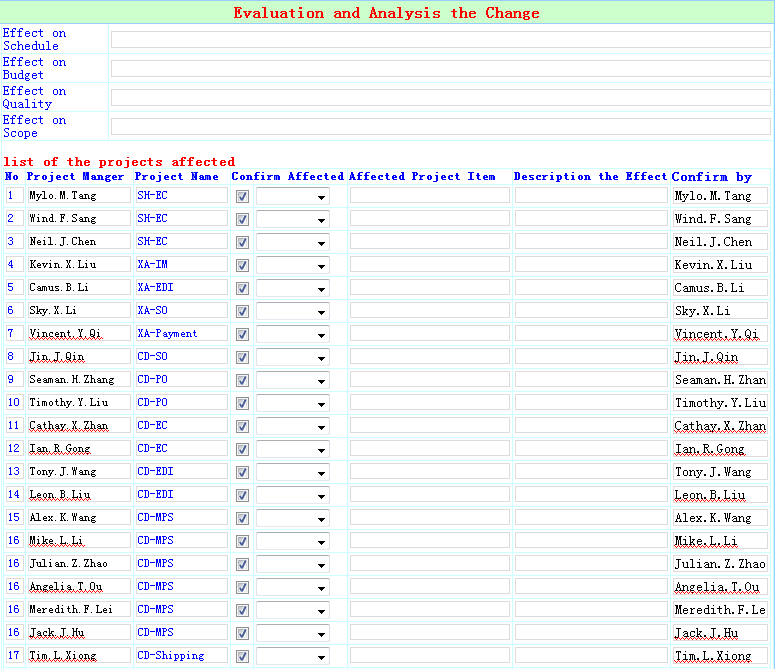
一个表的DataCache /Replication获取方式，请看下面DBMove Add in的介绍部分。



## Evaluation and Analysis the Change

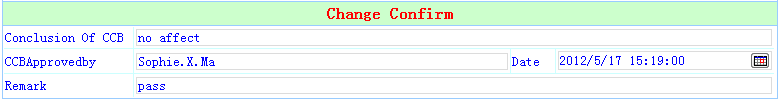
这一部分主要是考虑到组之间业务关联的情况，如果你们修改的表有其他组也在用，那你们在修改时应该询问一下相应的组该修改是否可以进行。这一栏就是在list of the projects affected中勾选上你们修改所影响到的组。这里注意的是，新的DBCC默认是将所有的组都勾选上的，每次开发人员在提交时都需要把没有影响到的组的勾去掉。

当有影响到其他组，开发人员将该DBCC提交到DBA,DBA会将该邮件转发到影响组的leader，让他们确认，如果他们都觉得没有问题，DBA 才能签名，然后将该签了名的DBCC返回给开发人员。



## change Confirm

这一部分是DBA在确认该DBCC中的请求是合理的，且都已经通过被影响组的leader的同意后，然后进行签名。开发人员拿到DBCC后，可以确认下DBA是否签名，签名了的DBCC代表是被审核通过的，这是DBCC No才算是有效的。



## 其他注意事项

* 1. Change Descripiton中应该包含你操作所在server，db信息
  2. 给某个表添加带有默认值约束的非空列步骤：

添加带有默认值的可空列

循环更新该列为默认值

修改该列为非可空列

Eg：

|  |
| --- |
| USE db  GO  --step1  ALTER TABLE dbo.tb\_item ADD state INT NULL  CONSTRAINT DF\_tb\_item\_state DEFAULT 1  --Step2  WHILE (1=1)  BEGIN  UPDATE TOP(1000) dbo.tb\_item ------批处理规范  SET state=1  WHERE state IS NULL;    IF @@ROWCOUNT<>1000  BREAK;  WAITFOR DELAY '00:00:10'  END  GO  --Step3  ALTER TABLE dbo.tb\_item ALTER COLUMN state INT NOT NULL |

* 1. 批量处理的规范

关于Production环境数据初始化、数据迁移以及批量处理数据，请按照以下规则：

* 1. 对于同步链上的表，一般要求每次更新 <=1000 条，Delay >=15 秒；
  2. 对于非同步链上的表，一般一次更新 <=2000，延迟 >= 5 秒。

# DBMove ADD in的使用

这里我只简单介绍我们经常使用的几个功能。

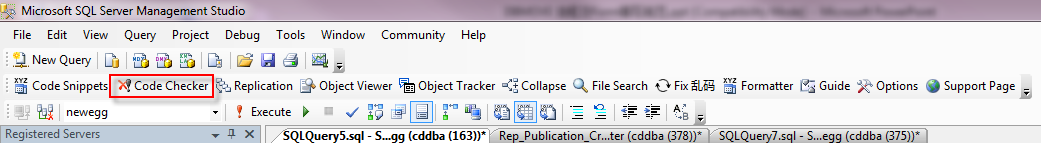
## Code Checker

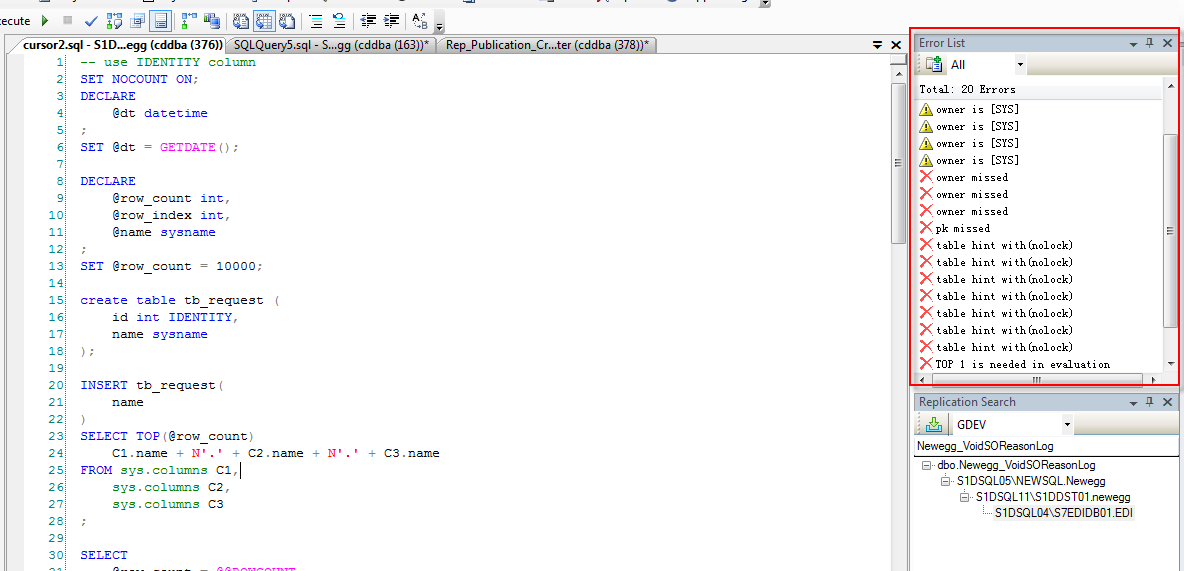
Code Checker：对我们的脚本进行初步的检查。

**使用方法：**

打开查询窗口，然后点击该按钮，这时就会弹出一个如下图Errorlist窗口，里面罗列了该脚本中存在一些问题，你可以将鼠标放上去，它就会弹出浮动信息，对该条信息进行详细说明，也可以双击某条信息，这样该条信息的详细说明会出现在查询窗口中。一般前面图标是红色叉的是比较严重的错误，应该都需要进行修改，感叹号图标是一些警告，这部分也是需要尽量做修改的。

一般开发人员在将脚本提交给DBA之前都最好先这样做一个初步的检查，免得后面因为这些简单的问题跟DBA邮件来来往往，浪费双方的时间





## Replication

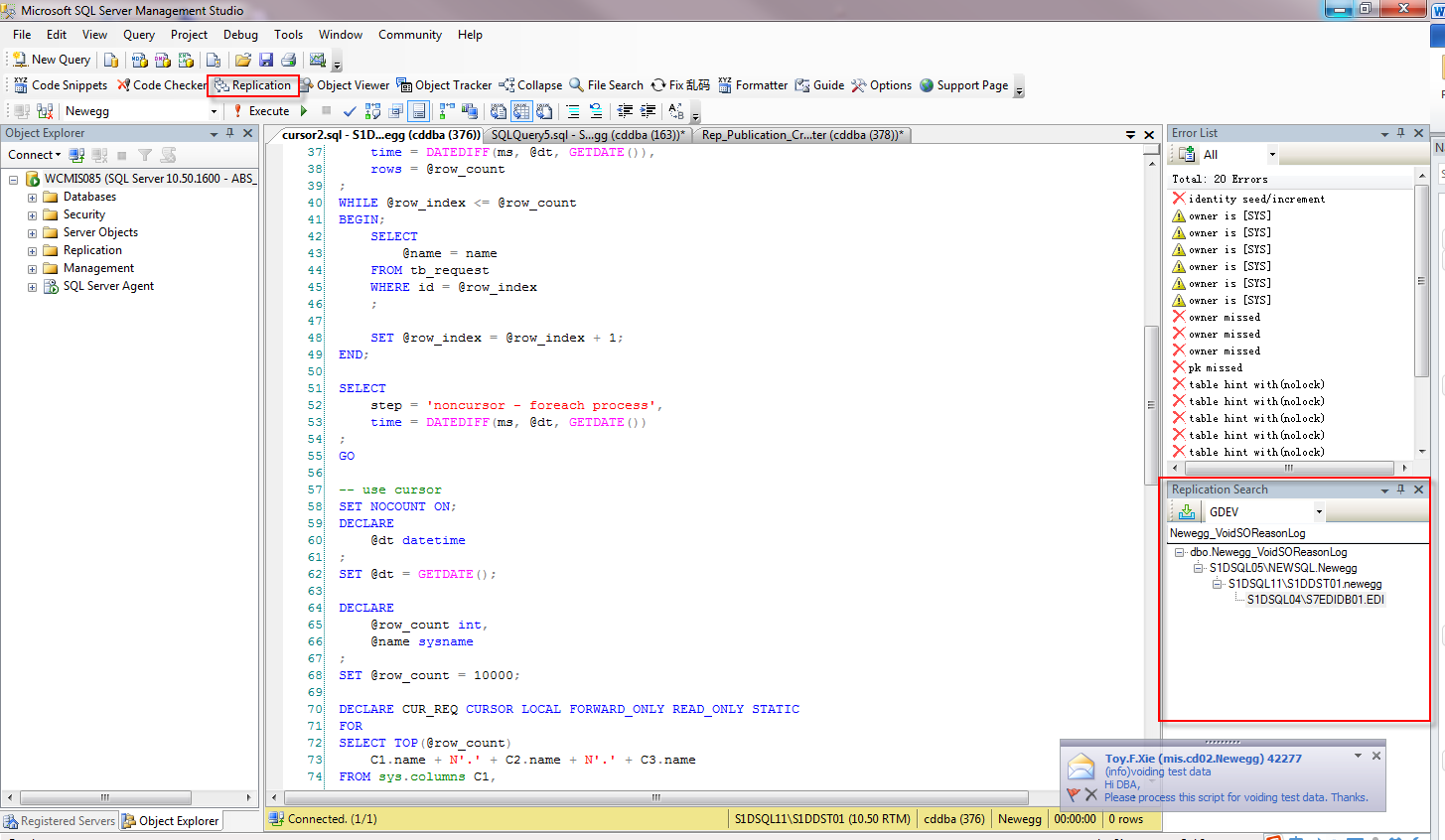
这个工具是供开发人员和DBA查询表的同步练关系，也可以确认该表下面有没有DataCache。

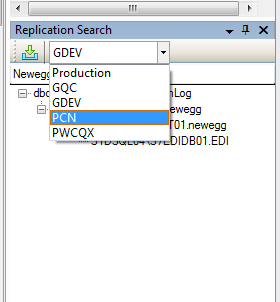
**使用方法：**

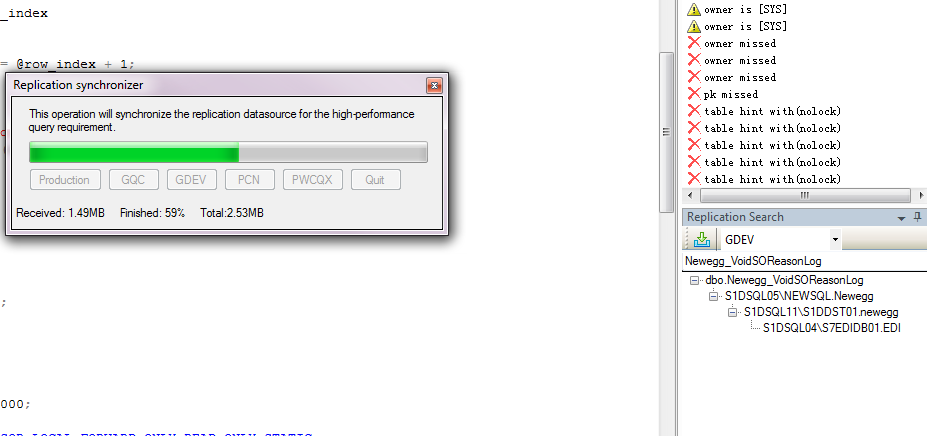
点击如图中的replication按钮，然后就会出现Replication Search的窗口。我们可以如下图所示，只用将我们的表名称填写在空白处，然后按回车键，同步练关系就出来了。第一行是表名称，第二行是同步练的源端服务器和db名称，后面的就是中转服务器和目的端，对于开发人员来讲可以将后面的都看成是目的端，表的同步关系如图中的层次关系一样，一层一层的同步下来。

**注意点：**

1. 图中显示GDEV字样的地方是可以进行选择的，这个是我们公司相应的几个数据库环境，开发人员可以根据自己的需求，选择相应的环境。
2. 因为我们各个环境中的表的同步练关系都是变化的，我们这个插件是一天会去更新一次，所以在你需要查询一个表的同步练关系前都需要更新一下，这样才能获取到最新的信息。绿色下载剪头按钮就是我们的更新按钮，点击该按钮，就会弹出如下图Replication synchronize窗口，最后我们也是选择自己所要查询的环境就可以了。







## object viewer

object viewer可以供我们获取到各个环境下对象的脚本

**使用方法:**

点击object viewer，会出现production object Source code Viewer窗口，我们根据上面的提示，在空白处填写相应的server，db，owner，objectname，然后点击start就可以获取到该对象的脚本，该脚本会显示在一个新的查询窗口中。

**注意点：**

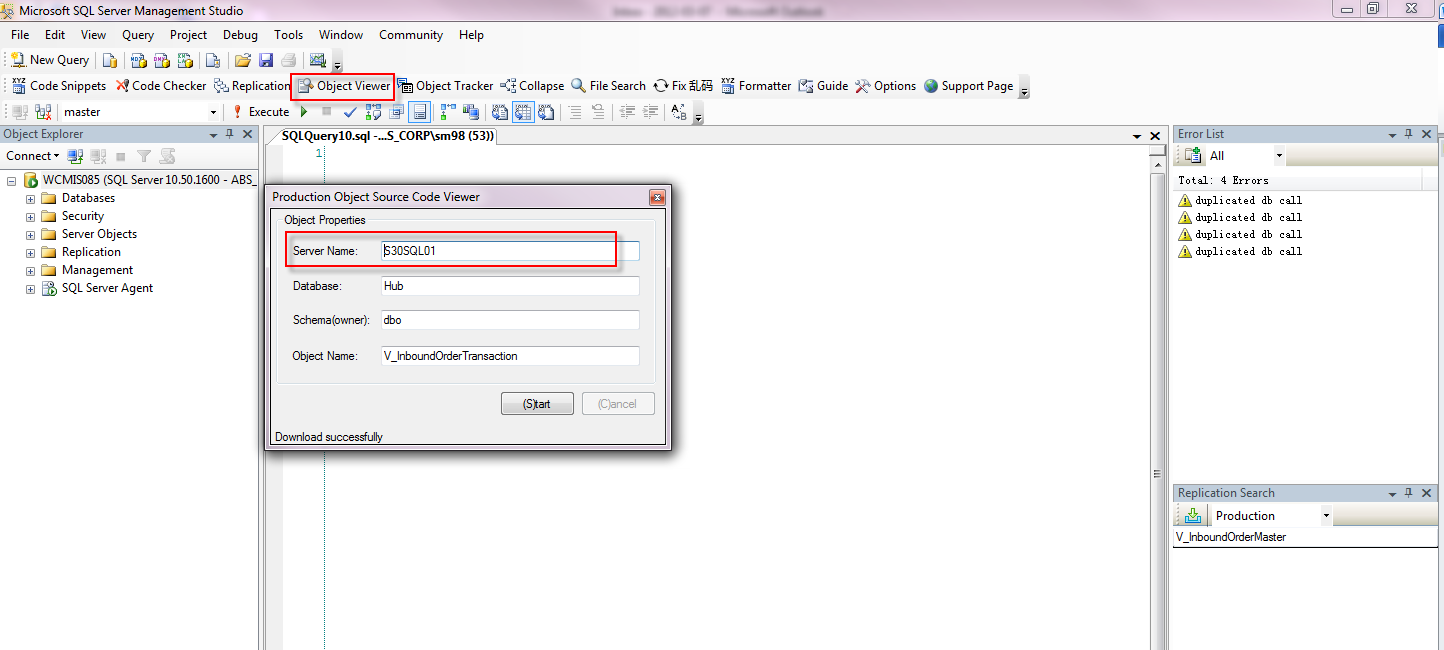
这里我们是根据servername来区分你获取的对象是QC，dev还是prd环境中的。

QC的servername组成：hostname\instance，hostname以S1Q开头，比如S1QSQL06\NEWSQL。

DEV的servername组成：hostname\instance，hostname 以S1D开头，比如：S1DSQL05\NEWSQL

PRD的servername组成：hostname。比如：NEWSQL

对于以上servername组成有疑问的，可以学习下第一章数据库环境说明部分。



## Formatter

用来规范我们的脚本。

**用法：**

选中你需要规范的脚本，然后点击该按钮，你的脚本就会被初步的规范。

**注意点：**

用这个规范完，有些地方还是会不尽人意，还有一些地方也许会弄错，比如本来是一行注释行，但是这个工具无法识别，有可能会换行，这样注释部分有一部分就会变成正文，所以在使用这个工具规范完之后必须再人为的检查下你的脚本，该纠正的地方进行修正。

## Guide

供开发人员拿一些的新的Form表单，DBCC，SSB表单和同步数据的表单。

**用法：**

点击Guide按钮，就出现如下图，然后你再根据上面的信息提示点击你想要的表单，这样就很容易的获取到了新的表单

