Service Broker

Contents

[1 Purpose 1](#_Toc36727044)

[2 Service Broker 2](#_Toc36727045)

[2.1 Document reference 2](#_Toc36727046)

[2.2 Services 2](#_Toc36727047)

[2.3 Queues 2](#_Toc36727048)

[2.4 Contract 2](#_Toc36727049)

[2.5 Service Broker Framework Setup 2](#_Toc36727050)

[2.5.1 Script to create Services and Queues 2](#_Toc36727051)

[2.5.2 Application objects 4](#_Toc36727052)

[3 Using the Service Broker 5](#_Toc36727053)

[3.1 Pending messages processing 5](#_Toc36727054)

[4 Maintenance 5](#_Toc36727055)

[4.1 Background Sessions 5](#_Toc36727056)

[4.2 Verify the Logs 6](#_Toc36727057)

[4.2.1 Service Broker Log 6](#_Toc36727058)

[4.2.2 Trigger Log 6](#_Toc36727059)

[4.2.3 Successful process 6](#_Toc36727060)

[4.2.4 Failure Process 7](#_Toc36727061)

# Purpose

application requires some of the database scripts, which can be executed in the background asynchronously.

The scripts which can be executed asynchronously do not have any dependencies on its completion time.

**Examples**: Deleting the data updating the data etc

Service Broker can trigger the scripts request in background asynchronously.

# Service Broker

## Document reference

<https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-service-broker?redirectedfrom=MSDN&view=sql-server-ver15>

## Services

* //SB\_AT/ProcessReceivingService
* //SB\_AT/ProcessStartingService

## Queues

* SB\_AT\_Request\_Queue
* SB\_AT\_Response\_Queue

## Contract

1. //SB\_AT/Contract

## Service Broker Framework Setup

This is one time setup on the required database. This framework is not limited to .

It can be utilized for any application.

### Script to create Services and Queues

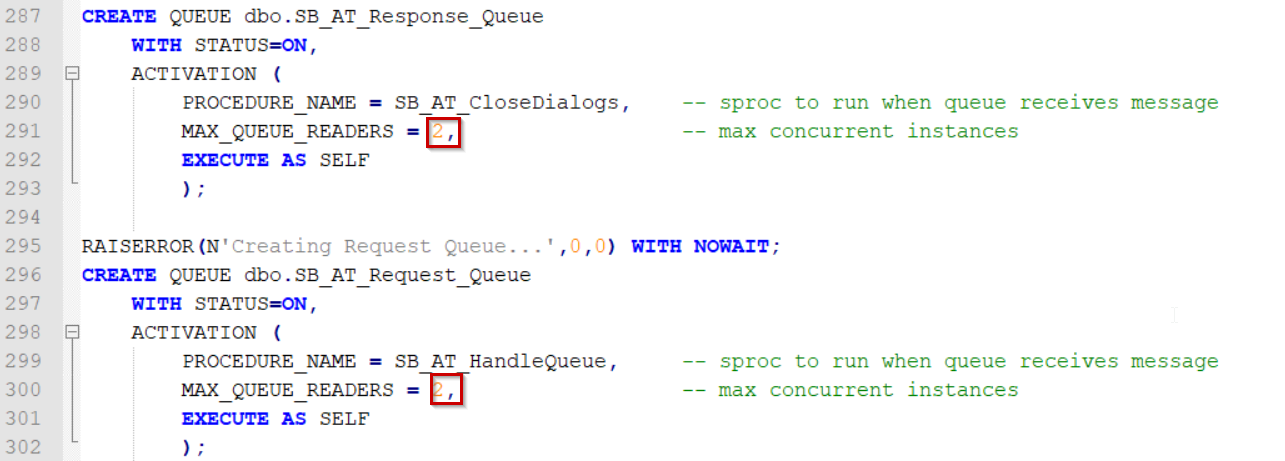


Script is prepare to Handle two request simultaneously.

If we need to change the simultaneous tasks, we need to make the below changes to the script.

Change the number ‘2’ to desired size.

This request will take background sessions. Make sure you have enough amount service capacity to handle multiple concurrent request.



This script shall create the below objects in the database.

**Stored Procedures**

SB\_AT\_Fire\_Trigger

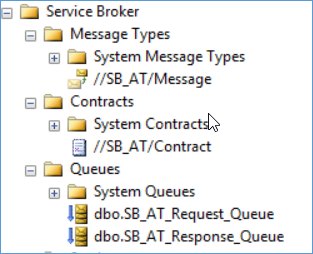
SB\_AT\_CloseDialogs

SB\_AT\_HandleQueue

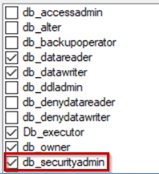
**Tables**

SB\_AT\_ServiceBrokerLogs

After execution of the above script, we can notice the below



**Note:**  Role required to view the above details is “db\_securityadmin” for the SQL Account.



### Application objects

1. To utilize the Services and Queues created in Service broker, application is having below objects

**Stored Procedures**

* + Trigger\_Create\_Job
  + Trigger\_Execute\_Job

**Tables**

* + Trigger\_Jobs

Table Schema details

**Job\_Request\_From**: Which Service / SP Requested this Job

**HostName:** Requested HostName

**Exec\_Sp\_Name:** Sp to be executed. It should be unique name for that kind of requests. Request

can be to execute a SP or a script to be executed.

**Job\_Query:** Actual query to be exucted.

**Job\_Created\_Time:**

When the Job created. This shall be created by the system, once the request is inserted.

**Start\_Time:**

Once the Job execution started by the Service broker, this Time shall be recorded

**End\_Time:**

Once the Job execution completed by the Service broker, this Time shall be recorded

**Status:** Status of the Job.

Pending, Started, Complete, Failed, Hold, Reject

***Pending****: Initial status of the request*

***Started****: Once the Job execution started. “Start\_Time” shall be updated.*

***Complete****: After completion of the Job. “End\_Time” shall be updated*

***Failed****: If the Job failed.*

***Hold****: If we need to hold the job not to run, until we enable to “pending”.*

*There is no UI to change this, we need to use SQL to change the status.*

***Reject****: If we do not want this Job to be executed any more.*

*There is no UI to change this, we need to use SQL to change the status.*

**Triggers**

* + Exec\_Trigger\_Jobs

  (on the Table “Trigger\_Jobs”**)**

# Using the Service Broker

## Steps to create in new application

1. Run the script [2.5.1] on the database where you want the asynchronous trigger(s).
2. Create a stored procedure which will receive the following two parameters:

@inserted XML,

@deleted XML

This procedure will be responsible for parsing the inserted/deleted data and executing the

actual trigger logic based on that data.

3. Inside the actual table trigger, use the following code:

DECLARE

@inserted XML,

@deleted XML;

SELECT @inserted =

( SELECT \* FROM inserted FOR XML PATH('row'), ROOT('inserted') );

SELECT @deleted =

( SELECT \* FROM deleted FOR XML PATH('row'), ROOT('deleted') );

EXECUTE SB\_AT\_Fire\_Trigger '{YourProcedureName}', @inserted, @deleted;

But replace {YourProcedureName} with the name of the procedure you've

created in step 2.

## Usage in the application

1. To utilize this framework we need to insert the required execution scripts as below

SET @P\_Job\_Query = 'EXEC [dbo].[DeleteFileData]’

EXEC [dbo].[Trigger\_Create\_Job] 'Madhu Test', DeleteFileData, @P\_Job\_Query

We need to pass the three parameters to the “Trigger\_Create\_Job”

It’s input parameters are as below

 @P\_Job\_Request\_From NVARCHAR(100)   - Which Service / SP sending this request, it can be by a person also

 @P\_Exec\_Sp\_Name NVARCHAR(100)  - This is the SP name to be executed, this is a non-unique Key field

 @P\_Job\_Query NVARCHAR(MAX)     - Query which need to be executed.

It will create the records in the table ‘Trigger\_Jobs’.

A Trigger will send a message to service broker to execute the SP ‘Trigger\_Execute\_Job’

## Pending messages processing

Once the request is received to the Queue, it will trigger the request to execute the Job.

If there are any jobs with the same “**Exec\_Sp\_Name”** type in the ‘Trigger\_Jobs’ table and the status is “Pending” then system shall process all the pending Jobs with that type.

This is step is taken to handle any failure cases in their previous execution.

# Maintenance

## Background Sessions

To check the number of sessions running in the background, execute the below query.

If there is no background sessions for service means, the service broker went to disabled state due to multiple errors.

SELECT sqltext.TEXT,

req.session\_id,

req.status,

req.command,

req.cpu\_time,

req.total\_elapsed\_time

FROM sys.dm\_exec\_requests req

CROSS APPLY sys.dm\_exec\_sql\_text(sql\_handle) AS sqltext

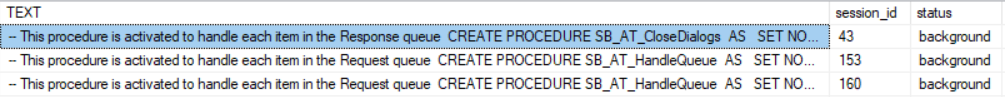
**Status**: background.

Status of this tasks always “background”

**Session ID:** This number varies, number is irrelevant. Other queries will not use these

session\_id.

**Text:** It provides the details of Stored Procedures using by Services of Service Broker.



## Verify the Logs

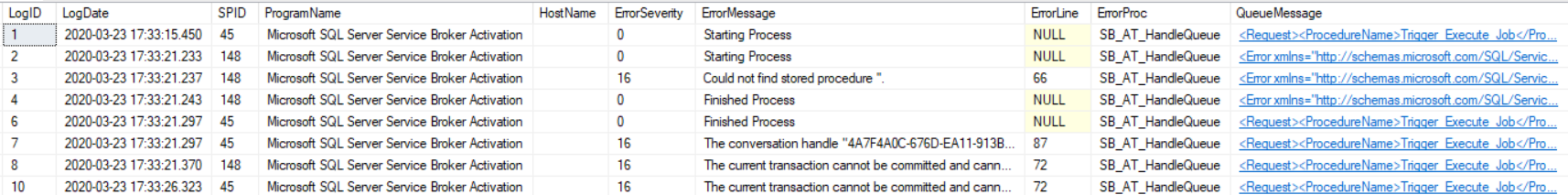
### Service Broker Log

Requests sent to the queue shall be retrieved from the below table

**Table Name**: [dbo].[SB\_AT\_ServiceBrokerLogs]

**Make sure to use query as below**

SELECT \* FROM [dbo].[SB\_AT\_ServiceBrokerLogs] WITH (NOLOCK)



### Trigger Log

Job Request submitted contains in the table: [Trigger\_Jobs]

Use the query is as below

SELECT \* FROM [dbo].[Trigger\_Jobs] WITH (NOLOCK)

### Successful process

If the message is successful, we can see the below details

**ErrorMessage**: Stating Process, Finished Process, means Jobs is successful.



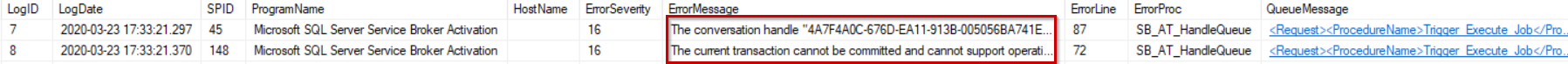
To check the message info click on the XML in “QueueMessage”

Message looks like below.

### Failure Process

SELECT \* FROM [dbo].[SB\_AT\_ServiceBrokerLogs] WITH (NOLOCK)

If the message is failed due to any unhandled exception while executing the Job.

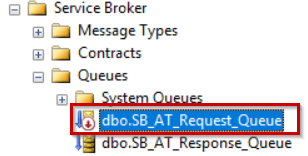


Job requests in the “Trigger\_Jobs” will be in pending state

SELECT \* FROM [dbo].[Trigger\_Jobs] WITH (NOLOCK)

It will provide the list with “Pending” state.

If there are multiple failures, then the queue shall go to disabled state.



It will remain in disable state, until we manually enable it by right click on that and enable it.

