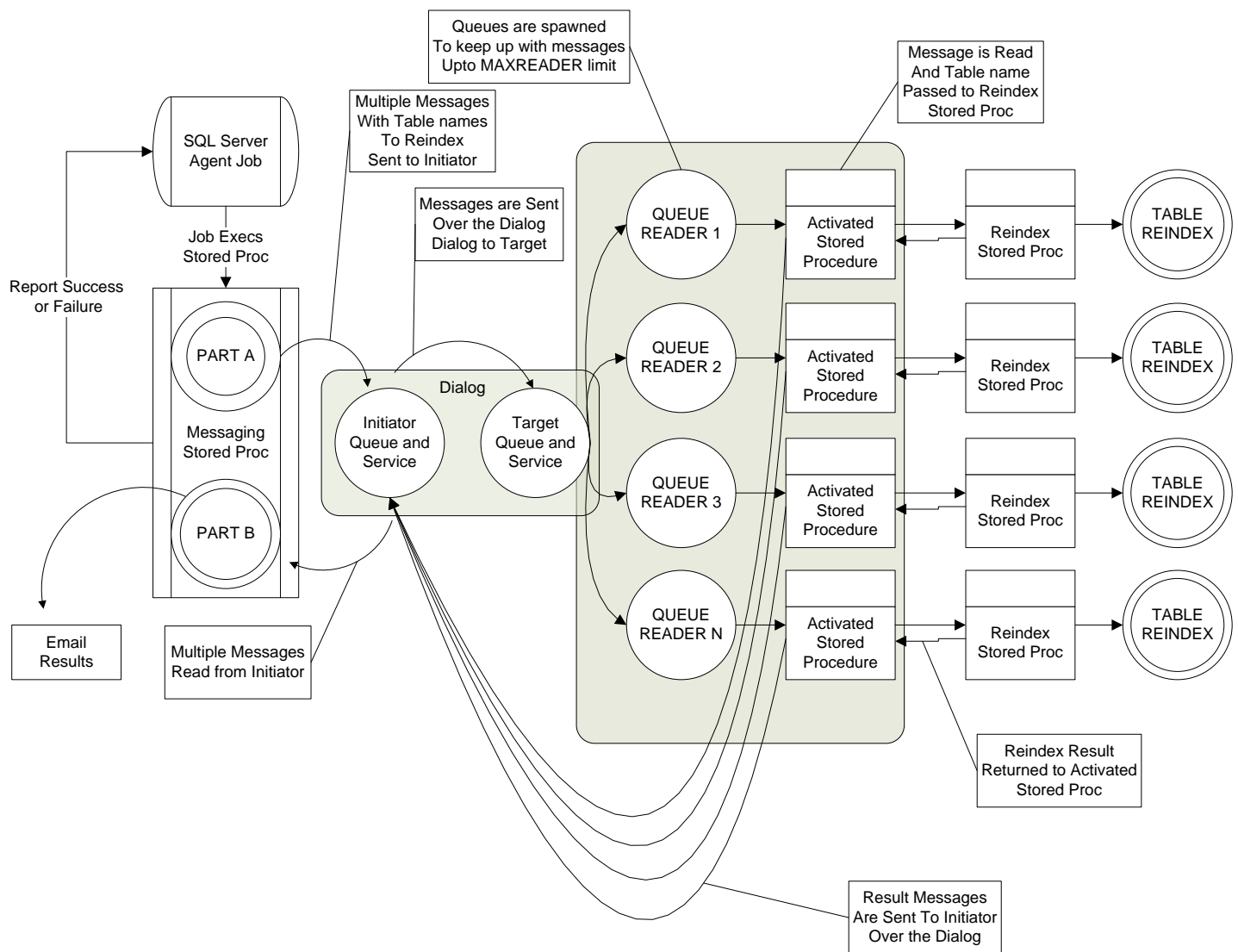


Using Service Broker to Enable Parallel Task Execution

Using a service broker queue with an activated stored procedure allows the service broker to instantiate multiple copies of the stored procedure as needed in response to the number of messages coming in.

In this case a simple system is designed so that multiple tables can be reindexed in a parallel fashion due to a service broker queue which has internal activation enabled and allows a maximum of N queue readers each of which can activate the stored procedure which will reindex a table.



The flow progresses as follows

1. A SQL server job is started which executes a stored proc (usp_servicebrokerdefrag)
2. The messaging stored proc has two parts of which PART A is described in this step.
 - a. PART A determines which tables are fragmented and then sends the message
 - i. The `sys.dm_db_index_physical_stats` DMV is queried for tables\indexes over X% fragmented
 - ii. The overall table names are put into a cursor

Copyright 2010 Enterim LLC by Paul M Drumm

- iii. Each loop through the cursor starts a dialog between the initiator service and the target service over the contract which defines the services involved and the message types.
 - iv. The message with the table name is sent in a conversation over that dialog
 - v. It then goes to part B
3. The message is received by the target service and put into its queue.
 4. The queue reader picks up the message. If many messages are waiting multiple queue readers will be spawned up to the MAXREADER value by the queue monitor (not pictured) each of which runs on a separate thread and hence the ability to do parallel task execution.
 5. The queues are internally activated which means they have a stored proc associate with them. The stored proc in this case reads the message body.
 6. The message body contains the table name to be reindexed and the activated stored proc calls the reindexing stored proc using the table name.
 7. Multiple tables are subsequently reindexed at the same time.
 8. Each activated stored proc waits for the reindexing stored proc to return the result back to it.
 9. Once the result is returned it is put into a message and passed back to the initiator queue over the same conversation handle.
 10. The messaging stored proc PART B has been waiting in receive mode for messages coming into the initiator queue.
 11. Each message is examined for success or failure of the specific table it was set to reindex.
 12. The messages results are coalesced in preparation for emailing.
 13. If any of the table that were reindex returned a failure code then the email is marked as error with all the results and an error code is returned to the SQL server agent job that initiated it which also results in notification to the DBA.
 14. If the message results are all success then an email stating success with all the results is sent to the DBA and a clean return code is sent to the SQL server agent job.

Object list – all objects in MSDB database. Service Broker is enabled there if you have set up database mail.

Messaging Stored Procedure – `dbo.usp_servicebrokerdefrag`

Activated Stored Procedure – `dbo.TargetActivProc`

Reindex Stored Procedure – `dbo.usp_reindex_table`

Request Message - `//MSDB/InternalAct/RequestMessage`

Reply Message - `//MSDB/InternalAct/ReplyMessage`

Initiator Queue - `dbo.InitiatorQueueIntAct`

Target Queue - `dbo.TargetQueueIntAct`

Initiator Service - `//MSDB/InternalAct/InitiatorService`

Target Service - `//MSDB/InternalAct/TargetService`

Contract - `//MSDB/InternalAct/SimpleContract`

SQL Service Agent Job – `WeeklyReindexJob`

Note: If you would like a copy of the code to do this then please email me at Paul@Enterim.com or Paul@CallPaul.com.