KDS Service Broker & WebSocket Integration — Full Documentation

1. Overview

Real-time notification system: trigger on dbo.tbl_TempKot sends messages to a Service Broker queue. A Python WebSocket server reads the queue and broadcasts JSON to browser clients.

2. Components

Message Type, Contract, Queue, Service, Trigger, Python listener (WAITFOR/RECEIVE), WebSocket server, Browser JS.

3. Workflow

(tbl_TempKot change) → Trigger → BEGIN DIALOG / SEND → KDS_TriggerQueue → Python sql_listener → broadcast_tickets() → WebSocket → Browser UI

4. SQL: Full scripts (cleanup, create, trigger, tests, debug)

Run blocks in SSMS connected to database [synopos-cp]. Execute each GO-separated batch.

A. Safe cleanup (drop in correct order)

```
USE [synopos-cp];
GO
DECLARE @ch UNIQUEIDENTIFIER;
WHILE EXISTS (SELECT 1 FROM sys.conversation_endpoints WHERE state IN ('CO','CD','DI','ER'))
BEGIN
    SELECT TOP 1 @ch = conversation_handle
    FROM sys.conversation_endpoints
    WHERE state IN ('CO','CD','DI','ER');
    END CONVERSATION @ch WITH CLEANUP;
END
GO
IF OBJECT_ID('trg_TempKot_Change', 'TR') IS NOT NULL DROP TRIGGER trg_TempKot_Change;
GO
IF EXISTS (SELECT * FROM sys.services WHERE name = 'KDS_TriggerService') DROP SERVICE [KDS_TriggerService];
GO
IF EXISTS (SELECT * FROM sys.service_queues WHERE name = 'KDS_TriggerQueue') DROP QUEUE [KDS_TriggerQueue];
GO
IF EXISTS (SELECT * FROM sys.service_contracts WHERE name = 'KDS_TriggerContract') DROP CONTRACT [KDS_TriggerContract];
GO
IF EXISTS (SELECT * FROM sys.service_message_types WHERE name = 'KDS_TriggerMessage') DROP MESSAGE TYPE [KDS_TriggerMes
GO
```

B. Create Service Broker objects

```
USE [synopos-cp];
GO
IF NOT EXISTS (SELECT * FROM sys.service_message_types WHERE name = 'KDS_TriggerMessage')
CREATE MESSAGE TYPE [KDS_TriggerMessage] VALIDATION = NONE;
GO
IF NOT EXISTS (SELECT * FROM sys.service_contracts WHERE name = 'KDS_TriggerContract')
CREATE CONTRACT [KDS_TriggerContract] ([KDS_TriggerMessage] SENT BY INITIATOR);
GO
IF NOT EXISTS (SELECT * FROM sys.service_queues WHERE name = 'KDS_TriggerQueue')
CREATE QUEUE [KDS_TriggerQueue] WITH STATUS = ON, RETENTION = OFF;
GO
IF NOT EXISTS (SELECT * FROM sys.services WHERE name = 'KDS_TriggerService')
CREATE SERVICE [KDS_TriggerService] ON QUEUE [KDS_TriggerQueue] ([KDS_TriggerContract]);
GO
```

C. Trigger on dbo.tbl_TempKot (send message)

```
USE [synopos-cp];
GO
IF OBJECT_ID('dbo.tbl_TempKot', 'U') IS NOT NULL
```

```
BEGIN
    IF OBJECT_ID('trg_TempKot_Change', 'TR') IS NOT NULL
        DROP TRIGGER trg_TempKot_Change;
    GO
    EXEC('CREATE TRIGGER trg_TempKot_Change
    ON dbo.tbl_TempKot
    AFTER INSERT, UPDATE, DELETE
    AS
    BEGIN
        DECLARE @msg NVARCHAR(MAX) = N''KOT table changed'';
        DECLARE @dialog UNIQUEIDENTIFIER;
        BEGIN DIALOG CONVERSATION @dialog
            FROM SERVICE [KDS_TriggerService]
            TO SERVICE ''KDS_TriggerService''
            ON CONTRACT [KDS_TriggerContract]
            WITH ENCRYPTION = OFF;
        SEND ON CONVERSATION @dialog MESSAGE TYPE [KDS_TriggerMessage] (@msg);
        END CONVERSATION @dialog;
    END');
END
ELSE PRINT 'Table dbo.tbl_TempKot not found. Trigger not created.';
GO
```

D. Test: update row + read queue

```
-- Make a small update (use a real KOT_NO)
UPDATE dbo.tbl_TempKot SET Qty = Qty WHERE KOT_NO = (SELECT TOP 1 KOT_NO FROM dbo.tbl_TempKot);
WAITFOR (
    RECEIVE TOP(1)
        message_type_name,
        CAST(message_body AS NVARCHAR(MAX)) AS message_body
    FROM [KDS_TriggerQueue]
), TIMEOUT 10000;
GO
```

E. Debugging checks

```
SELECT name, is_broker_enabled FROM sys.databases WHERE name = 'synopos-cp';
SELECT * FROM sys.service_queues;
SELECT * FROM sys.services;
SELECT * FROM sys.service_contracts;
SELECT * FROM sys.service_message_types;
SELECT * FROM sys.transmission_queue;
SELECT conversation_handle, state, far_service FROM sys.conversation_endpoints;
```

5. Python server (server.py) — notes + snippet

```
# Notes:
# - run HTTP static server + websocket in asyncio main loop
# - start sql_listener in a background thread passing main loop
# - sql_listener uses WAITFOR(RECEIVE) on KDS_TriggerQueue; on message call loop.call_soon_threadsafe(asyncio.create_ta

import asyncio, json, pyodbc, websockets, threading
CONN_STR = "DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-NKDVK7U;DATABASE=synopos-cp;UID=posgst11;PWD=hello213

async def broadcast_tickets():
    tickets = fetch_tickets()  # your fetch_tickets function
    data = json.dumps(tickets)
    for c in clients.copy():
        try: await c.send(data)
        except: clients.discard(c)

def sql_listener(loop):
    conn = pyodbc.connect(CONN_STR)
    cur = conn.cursor()
    while True:
        cur.execute("""WAITFOR (
            RECEIVE TOP(1)
                conversation_handle,
                message_type_name,
                message_body
            FROM [KDS_TriggerQueue]
        ), TIMEOUT 10000;""")
        row = cur.fetchone()
        if row:
            convo = row[0]
```

```
            try:
                cur.execute("END CONVERSATION ?", convo)
                conn.commit()
            except: pass
            loop.call_soon_threadsafe(asyncio.create_task, broadcast_tickets())

# in main():
# loop = asyncio.get_running_loop()
# threading.Thread(target=sql_listener, args=(loop,), daemon=True).start()
```

6. Browser JS — WebSocket consumer (KDS UI)

```
const ws = new WebSocket("ws://<server-ip>:9999");
ws.onopen = () => console.log("WS connected");
ws.onmessage = (msg) => {
  const data = JSON.parse(msg.data);
  // update UI state and renderTickets()
};
```

7. Ownership & permissions notes

If messages appear in sys.transmission_queue with error 15517, fix DB principal/orphaned dbo. Common fix: set DB owner to sa: ALTER AUTHORIZATION ON DATABASE::[synopos-cp] TO [sa];

8. Wrap-up & Checklist

Checklist: 1) Service Broker enabled 2) KDS_TriggerQueue, KDS_TriggerService, KDS_TriggerContract, KDS_TriggerMessage exist 3) trg_TempKot_Change exists and fires 4) server.py's listener receives messages and calls broadcast_tickets() 5) Browser WebSocket receives and re-renders tickets