T-SQL static code analysis: Track lack of SQL Server session configuration

2 minutes

SQL Server can be tuned at `PROCEDURE` and `TRIGGER` levels thanks to several `SET` statements that change the current session handling of specific information.

This rule raises an issue when expected configuration is not set or is set with an unexpected value between the beginning of the `PROCEDURE` (or `TRIGGER`) definition and the first statement that is not a `SET`, `IF` or `DECLARE`.

Noncompliant Code Example

When this rule is configured with ARITHABORT and ON.

```
CREATE PROCEDURE dbo.MyProc
AS
BEGIN
  SET ARITHABORT OFF; -- Noncompliant; ARITHABORT is
OFF
  SELECT COUNT(*) FROM MY_TABLE
END;

ALTER PROCEDURE dbo.MyProc
AS
BEGIN
```

```
  SELECT COUNT(*) FROM MY_TABLE
  SET ARITHABORT ON; -- Noncompliant; ARITHABORT is not
set at the beginning of the procedure definition
  [...]
END;

CREATE PROCEDURE dbo.MyProc
AS
BEGIN
  -- Noncompliant; ARITHABORT is not set at all, so default
value of SQL Server will be applied
  SELECT COUNT(*) FROM MY_TABLE
END;
```

Compliant Solution

```
CREATE PROCEDURE dbo.MyProc(@setConfig INT)
AS
BEGIN
  IF @setConfig=1
    BEGIN
      SET ARITHABORT ON;
    END
  SELECT COUNT(*) FROM MY_TABLE
END;

ALTER PROCEDURE dbo.MyProc
AS
BEGIN
  DECLARE @var INT;
  SET ARITHABORT ON;
  [...]
END;
```

See

- SET Statements - SQL Server (Transact-SQL) Documentation