Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

Migrating **SSIS packages from SQL Server 2014/2016 to 2019** is not just a "copy and paste" job.

We need to deal with **package format changes**, **compatibility level upgrades**, and sometimes **component deprecations**.

Here's the full **step-by-step migration playbook**.

1. Pre-Migration Assessment

Before touching anything:

Step	What to Do	Why
Inventory Packages	List all packages in MSDB, SSISDB (Catalog), and file system (.dtsx files).	Know exactly what needs moving.
Check Package Protection Level	Open in SSDT → Properties → `ProtectionLevel` (e.g., EncryptSensitiveWithUserKey).	Avoid password/key mismatch issues after migration.
Review Project Deployment Model	Determine if you're using **Project Deployment Model** (SSISDB) or **Package Deployment Model** (MSDB/File system).	The migration process differs.
Check References	Identify **data sources**, **connections**, **assemblies**, and **custom scripts/components**.	Some older components may be deprecated in 2019.
Document Server Config	SQL Agent jobs, environment variables, SSIS Catalog configuration.	Needed to rebuild jobs/environments later.

2. Environment Setup for Migration

You'll need:

- **SQL Server 2019 instance** (with **SSIS feature** installed).
- **Visual Studio 2019/2022** with **SQL Server Data Tools (SSDT)**.
- Latest **SQL Server Integration Services Projects** extension for VS.

3. Extract & Prepare Packages

For Project Deployment Model (SSISDB)

1. In **SQL Server Management Studio (SSMS)**, connect to **Integration Services Catalogs**.
2. **Export Project** from SSISDB:
    - Right-click project → **Export** → Save `.ispac` file.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

For Package Deployment Model (MSDB/File System)

1. **Export Packages**:
    - MSDB: Use SSMS → Connect to Integration Services → Export each package to `.dtsx`.
    - File system: Copy `.dtsx` directly.

4. Upgrade Packages in Visual Studio

1. Open **Visual Studio 2019/2022**.
2. Create or open a **new SSIS Project**.
3. **Import**:
    - `.ispac` → **SSIS > Import Project**.
    - `.dtsx` → Right-click project → Add Existing Package.
4. When prompted, run the **Package Upgrade Wizard**:
    - **Target Server Version** → Select **SQL Server 2019**.
    - This upgrades **PackageFormatVersion** to match 2019.

5. Resolve warnings:
    - Update **connection managers** to new providers (e.g., `SQLNCLI` → `MSOLEDBSQL` or `ODBC`).
    - Replace deprecated components (`Data Flow Destination Assistant`, `ADO.NET Source` changes, etc.).
6. Rebuild & Validate project.

5. Handle Script Tasks/Components

- **Script Task** or **Script Component** code must be recompiled in the new SSDT version.
- Open each script, click **OK** (forces recompile), and fix any .NET Framework version changes.
- Ensure your target SQL Server 2019 still supports the .NET Framework version you're using.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

6. Deploy to SQL Server 2019

For Project Deployment Model

1. Build the SSIS Project → Output `.ispac`.
2. In SSMS → Connect to SQL Server 2019 → **Integration Services Catalogs**.
3. Right-click **SSISDB** → Deploy Project → Select `.ispac`.

For Package Deployment Model

1. Use **Import Package** in SSMS or **dtutil** command:

```
dtutil /FILE "C:\SSIS\MyPackage.dtsx" /DestServer SQL2019Server /COPY SQL;"\MSDB\MyPackage"
```

7. Reconfigure Environments & Jobs

- **Environments** (SSISDB): Recreate variables, assign values, and link to projects.
- **SQL Agent Jobs**: Update job steps to point to the new server/SSISDB/project paths.
- If using **Package Deployment Model**, update job steps to use the new package locations.

8. Testing & Validation

1. **Unit Test** each package individually.
2. Run **end-to-end workflow**.
3. Validate:
    - Data correctness.
    - Performance (sometimes SQL 2019 runs faster/slower due to execution engine changes).
    - Logging works as expected.
4. Compare execution results & row counts with the pre-migration runs.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

9. Cutover Plan

If this is production:

1. Freeze changes in the old environment.
2. Deploy final updated packages to 2019.
3. Redirect jobs and applications to new SSIS location.
4. Monitor the first few runs closely.

10. Post-Migration

- Backup SSISDB after deployment.
- Document changes in connection managers, script tasks, job configs.
- Keep the old environment in **read-only** for rollback until stable.

Example: Upgrade Command for Package Deployment

If you need to upgrade packages in bulk from the command line:

ISDeploymentWizard.exe /SourcePath "C:\SSIS2014\MyProject.ispac" /DestinationServer SQL2019Server /DestinationPath "\SSISDB\MyFolder\MyProject"

✅ **Key Migration Pitfalls to Avoid**

- Forgetting to change **SQL Native Client** provider (deprecated) to **ODBC** or **OLEDB**.
- Not recompiling **Script Tasks** after version change.
- Losing **sensitive data** in connection strings due to `ProtectionLevel` mismatch.
- Package runtime issues if **32-bit runtime** is needed (still toggle in job step).

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

SSIS 2014/2016 → 2019 Migration Checklist

1. Inventory & Assess: - List packages (SSISDB/MSDB/File System) - Note ProtectionLevel, deployment model, dependencies
2. Environment Prep: - Install SQL Server 2019 (SSIS) - Install Visual Studio 2019/2022 + SSIS extension
3. Extract Packages/Projects: - Export .ispac from SSISDB - Export .dtsx from MSDB/File System
4. Upgrade in Visual Studio: - Import project/packages - Run Package Upgrade Wizard (Target=SQL 2019) - Fix deprecations/providers
5. Script/Custom Components: - Open + recompile Script Tasks/Components - Resolve .NET references
6. Parameterization & Config: - Map project/package parameters - Recreate environments/variables (if using SSISDB)
7. Build & Validate: - Build project - Execute packages locally - Verify logs, connections, row counts
8. Deploy to 2019: - Deploy .ispac to SSISDB (or dtutil to MSDB) - Configure references to environments
9. Update SQL Agent Jobs: - Point to new server/project/package - 32-bit runtime toggle if needed
10. Integrated Testing: - End-to-end runs - Compare performance & results to baseline
11. Cutover: - Freeze changes in old environment - Final deploy - Switch schedules/connection strings
12. Post-Migration: - Perform post checks & documentation - Monitor, backup SSISDB, keep rollback window