**Microsoft Business Intelligence**

**Best Practices**

March – 2015



|  |  |
| --- | --- |
| AUTHORED BY | |
| Name: | Title: | Phone/Email |
| Delora Bradish | Business Consultant V | delora.j.bradish@hp.com |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| VERSION CONTROL | | | |
| Ver. | Date | Change By | Description of Change | |
| 1.0 | 3/1/2015 |  | Initial MS BI Best Practice Documentation | |
|  |  |  |  | |
|  |  |  |  | |

|  |  |
| --- | --- |
| APPROVED BY | |
| Name: | Title: | | Method of Approval | Date of Approval |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |

|  |  |
| --- | --- |
| RELATED DOCUMENTS | |
| Name: | Location: | | Description |
|  |  | |  |
|  |  | |  |
|  |  | |  |

|  |  |
| --- | --- |
| Contributors | |
| Name: | Subject Area: | |
| Kevin Hoskins | Reporting & Analytics / 6.1 Best Practices for Most Reports | |
|  |  | |
|  |  | |

**Table of Contents**

[1. Introduction 3](#_Toc413427827)

[1.1. Purpose of this Document 3](#_Toc413427828)

[1.2. Acrononyms 3](#_Toc413427829)

[1.3. Best Practice #1: Create & Follow a BI Roadmap 5](#_Toc413427830)

[2. SQL Server Configuration 7](#_Toc413427831)

[2.1. The Ideal 40 / 50 TB SQL Server Setup 7](#_Toc413427832)

[2.2. Best Practice Considerations 8](#_Toc413427833)

[2.2.1. Infrastructure 8](#_Toc413427834)

[2.2.2. SQL Server 9](#_Toc413427835)

[2.2.3. Security 10](#_Toc413427836)

[3. Data Modeling for SQL Server Data Warehouses 11](#_Toc413427837)

[4. Database Administration 14](#_Toc413427838)

[4.1. Implementing Data Definition Language (DDL) Scripts 14](#_Toc413427839)

[4.2. T-Sql Scripts for common DDL changes 14](#_Toc413427840)

[5. Data Integration with SSIS 16](#_Toc413427841)

[5.1. Design Considerations 16](#_Toc413427842)

[5.2. Performance Impacting Considerations 21](#_Toc413427843)

[5.3. Additional Considerations 26](#_Toc413427844)

[5.4. Common SSIS Add-Ins 27](#_Toc413427845)

[6. Scripting 29](#_Toc413427846)

[7. Reporting with SSRS 33](#_Toc413427847)

[7.1. Best Practices for Most Reports 33](#_Toc413427848)

[7.2. Best Practices for non-cube Based Reports 40](#_Toc413427849)

[7.3. SSRS Drill-Through Reports 40](#_Toc413427850)

[8. Analytics with SSAS 42](#_Toc413427851)

[9. SharePoint Integration 45](#_Toc413427852)

[9.1. Installing SharePoint 45](#_Toc413427853)

[9.2. MS BI Security Spaces 45](#_Toc413427854)

[9.3. Integrating SQL Server with SharePoint 46](#_Toc413427855)

[9.4. Creating a BI Site Collection 50](#_Toc413427856)

[9.5. Verify Your SSRS Configuration 53](#_Toc413427857)

[9.6. Deploying SSRS Reports to SharePoint 53](#_Toc413427858)

[9.6.1. SharePoint Prerequisites 53](#_Toc413427859)

[9.6.2. Visual Studio Prerequisites 53](#_Toc413427860)

[9.6.3. Deploying reporting objects from Visual Studio 54](#_Toc413427861)

[9.7. Change Diagnostic Logging 55](#_Toc413427862)

[9.7.1. PowerPivot Service 55](#_Toc413427863)

[9.7.2. Excel, PPS, SSRS & Visio Services 56](#_Toc413427864)

[9.8. Possible ShaerPoint Reporting Errors 56](#_Toc413427865)

[9.8.1. Shared Data Source is Invalid 56](#_Toc413427866)

[9.8.2. Permissions to Access the Data Source 57](#_Toc413427867)

[9.8.3. Data source has been disabled 58](#_Toc413427868)

[9.8.4. BISM for PowerView support of query parameters 59](#_Toc413427869)

[9.8.5. BISM for PowerView cannot locate a server 59](#_Toc413427870)

[9.8.6. BISM In memory storage engine 61](#_Toc413427871)

[9.8.7. BISM Cannot connect 61](#_Toc413427872)

[9.8.8. Visual Studio Incompatible versions 61](#_Toc413427873)

[10. Team Foundation Server 64](#_Toc413427874)

# Introduction

This document is intended for solution architects, data architects and developers focusing on delivery of the Microsoft Business Intelligence (MS BI) tool stack on a Symmetric Multi-Processing (SMP) platform. This includes specific data modeling considerations for SQL Server data warehouse, SQL Server Administration, data integration, t-SQL scripting, reporting and analytics (R&A), SharePoint integration, and Team Foundation Server (TFS).

## Purpose of this Document

The intent of this document is to articulate MS BI best practices. HP’s Global Methods for BI can be found in the Methodology Navigator and should be consulted for tool agnostic BI methodology. This document does not intend to focus on BI in general, but rather to provide **MS BI specific** best practices for a successful **MS BI implementation**. This document *is not tool agnostic*, and although MS BI best practices are often applicable to other tools, this document is truly intended for the Microsoft solution architect or developer.

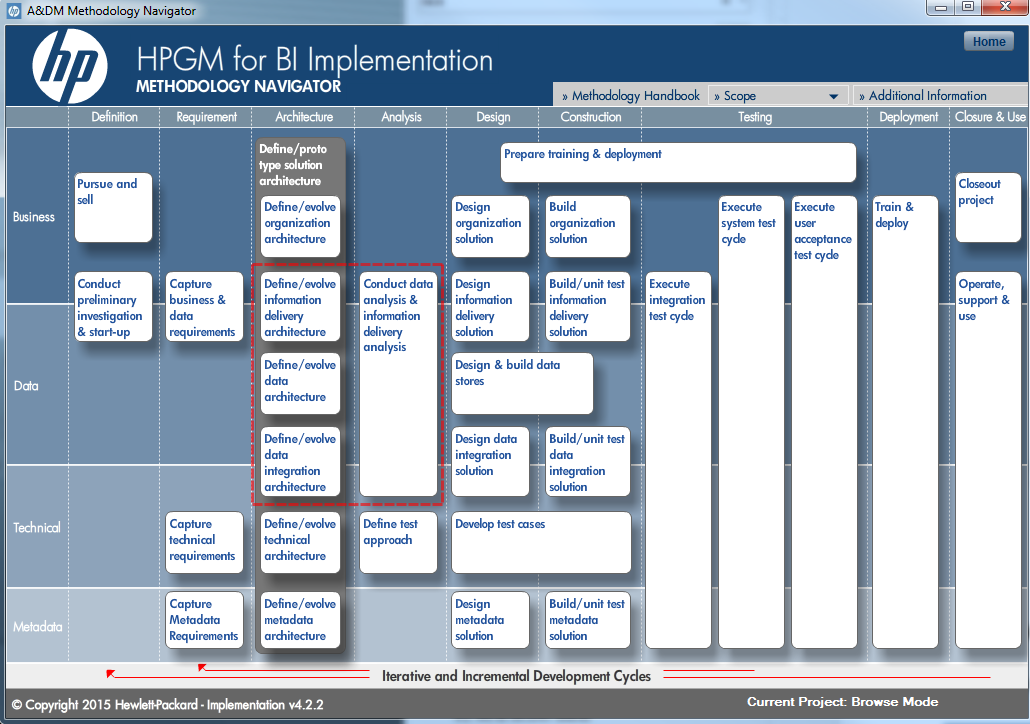
This document recognizes that there are valid exceptions to many best practices. Every client has a unique history, current circumstances, and future goals. The statements given in this document are set out to be guidelines, are not all-inclusive, and largely reflect lessons learned on several MS BI implementations.

## Acrononyms

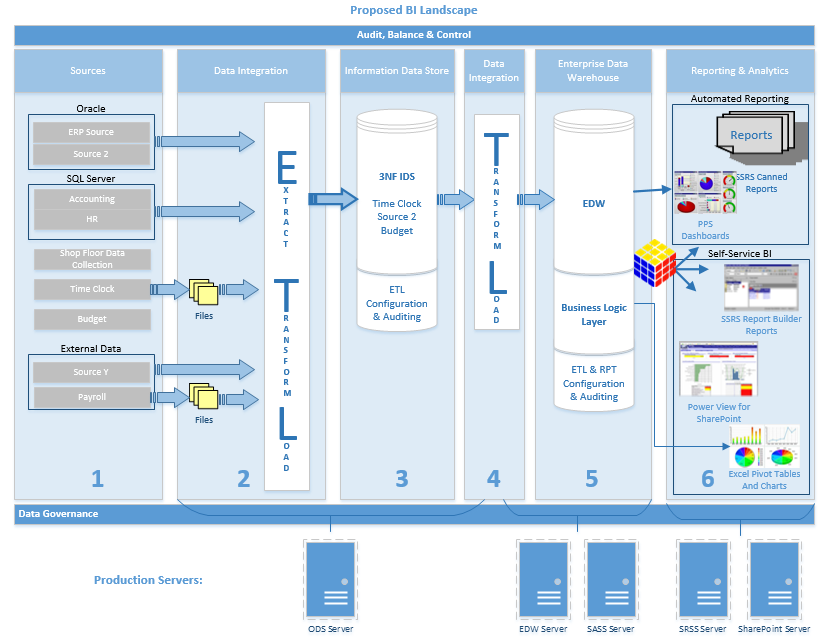
|  |  |
| --- | --- |
| Acronym | Explanation |
| 3NF | Third Normal Form |
| ABC | Audit, Balance & Control |
| BI | Business Intelligence |
| BIDS | Business Intelligence Development Studio |
| BLOB | Binary Large Object |
| CDC | Change Data Capture |
| CLOB | Character Large Object |
| CRF | Common Record Format |
| DDL | Data Definition Language (Create, Drop, Alter…) |
| DEV | Development Environment |
| DLL | Dynamic Link Library |
| DML | Data Manipulation Language (Select, Update, Delete…) |
| DMV | Dynamic Management Views |
| DMX | Data Mining Extension |
| DSV | Data Source View |
| DW | Data Warehouse |
| EDW | Enterprise Data Warehouse |
| ELT | Extract, load and transform (used frequently in MPP architectures) |
| ETL | Extract, transform and load (used frequently in SMP architectures) |
| GUID | Globally Unique Identifier |
| HOLAP | Hybrid Online Analytical Processing |
| IDS | Information Data Store |
| MDX | Multidimensional Expression |
| MOLAP | Multidimensional Online Analytical Processing |
| MPP | Massively Parallel Processing |
| MPP | Massively Parallel Processing |
| MS BI | Microsoft Business Intelligence |
| ODS | Operational Data Store |
| OLAP | Online Analytical Processing |
| OLTP | Online Transaction Processing |
| PDW | Parallel Data Warehouse |
| PII | Personal Identifiable Information |
| prePROD | Pre-Production Environment |
| PROD | Production Environment |
| QA | Quality Assurance |
| ROI | Return on Investment |
| ROLAP | Relational Online Analytical Processing |
| SCD | Slowly Changing Dimension |
| SFTP | Secure File Transfer Protocol |
| SMP | Symmetric Multiprocessing |
| SSDT | SQL Server Data Tools |
| SSIS | SQL Server Integration Services |
| SSMS | SQL Server Management Studio |
| SSRS | SQL Server Reporting Services |
| UAT | User Acceptance Testing |
| UDF | User defined function |
| USP | User Stored Procedure |
| VS | Visual Studio |
| XML | Extensible Markup Language |

## Best Practice #1: Create & Follow a BI Roadmap

HP has a tried and proven BI methodology available via the HPGM for BI Methodology Navigator. This is HP’s roadmap for business intelligence and these best practices for MS BI are intended to support this corporate structure.



You will lose a single version of the truth if you do not follow an organized repeatable process throughout your entire BI project. Similar to a roadmap, for every process there is a most expedient route and an appropriate time.



Although most tools in a BI project can often perform multiple tasks, each tool generally does have a singular purpose of “what it does best”. Consequently, each BI object (ODS/IDS, EDW, MultiD cubes, tabular models, SSRS, Excel, and Performance Point Services) should have an explicitly identified function in the BI roadmap. For example…

Your ETL process is meant for data integration, cleansing, and complex business logic transforms.

Your IDS is intended for data that has been completely integrated, normalized and cleansed.

Your EDW data model is optimized for reporting and is completely sourced from the IDS.

Your SSAS DSV simply points to a series of EDW views, but should not be used for data integration. If your SSAS DSV needs an EDW view change, additional table or a new column, that change request is cycled into the BI project backlog, prioritized, and pushed to production along the standard process of the BI roadmap.

Your report visualization tools are primarily “data formatters”. Reports should not be integrating or cleansing data. They should be selecting from dependable EDW views, EDW USPs, and/or cubes and slicing and aggregating data in an effort to turn data into information.

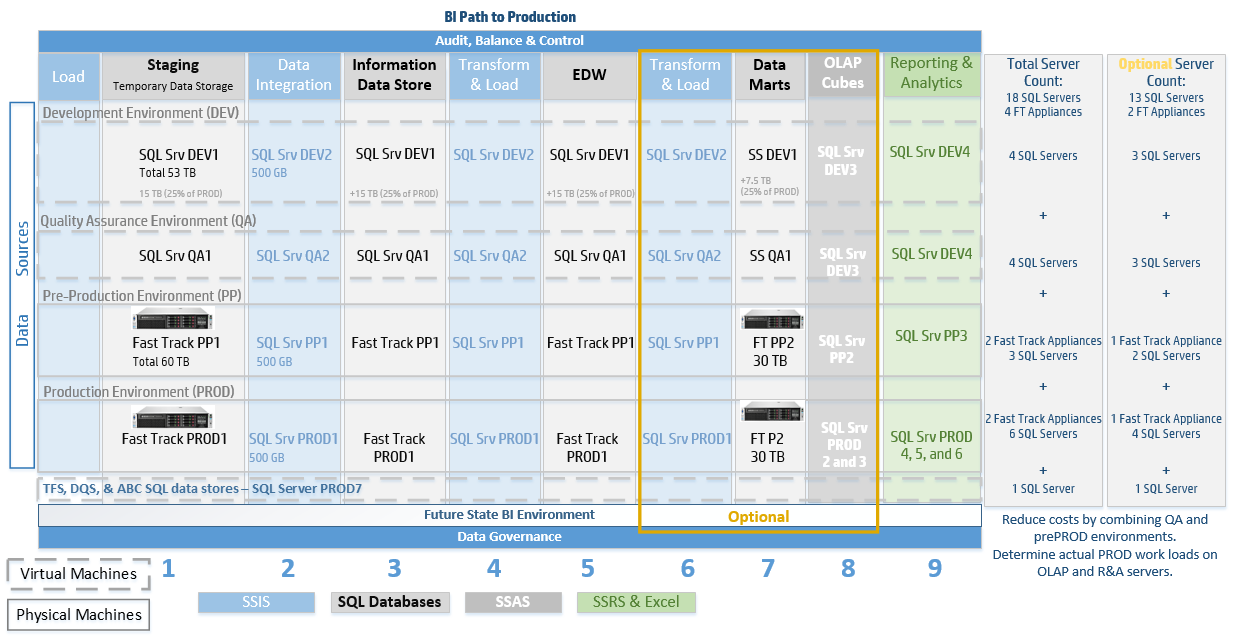
Either PPS, Excel or SSRS should be selected as your dashboard visualization tool of choice, but producing dashboards, scorecards, gauges and KPI indicators in multiple tools becomes confusing and very difficult to maintain.

A good BI roadmap will need to be flexible in the initial planning, but should require very few exceptions. If you have decided on the use of star schemas, for example, but in your first cube you are making exceptions for snowflake schemas, you should probably revisit your BI roadmap. Nothing happens in a BI project ‘just once’, but more often exceptions become precedents.

**Determine to make your first exception to your BI roadmap on the last day of the twelfth month of the fifth year after you go to production!**

# SQL Server Configuration

## The Ideal 40 / 50 TB SQL Server Setup



**General Assumptions:**

1. prePROD and PROD is estimated at 30 – 50 TB of data over a 3 year historical + current fiscal year span of time.
   1. prePROD is an exact replica of PROD.
   2. prePROD is used for UAT.
   3. prePROD is also used for stress testing with actual data volumes
2. DEV and QA are estimated at 25% of total data volume.
3. There is a *near real-time* reporting and analytics requirement.
4. There is full use of all MS BI tool stack components including multidimensional and tabular cubes.
5. A simply recovery model is assumed with backup and recovery undefined.

**The Reality:**

Although most consultants are absolutely delighted to build a brand new BI environment with unlimited finances, it rarely works that way. Also estimating thirteen SQL Servers and four Fast Track Appliances before you even factor in hardware needed for AlwaysOn Availability Groups, may result in your competitor being awarded the delivery. Here are some ways to cut costs and build up to the optimum SQL Server setup.

1. Combine QA with either DEV or prePROD.
2. Determine your actual PROD workloads for OLAP (cubes) and R&A, and reduce SQL Server count appropriately.
3. Identify the risk (!) and possibly reduce your prePROD hardware.
4. Understand the implications (!), and combine your OLAP and R&A servers in DEV and/or QA for a single Information Delivery server.
5. Understand the implications (!), and combine your SSIS and SQL Database servers in DEV and/or QA for a single Data Integration server.

## Best Practice Considerations

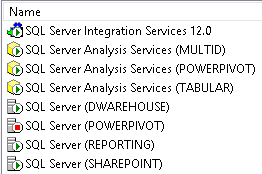
### Infrastructure

1. It is generally better to **have multiple smaller physical discs vs. fewer large discs** in a SQL Server environment. This is a fairly well-known best practice, but for more information, read about Administering Storage, I/O, and Partitioning in MS2012 in Ross Mistry and Shirmattie Seenarine’s book *Microsoft SQL Server 2012 Management and Administration, 2nd Edition***.**
2. **Use virtual servers in a MS BI environment with extreme care**. Virtual servers have many cost and administration benefits that will not be elaborated on here; however, both your infrastructure team and your data warehouse team must have a deep understanding of how SQL Server stores, moves and removes data on the physical file system. It is critical to also understand with whom SQL Server is sharing the physical server components. When in doubt, opt for actual physical servers and avoid the time spent trying to prove out a virtual server setup issue vs. a data integration or report design issue.

If you are going to use virtual servers, first take time to check out *Virtualizing SQL Server with VMware: Doing IT Right* by Michael Corey, Jeff Szastak and Michael Webster.

1. Ideally, SSIS, SSRS, SSAS, your EDW, data marts, SharePoint, and TFS should all be given their own dedicated servers(s). Not seeking to address clustering or AlwaysOn options in this section, having **separate servers for separate processes** is necessary to avoid resource contention. However, in reality, projects rarely have the budgets to put up multiple servers for DEV, QA, UAT, prePROD and PROD. Compromise DEV, QA, UAT and prePROD as needed, but **rarely, if ever, compromise your PROD environment**.
   * If you have SSIS, SSRS and your data warehouse sharing resources, it is almost impossible to trouble-shoot performance problems without completely shutting down two of the three services.
   * Your SSIS packages have many performance considerations without having to take into account trying to avoid SSRS requests or complex t-SQL in views or stored procedures that are not related to data integration.

Creative solutions are often necessary in DEV, QA and UAT. Here is an example of a DEV environment that had a single SQL Server, but used separate instances to minimally separate processes and avoided impacting other BI teams when an instance restart was needed.



If you cannot afford a properly structured PROD environment, you should stop and reevaluate the potential success of your BI project.

* + Starting with a single server for PROD and planning to “expand as needed” rarely works well. Design for **12 – 18 month growth** right at the start.
  + See concerns with virtual environments above.

1. For large data sets, **correlate your SSIS connection manager packet size property with your Ethernet card and network switch properties**. Jumbo frames will most likely be disabled.

### SQL Server

1. **Use filegroups.** Assign tables or indexes to different filegroups and place the file on strategically planned physical drives, RAID arrays or LUNs. Give SQL Server every opportunity to use parallel I/O operations that will process complex queries more efficiently.
   1. Before you plan your filegroups, you must understand your reporting and analysis requirements. Will your data warehouse be primarily used for large sequential data sets, or small random queries focused on the lowest level of granularity?
   2. Multiple disks sharing the same fiber channel switch will yield no benefit. You have to look at your complete physical hardware design and understand if your hardware + filegroups is optimized for your unique reporting requirements.
   3. Generally separate physical storage of heavily used tables, logs, indexes and backup files.
2. **For large data volumes (TBs), use partitioning**. Often sliced by year(s), data partitions improve performance of both ETL and cube processing operations. Minimally, have one large partition for history, and one smaller partition for current data.
3. **Plan your TempDB database!**
   1. Do not leave it on the default drive (generally “C”) where it often is placed during a SQL Server software installation.
   2. Place your TempDB on its own dedicated drive.
   3. By default, calculate TempDB at 30% of total data storage.
4. For OLAP environments, **set your database recovery model to ‘Simple’**. As OLAP environment are not transactional (OLTP) and your ETL process should be written to always be able to rerun past days upon demand, selecting a full recovery model is usually unnecessary.

### Security

1. **For added security, consider** [**TDE**](https://msdn.microsoft.com/en-us/library/bb934049.aspx) **(transparent data encryption).** This becomes critical if your development team is pulling any sort of company data down onto their individual laptops.
2. **Incorporate SQL Server schemas into your security methodology.** A “schema” in SQL Server can be defined loosely as a “container for objects”. ‘dbo’ has always been the default schema, but use this SQL Server feature to your advantage and consider…
   1. A ‘rpt’ schema for report objects (views, stored procedures etc.)
   2. A ‘etl’ schema for objects used exclusively by the ETL process
   3. An ‘ods’ or ‘ids’ schema for objects designated for your operational or informational data store
   4. An ‘edw’ schema for objects denormalized for reporting
   5. A ‘pii’ schema for masked Personal Identifiable Information objects
   6. Application specific schemas, such as ‘app1’ or ‘app2’ for application-specific objects not shared by other functions
3. **Use a separate AD login for each SQL *service*.**
   1. Each SQL Server database instances
   2. SSIS
   3. Each SSAS instance
   4. Each SSRS instance
   5. Each SQL Agent instance
   6. SQL Browser
4. **Use a separate AD login for each SQL *process*.**
   1. SharePoint service logins that will need access to databases, SSAS cubes or SSRS
   2. BISM SSRS login, if you do not use Kerberos
   3. SSAS Administrators
5. Use a **generic** SQL or AD **login** **for all** **object ownership** -- you don't want "John", "Jane" or "Susan" owning any SQL object as no one remains on a project forever.
6. **Manage your sa account**.
   * 1. Use a strong password and change it often.
     2. View you **sa** account as your login of last resort. Instead, assign *sysadmin* privileges to an AD or another SQL-based user login.
7. Consider the **principle of least privilege**, but balance this with empowering your end users to work effectively in a self-service BI environment.

# Data Modeling for SQL Server Data Warehouses

1. **Model the IDS (Information Data Store) / ODS (Operational Data Store) for data integration, not for reporting.** When working with large data warehouse with multiple desperate data sources, and the resulting MDM (Master Data Management) and DQS (Data Quality Services) requirements, the purpose of the IDS becomes data normalization, cleansing and integration. Granted, there are successful DW projects that only have an IDS as they have taken a “Bill Inmon approach” to DW design. However, when you work with Microsoft BI tools that are built on the “Ralf Kimball approach”, it is generally better to follow a source 🡪 Stage 🡪 IDS 🡪 EDW 🡪 cube 🡪 reporting BI roadmap.
2. **Model the EDW for optimized analytics and reporting.** If you assign to your IDS the function of data integration, cleansing and normalization, then your transform and load of cleansed data into an EDW is fairly straight forward. You simply rearrange your tables into a denormalized format without further data cleansing and only additional business logic that is now made possible by the new schema.

If you denormalized your EDW into a star-schema, you will set up your SSAS data source view and resulting cubes to be quite happy! Excel pivot tables will also thank you.

1. **Model for ABC (Audit, Balance and Control).** What we are talking about here is ‘date entered’, ‘last date updated’, SSIS package IDs, and when needed, hash values. Your individual table ABC columns do not need to result in heavy overhead, especially if you keep a well-designed audit (ABC) database. In the least, you should retain the SSIS package run time ID that last performed a DML statement on your table. From this ID you should be able to determine ‘data added to the warehouse’, ‘date last updated in the warehouse’, and “row source system’.

* It is not advisable, but sometimes you may need to keep row-level source system metadata in your IDS. Again, model the standard, and sometimes unique, ABC columns right up front keeping each table’s ABC columns as consistent as possible.
* Coordinate your ABC model with the ETL team. SSIS has built-in event handlers and the ABC model for SQL Server will not be identical to the ABC model for another tool, such as Informatica.

1. **Model in step with agile development methodologies.** As the BI project cycles through iterations, the IDS and EDW model plan needs to remain flexible. Each iteration tends to serve as a learning curve for end users and requirements are consequently changed under established change control. If data modeling gets too far ahead of the data integration, information delivery, testing and production cycle, you may find yourself with a bit of rework.
2. **Do not sacrifice report performance on the altar of an easier ETL or modeling strategy.** The entire purpose of a BI project is to provide a single version of the truth on which to make better business decisions. The data model and ETL effort exists to serve analytics and reporting. Often, this will require complex changes to what might have been a simple data model, and additional data transformations by the ETL process. When a BI development team is on a timeline, it is easy to take the easiest solution and let reporting “deal with it later”; however, don’t do it! Sacrifice your modeling rook or your ETL knight, but keep your analytics and reporting king protected!
3. **Model deleted source records with thought for reporting performance.** This best practice suggestion follows #5 because a delete strategy often falls victim to development time constraints. How the tables are designed has a significant impact on the ETL process. Two popular methods are as follows. *In either case, deleted rows can be referenced at a future time to help explain a data anomaly.*

**Add an ‘IsDeleted’ flag to each table.** This solution is most appropriate for clients who know they will *want to include deleted source system data in their daily reporting and analytics*.

|  |  |
| --- | --- |
| Pros | Cons |
| 1. The ETL process does not need to INSERT to another location and DELETE, but only UPDATE 2. Queries can ignore the IsDeleted flag and select all rows as they were at any point in time. | 1. You’ll need to index this column as every query that wants only existing source system records will need to now filter “Where TableName.IsDeletedFlag = ‘N’” or something similar. 2. Even if you build a set of views for each table that filters out deleted rows, you still have an index consideration. Do you want the overhead of keeping two views per table in sync with DML changes? 3. Without an extra set of views, all of your queries that want the current state of data will now have to filter out deleted rows. |

Here is an actual example of a query from a data warehouse system that elected to keep deleted rows with an indicator flag. Every SELECT and every JOIN had to continually filter on the delete indicator column.

SELECT [columns]

FROM dbo.BILL\_ENTY\_DIM bill\_enty WITH (NOLOCK)

INNER JOIN dbo.CMSN\_REL\_FACT cmsn\_rel WITH (NOLOCK) on bill\_enty.BILL\_ENTY\_SK = cmsn\_rel.BILL\_ENTY\_SK

AND cmsn\_rel.LOGIC\_DEL\_IND = 'N'

INNER JOIN dbo.CMSN\_ENTY\_DIM cmsn WITH (NOLOCK) on cmsn\_rel.CMSN\_ENTY\_SK = cmsn.CMSN\_ENTY\_SK

AND cmsn.LOGIC\_DEL\_IND = 'N'

AND @lUpdateInt between cmsn.CMSN\_ENTY\_EDW\_EFF\_DT\_SK and cmsn.CMSN\_ENTY\_EDW\_TRMN\_DT\_SK

AND (@lEndYM >= LEFT(CONVERT(int,CONVERT(char(8),cmsn.CMSN\_ENTY\_EFF\_DT\_SK,112)),6)

AND @lStartYM <=

LEFT(CONVERT(int,CONVERT(char(8),cmsn.CMSN\_ENTY\_TRMN\_DT\_SK,112)),6)

WHERE bill\_enty.LOGIC\_DEL\_IND = 'N'

AND @lUpdateInt between bill\_enty.BILL\_ENTY\_EDW\_EFF\_DT\_SK and bill\_enty.BILL\_ENTY\_EDW\_TRMN\_DT\_SK

* 1. **Move the deleted records to a ‘deleted’ schema.** This method is most appropriate for clients who want to be able to look up what was deleted from their source system, but 99% of the time, *they are not interested in including deleted data in their analytics.*

|  |  |
| --- | --- |
| Pros | Cons |
| 1. Your index logic and query scripts do not have to continually be aware of an IsDeleted flag column. 2. Deleted source rows are by default omitted from all queries, which is highly beneficial when opening up the data warehouse to self-service BI. | 1. This method is more work for the ETL process as you now have to identify the deleted row, figure out how many table(s) in the CRF, IDS and EDW it impacts, copy the deleted data into a different table, then finally, delete it from the database. 2. Depending on how you implement this, your CRF, IDS and/or EDW databases may end up with two identically structured tables. One named “dbo.Customer” and one named “deleted.Customer”, for example. 3. If the client wants to include deleted source rows in a report, they will have to UNION the two tables together |

**Note:** a “schema” in this context refers to a “container for grouping tables”. It is the leading prefix of a table name in many data warehouse systems. “ods.Customer”, “edw.Customer”, “etl.Customer” and “deleted.Customer” may or may not have different columns, but they do all belong to separate “containers” or “schemas” within a single database.

1. **Incorporate slowly changing dimensions (SCD) only with full understanding of their impact on analytics and reporting (A&R).** Clients tend to like the *concept* of knowing dimension data “at time of fact”. BI data administrators often find the knowledge of “when did this value change” to be extremely helpful when researching data issues. However, very few business users truly understand the impact of grouping data by [last year’s] dimension data values. The majority of reporting requirements are often “current”: current customer, current product, current bill of material, current policy … data.

* To recap briefly, there are six types of SCDs, but the following two are most common:
  + Type 1 – overwrite changed information and therefore all your dimension rows are “current”
  + Type 2 – track historical data by creating second and subsequent rows held together by a key column
* The type of dimension table you choose heavily impacts table design and each ETL package.
* You can obtain current dimension values from a Type 2 SCD table. Your SELECT statement may look something like this:

--get the current value

SELECT [columns]

FROM dbo.FactInvoiceLine fil

LEFT JOIN dbo.DimCustomer dc on fil.cusID = dc.cusID

AND dc.IsCurrent = 'Y'

* SCDs are not governed by a switch. You cannot easily pop back and forth. The same SELECT statement that pulls the current value is not parameter driven to pull the historical value on demand.

--get the historical value

SELECT [columns]

FROM dbo.FactInvoiceLine fil

LEFT JOIN dbo.DimCustomer dc on fil.cusID = dc.cusID

AND fil.InvoiceDate between dc.Eff\_date and dc.Exp\_date)

* If users generally want current dimension data, but occasionally would like to look back and evaluate facts by attributes “at time of fact”, how are you going to provide this functionality?
  + Will you have two sets of views for each dimension table?
  + What type of dimension view will your cubes and tabular models use by default?
  + How will you differentiate a dbo.CurrentCustomerView from a dbo.HistoricalCustomerView for self-service BI?
  + How will you maintain a single version of the truth with users slicing from the dimension type of choice?
  + How will your automated SSRS reports communicate the type of dimension used in the USP data source?

When choosing a dimension type, there are several other factors to consider with many considerations being unique to each client. Do not make the mistake of assuming that “reporting will figure that out later”. *Remember that the data model and the ETL process exist for the purpose of analytics and reporting (A&R), so well thought out A&R decisions need to be made prior to the data model and data integration steps*.

# Database Administration

## Implementing Data Definition Language (DDL) Scripts

1. Although there are several tools, including Visual Studio, that will compare two SQL databases and generate CREATE, ALTER and DELETE statements for objects (tables, views, stored procedures etc.), these tools do not always generate trap for errors and create a corresponding UNDO script.
   * + 1. **Always trap for errors.** When working in large database projects, your DDL scripts will often impact multiple objects. You never want to run the risk of having a partial DDL committed to your database.
       2. **Always use transaction processing.** BEGIN…END TRAN so that if an error occurs you can ROLLBACK. Only COMMIT if all DDL changes occur without error.
       3. **Always generate a corresponding undo DDL script**. This isn’t a ROLLBACK, but is your DDL update in reverse. If you CREATEd something, this script should DROP it. If you ALTERed something, this script should ALTER your object back to its original state. If you DELETEd something, this script should reCREATE it.
       4. **Check both the DDL and the undo into TFS**. You always want to be able to be able to retrace your steps.

**Note:** An excellent tool that has this built-in capability is Red Gate’s SQL Compare.

1. If a table containing data has a DDL change that includes a column with a **NOT NULL constraint**, your DDL should either include a TRUNCATE TABLE (not a good idea once the database is in production) or the DDL must following this seven step process:
   1. Create a temporary table
   2. Drop relationships to child tables
   3. Copy the current table contents to this temporary table
   4. Truncate (delete) the table contents
   5. Alter the table definition
   6. Copy back the data from the temp table inserting a default column value for new ‘not null’ constraints
   7. Recreate relationships to child tables

## T-Sql Scripts for common DDL changes

1. To change a columns **data type**, use the following syntax where ‘nvarchar(50)’ is the new column type.  However, if you are changing between numbers and characters or between Unicode and non-unicode, you may need to copy your data out to a temp table first and following some of the same logic in #2 above.

ALTER TABLE TestTable

ALTER COLUMN TextColumn nvarchar(50)

1. To change a **column that is a FK** (foreign key) in a child table, use sp\_rename.  The relationship to the child table stays intact.

EXEC sp\_rename 'dbo.TestTable.testID', 'TestTableID', 'COLUMN';

1. To change a **column that is an AK** (alternate key) **or UK** (unique key), use sp\_rename as well.  If you have used smart index names, that will require a second step.

--rename your column

EXEC sp\_rename 'dbo.TestTable.IntColumn', 'NumericColumn', 'COLUMN';

--rename your index

EXEC sp\_rename N'dbo.TestTable.UX\_TextCol\_IntCol', N'UX\_TextCol\_NumCol', N'INDEX';

1. To change a **column with a computation**, use this syntax:

--create the computed column to start with

ALTER TABLE TestTable

ADD CalcColumn AS 'This is my calculated column'

--drop the computed column in anticipation of a change

ALTER TABLE TestTable

DROP Column CalcColumn

--recreate the computed column with the new computation

ALTER TABLE TestTable

ADD CalcColumn AS 'This is my new calculated column'

1. To join tables of different collations, use a version of this script:

SELECT x.\*, y.col\_SystemDataType, y.col\_ColumnLength

FROM POC\_Security.dbo.PHI\_METADATA x

LEFT JOIN ServerName.DatabaseName.dbo.vw\_Metadata\_Columns y on x.DatabaseName COLLATE Latin1\_General\_BIN = y.col\_DatabaseName

AND x.TableName COLLATE Latin1\_General\_BIN = y.col\_ObjectName

AND x.ColumnName COLLATE Latin1\_General\_BIN = y.col\_ColumnName

# Data Integration with SSIS

## Design Considerations

1. **A good SSIS package design will be repeatable**. If you find yourself adding new tasks and data flow exceptions to your packages, you need to stop and reevaluate the original layout. One SSIS project will have several “templates” that are reused without a single design change for the life of the data warehouse. You should only have one or two template variations for each of these areas:

* Staging package
* Fact package
* dimension package
* Slowly changing dimension type 2 package
* Control package

The key to manageable QA and the longevity of your ETL process is using a repeatable process. The same thing should happen over and over again as data is moved from staging to your IDS to your EDW. Use t-SQL in USPs to work out complex business logic.

1. **Plan for restartability.** As of SQL 2014, SSIS checkpoint files still did not work with sequence containers. (The whole sequence container will restart including successfully completed tasks.) The solution is to build Restartability into your ABC framework.
2. **Design your SSIS packages so that you can perform an impact analysis when your source changes.** If you have hundreds of DML statements embedded in SSIS tasks, the only way to know for certain what DTSX files will be impacted by a data source DDL change is to pull the XML into SQL Server and parse out the text. One way to do this is through the use of OpenRowSet(), temp tables, and a fairly complex stored procedure.

An easier option is to use well thought out naming conventions and keep all of your DML statements in SPs or UDFs. You can then use DMVs to easily select all SSIS objects that consume a particular column.

As with every design choice, there are ‘pros’ and ‘cons’ and the same applies here. People who embed their DML in SSIS variables and tasks generally do so for these reasons:

* They have simple ETLs that work with one source table at a time.
* They use a heavily parameter-driven SSIS design methodology. If you write your parameter values (DML statements) to your ABC framework DB, you get the same benefit provided by DMVs
* They use many row-level SSIS tasks, such as ‘data conversion’ or ‘derived columns’ instead of JOINs

People who use SPs and UDFs, generally do so for these reasons:

* They have complex data transforms that exceed the [4000 character variable limitation](http://technet.microsoft.com/en-us/library/ms141085(v=sql.120).aspx)
* They want to use DMVs to find table and column usages
* They want to reuse their data transform logic in a verification process

Further from <http://denglishbi.wordpress.com/2011/02/27/ssis-data-flow-vs-hand-coded-etl/>, advantages of an ETL tool vs t-SQL

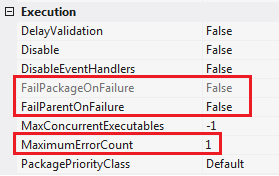
* control flow,
* transformations,
* logging,
* visual representation of work,
* more options than just source-to-target,
* reusability,
* audit details,
* configurations,
* memory management,
* etc.

The point to take away is to consider administration in your SSIS package design. It is worth modifying your optimal design in order to better manage your data integration later.

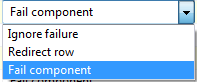
1. **Stage raw data vs. completing data transformations *on the source system***

This is rarely appropriate, but “never say never” as you could have a SQL Server source that is a replicated copy which in effect is staged data at the ready. More likely, you are pulling from a transaction system and you have a very limited time window to “get in and get out”. Best practice is to pull the raw data down into a staging database “as is” and transform and load from staging. This also allows you to look back and view the unadulterated data in staging when values are not landing as anticipated in your IDS or EDW. Experience has shown that when troubleshooting a data transform, it is more effort to test 1.) transforms done on the source system during the pull to staging, and then 2.) additional transforms done in the ETL package. Keep your data transformations in one place.

1. **Verify your ETL process**. Just because your ETL finished without error – or you successfully handled your errors, doesn’t necessarily mean the SUM() of SourceSystemNet equals the SUM() of SSAScubeNet. Use your verification process this way.
   1. It should be the final step of your ETL / ELT process
   2. It should confirm that strategic SUM() and row COUNT() are accurate
   3. It should report on dropped rows discarded during the ETL from an INNER JOIN or WHERE clause
   4. It should automatically send emails of errors that have been allowed to “report and pass”.
2. **Collect Metadata**! Audit, balance and control (ABC) should be planned for and implemented from the very first day. You should store this ABC data in a separate SQL Server database, and at any point in time be able to determine the following:
   1. What packages are currently running? (Your SQL Agent will kick off a master package but won’t tell you what child packages / sequence containers are in process.)
   2. When did a package last successfully execute?
   3. How many records were selected vs. inserted, updated or deleted from any given task?
   4. At what data flow process did a package fail, and where should it restart?
   5. How long did each task take to run?
   6. How long did each package take to run?
   7. What tasks are taking the longest to execute?
3. **Trap for Errors both through On Error events and through precedence constraints.** There are two types of errors to successfully handle in an ETL / ELT process
   1. Report and fail. Use Event Handles and your package properties for this.



* 1. Report and pass. Use your precedence constraints to allow for errors, but always keep row-level reporting so that someone can be notified and the problem can be researched and corrected



With everything else, there must be a balance. It is not appropriate to have every single task with an Error Output. Choose wisely. It is appropriate often to ‘report and fail’, but when there are hundreds of packages and tables churning through your ETL system, you cannot have it breaking constantly. (This is often the case when source systems keep sending through data quality “surprises”.) *Don’t jeopardize the reliability of your data warehouse by not handling a source system that continues to behave badly. The DW is your responsibility.* The source system is often outside of your control.

1. **Avoid hard-coding variables and data connections.**  Instead, use package parameters or project-level configurations. Failing to provide variable values via expressions, and using package-level data connections is often done when first starting out, but the use of a well thought-out SSIS template(s) will help. Anytime you find yourself copying and pasting an object or code, stop and strongly consider the use of a package variables, project parameters, and project-level Connection Managers.
2. **Avoid repeating code** vs. pushing shared scripts into a view or package configuration

If multiple packages use the same object, or business logic, then let that be a shared object that will only have to be updated once in future modifications. Consider the use of views, stored procedures, user-defined functions, package variables, and project parameters.

1. **Do not duplicate your packages for different DMLs or for incremental loads.** SSIS offers excellent control flow and there are often different business rules for each DML task. Some development teams will copy an INSERT package for an UPDATE, or they will copy an ‘initial load’ for an ‘incremental load’, and then make small changes. Now they have two or more packages to send through their QA process. Don’t do it! Use conditional splits and keep the data transform for a single table in a single package.

* Do not assume that you will only need to ‘initial load’ one time. Production might be a one-stop shop, but QA and DEV environment will constantly need rebuilds.
* Design your packages so that one package will build a single table from zero rows to reoccurring inserts, updates and deletes.

1. **Use containers.**  Grouping similar data flows and tasks into meaningful work units. Use the transaction property to commit or rollback a group of tasks as a single unit.

* [Foreach Loop Container](https://msdn.microsoft.com/en-us/library/ms141724.aspx)
* [For Loop Container](https://msdn.microsoft.com/en-us/library/ms139956.aspx)
* [Sequence Container](https://msdn.microsoft.com/en-us/library/ms139855.aspx)
* [Task Host Container](https://msdn.microsoft.com/en-us/library/ms139911.aspx)

1. **Use well thought out naming conventions.** This becomes important in order to understand the data flow and to find objects in the XML.  We suggest naming each object with some convention similar to the following: SSIS Object Abbreviation + Action + ObjectActedUpon

**Examples:**

SEQ XFormLoad Client Sequence container that transform and loads the [Client] table

SQL Truncate ModifiedClient t-SQL task that truncates the [ModifiedClient] table

cid:image027.png@01CEFCAA.C1FA0170

cid:image032.png@01CEFCAA.C1FA0170

DF Load Client  Data Flow task the loads data into [Client] table

RC Selected Client Row count of the client source

cid:image024.png@01CEFCAA.2864AF10

cid:image025.png@01CEFCAA.2864AF10

CSPL…  conditional split



MHSH    Multiple Hash

cid:image028.png@01CEFCAA.2864AF10

OLE\_SRC   data source

cid:image030.png@01CEFCAA.6513CF40

OLE\_DST    data destinations

cid:image031.png@01CEFCAA.6513CF40

DER      derived columns

cid:image034.png@01CEFCAA.C1FA0170

cid:image035.png@01CEFCAA.C1FA0170

Additional popular object naming concatenations are given here. It is up to you to decide to use spaces to separate objects, or underscores.

|  |  |  |
| --- | --- | --- |
| Concatenation | Examples | Definition |
| Task Type Abbreviation  + Action  + Object Type or Schema  + Object Name | “SQL Insert audit TableAuditLog” | An Execute SQL Task object that inserts a row into the audit.TableAuditLog table |
| “DFT Load del Customer” | A Data Flow Transformation Task object that loads data into the del.Customer table |
| “SCR Update ids Customer” | A Script Task that updates the ids.Customer table |
| “SQL Update var AuditPackageBatchLogID” | An Execute SQL Task that updates the Audit…ID package variable |
| “SEQ Extract src1 Customer” | A Sequence Container object that contains multiple tasks for loading the src1.Customer table |
| “CNT Selected src1 Customer”  “CNT Inserted etl Customer”  “CNT Deleted edw CUSTOMER”  “CNT Updated dbo FACT\_INV”  “CNT Unchanged src1 Customer” | A Record Count task that returns the number of rows being selected/inserted/updated/changed/not-changed from/to the [schema.TableName] |
| Object Type Abbreviation  + Server Type  + Server or Instance Name  + Database Name | “PRJ Oracle BI1 FACETS” | A project-level connection manager that is pointing to the Oracle FACETS database on the BI1 server |
| “PKG SQL BI1 FACETS\_COPY” | A package-level connection manager that is pointing to the SQL Server FACET\_COPY database on the BI1 server |
| “PRJ SQL DWAREHOUSE STG” | A project-level connection manager that is pointing to the SQL Server STG (staging) database on the DWAREHOUSE SQL Server instance |
| Object Type Abbreviation  + Server or Data Type  + Object Name | “PKG Excel Calendar\_Date\_Seeding”  /or/  “PKG Excel Time\_Date\_Seeding” | A package-level data connection object that is pointing to an Excel workbook containing date information that is intended to populate the DIM\_CALENDAR /or/ DIM\_TIME tables |

|  |  |  |
| --- | --- | --- |
| Prefix | Example | Definition |
| OLESRC | OLESRC FACETS CUSTOMER | An OLE DB source editor using the FACETS source system CUSTOMER table |
| OLEDST | OLEDST STG src1 CUSTOMER | An OLE DB destination editor using the STG database CUSTOMER table belonging to the ‘src1’ schema container |
| NETSRC | NETSRC FACETS CUSTOMER | A dot NET source editor using the FACETS source system CUSTOMER table |
| NETDST | NETSRC CRF ods CUSTOMER | A dot NET destination editor using the CRF database CUSTOMER table belonging to the ‘ods’ schema container |
| CNT | CNT Changed etl CUSTOMER | A record count of records that were changed in the etl.CUSTOMER table |
| SQL | SQL Update dbo TableAuditLog | An execute SQL task… |
| SEQ | SEQ Package Logic | A sequence container… |
| SCR | SCR Throw Error | A script task… |
| DFT | DFT XformLoad CUSTOMER | A data flow transformation task… |
| SRT | SRT src1 CUSTOMER | A sort task… |
| DER | DER Add Audit Keys  DER Add Modification Type | A derived column task… |
| MRG | MRG CUSTOMER PERSON | A merge tables task… |
| CSPL | CSPL Status etl CUSTOMER | A conditional split task… |
| MLT | MLT Deletes ids CUSTOMER | A multicast task… |

## Performance Impacting Considerations

Microsoft has built in multiple data flow performance features that you can read about here (<http://msdn.microsoft.com/en-us/library/ms141031.aspx>). The following list is not all-inclusive, but the following best practices will help you to avoid the majority of common SSIS oversights and mistakes.

1. **Give your SSIS process its own server.**

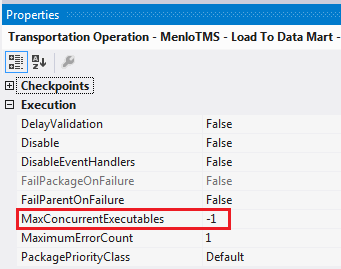
The resources needed for data integration, primary memory and lots of it, are different than for data storage. Granted, if your entire ETL process runs in just a few hours during the night when no end users are connecting, the case can be made to share servers; however, more often, real-time requirements and on-demand transformation are reasons to give your ETL process dedicated hardware.

1. **Only update rows in your data warehouse that have been changed or deleted** from your source system(s).
   1. Use SQL Server’s change data capture abilities whenever possible if working with many updates and deletes in the source system. This works well in a one-source-to-one-destination transform, but can become quite complex in a multi-source-to-one-destination ETL package.
   2. Consider the use of hash values when your source system does not indicate last update date time, or when you transforms involve multi-source-to-one-destination transforms.
2. If possible, **extract into staging only rows that have been changed** in your source system.

Again, you will need a reliable CDC process or last update date time stamp in our source. If you do not have this, you really have little alternative other than to pull all rows into staging. However, if you find yourself in this scenario, try and get the client to agree upon a time box as the cost to extract and hash terabytes of data can have significant hardware cost and ETL package performance considerations.

1. **Take note of your MaxConcurrentExecutables *package property*.**

This defines how many tasks can run simultaneously. The default value is negative one (-1), which means the number of physical or logical processors plus two (2) is the number of control flow items that can be executed in parallel. It is generally recommended that you leave this setting at the default unless you are absolutely sure that parallelism is causing an issue.



1. **Install and test for adequate RAM on the SSIS server.**

Sorting 2TB of data requires 2TB of RAM, and SSIS will start to write the data to disc when all available memory is taken. As part of your Test / QA methodology, you should use Performance Monitor and have your network team notify you whenever *Buffers Spooled* goes above zero or *Avg. Disk sec/Transfer* gets above 10.

Test and monitor the following PerfMon counters.

* + - * % Processor Time
      * Avg Disk sec/Transfer
      * Available Mbytes
      * Buffer memory
      * Buffers in use
      * Buffers spooled
      * Private buffer memory
      * Private buffers in use

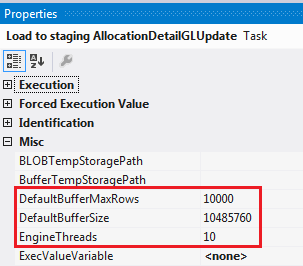
1. **Use the in-memory capabilities of SSIS.**

Developers with strong t-SQL backgrounds can do just about any data transformation directly through a stored procedure or user defined function and may have a tendency to use SSIS only as a data flow container, or for audit logging. However, the memory optimized capabilities of SSIS can have strong advantages over a stored procedure data transform methodology. Here are a few examples:

* + - * Split a derived column transformation into several transformations whereby boosting throughput
      * Split a large flat file into multiple raw files which process simultaneously
      * Take advantage of the cache options available in the Lookup transformations

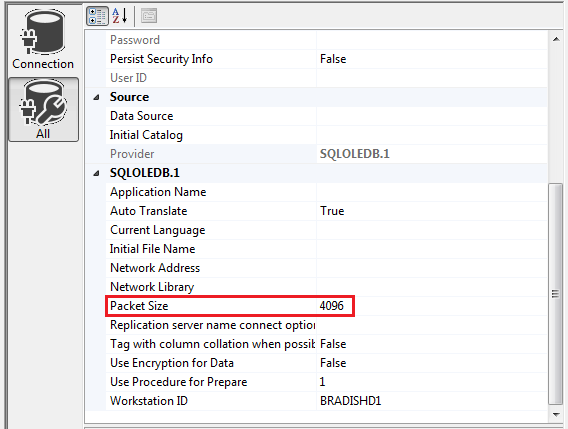
1. **Adequately test and update *data flow properties* that impact performance.**
   1. EngineThreads
   2. DefaultBufferMaxRows
   3. DefaultBufferSize

You can find a plethora of information on these package properties on the Internet, and so their definitions will not reiterated here; however, the important thing is not to push a large ETL project into production *with the SSIS defaults*. Best practice is to develop small, mid-sized and large data transforms with the SSIS package defaults and then take the time to test changes to each of these properties based on volume. Unfortunately, there is no exact science.



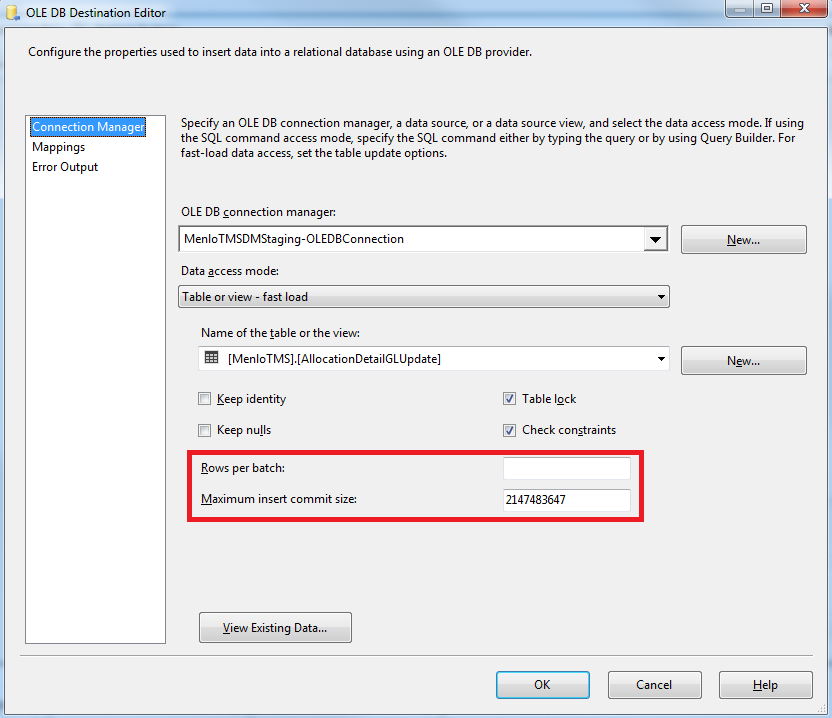
1. **Do not ignore the Packet Size property of your *OleDB Connections.***

This setting determines the size of the packets that are sent across the network. By default, this value is 4096 (4 KB). The maximum value for network packet size is 32768 (32 KB). Your package performance will decrease if your packet size it too large, so test (!). Also keep in mind that your source and destination packet size setting will need to be in sync.



1. **Review the default Ole DB Destination properties that impact performance.**
   1. **Rows per batch:** Estimate of the number of rows in your source; Used by SQL Server to optimize the query plan
   2. **Maximum insert commit size:** If set, will over-ride ‘rows per batch’

As a general rule, the wider the table, the lower your Maximum Insert Commit Size value should be, so keeping the same Maximum Insert Commit Size identical for every size load is an oversight.



1. **Decide on the best TransactionOption property value for each Data Flow and/or Sequence Container.**

By default, the TransactionOption property value is set to ‘Supported’. Often it is appropriate to want an “all or nothing” methodology so you do not end up with a portion of your DML committed. Consider the ‘Required’ TransactionOption in the instances where you would prefer to have all your updates or inserts or none of them.

1. **Use asynchronous transformations sparingly.**

Also known as “blocking transformations”, examples of asynchronous transformation are Data Mining Query, Merge, Merge Join, Pivot, Unpivot, Term Lookup, Union All, Aggregate, Fuzzy Group, Fuzzy Lookup, Row Sampling, Sort and Term Extraction. They require additional buffer memory as output is not synchronous with input. The number of records IN does not equal the number of records OUT. Consider alternatives, like an ORDER BY statement in your SELECT clause, instead of the asynchronous Sort task.

Note: If you do sort your data in t-SQL, do not forget to set your SortKeyPosition and IsSorted properties in Advanced Editor.

1. **Do not load data into indexed target tables with indexes on.**

It is generally best practice to DROP indexes prior to a DML operations, then reCREATE INDEX when the DML statement completes, especially when working with large data sets. Be sure to build this logic into your package template – it can always be disabled for tables without indexes.

1. **Consider placing a clustered index on your staged tables.** Often the business key is the most logical place; however, it really depends on your transformation. Be sure to DROP the index prior to data load and CREATE after data has been loaded into SQL.
2. **Do not rely on implicit typecast.**

When using a flat file connection manager, SSIS treats all columns as a DT\_STR (string). Explicitly convert your numeric values appropriately so you are not forcing SSIS to acquire additional space in the buffer.

1. **Do not use the SCD task that is packaged with SSIS**

There are third-party SCD tasks, including ones available for free from CodePlex, that have much faster performance output. Another option is to write your own SCD methodology, or utilize one implemented at other HP clients.

1. **Write optimized queries!**

Certainly this is never an intent, but it sometimes happens when writing complex select statements from multiple source tables. Code review by senior level developers and the frequent use of query plan analysis in SQL Server Management Studio (SSMS) will help to avoid unoptimized t-SQL.

1. **Use optimized data connection drivers**

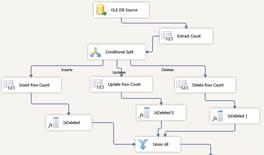
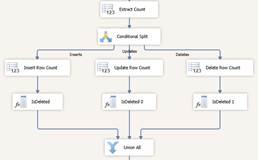
If pulling or writing to Oracle or Teradata, use the drivers for SSIS available from Attunity. If working with SQL Server, understand that the OleDB driver is said to be deprecated in the future by Microsoft, but at the time this document was written, OleDB continues to outperform .NET and ODBC, and remains the top performing data connection option.

1. **Omit unnecessary data.** 
   1. Use CDC or some equivalent on your source system to pull only new, modified, or deleted rows. If you do not have this luxury, incorporate the use of a hash value into your ETL process.
   2. BLOB or CLOB data that should be handled differently in your ETL process. Don’t mix it up with your standard OLAP tables.
   3. Do not confuse this rule of thumb with partial table selections. **If your source system took the time to enter it, you should probably take the time to ETL it.** “Unnecessary data” is rarely one of ten columns in a table, but is usually an entire table. Just because there is no report requirement today, does not mean data being actively entered and updated in a source system will not be required for self-service BI tomorrow. Also, business rules change and a proper DW source-to-EDW is quite involved. What are your resources for cycling through the entire Source 🡪 Stage 🡪 IDS 🡪 CRF 🡪 EDW to add a column into a table that has been running successfully because last month “we didn’t need that”?
2. **Master package design.** Each ETL tool and scheduler has a different built-in methodology, but the important thing to keep in mind is manageability, performance, and minimal impact from errors. The ETL process for an EDW rarely starts with a single master package, but package execution is often sliced up by subject area with consideration to the time allowed to access the source system(s). Keep in mind the following when planning your master packages:
3. Not all your EDW data is likely to run under the same real-time or historical time windows.
4. Packages should be organized so that there is concurrent processing whenever possible. Design package dependencies only were appropriate. For example, FACT\_INVOICE needs to wait for DIM\_CUSTOMER keys, but your HR and Accounting / GL packages can probably be running concurrently.
5. The more granular and independent your master package design, the less likely that an ETL failure in one subject area will have an impact on another subject area.

## Additional Considerations

1. For the sake of the person who “comes behind”, all SSIS task components should be aligned neatly moving top to bottom and breaking out left to right, in straight clean lines.  “Align centers” for columns and “align tops” for horizontal rows.

  becomes  

  becomes  

1. Always **reset your GUIDs** using BIDS Helper when you copy and rename a package. This really is not “cosmetic”, but neither does your GUID impact performance. It is important that all of your packages have unique internal identifiers as these GUIDs are used for logging, and are often written to your audit database.
2. **Be mindful of precious issues.** *Decimal*, *float*, *numeric(x,x)* and *real* data types are not created equal. *Money* is not on the SQL 2012 or SQL 2014 deprecation list, but still use it with caution, and possibly after you review the scripts below, not at all.

Example t-SQL for demonstrating numeric data types in SQL Server 2014.

--first create your test table

CREATE TABLE [dbo].[NumericDataTypes](

[Numeric\_DT] [numeric](18, 6) NULL,

[Money\_DT] [money] NULL,

[SmallMoney\_DT] [smallmoney] NULL,

[Decimal\_DT] [decimal](18, 6) NULL,

[Float\_DT] [float] NULL,

[Real\_DT] [real] NULL,

[Int\_DT] [int] NULL,

[BigInt\_DT] [bigint] NULL,

[TinyInt\_DT] [tinyint] NULL

) ON [PRIMARY]

--insert some rows

insert dbo.numericdatatypes

values(9.999999, 9.999999, 9.999999, 9.999999, 9.999999, 9.999999, 9.9999999, 9.999999, 9.999999)

insert dbo.numericdatatypes

values(100, 100, 100, 100, 100, 100, 100, 100, 100)

--SELECT the resulting data

SELECT \*

FROM [dbo].[NumericDataTypes]

**Result:**



1. Keep current on SQL Server deprecated features. Every new SQL version has a new list. You can find them on MSDN here: <https://msdn.microsoft.com/en-us/library/ms143729(v=sql.110).aspx>

## Common SSIS Add-Ins

1. **BIDS Helper.** The latest version of Business Intelligence Development Studio (BIDS) Helper can be obtained from CodePlex. BIDS Helper offers a variety of helpful add-ins, but minimally you will probably be using the following:
   1. Reset GUIDS in SSIS
   2. Script task expression highlighter in SSIS
   3. Dimension Health Check in SSAS
   4. Delete dataset cache files in SSRS
2. **SFTP.** Visual Studio 2012 does not come packaged with a Secure File Transfer Protocol (SFTP) task. CodePlex offers a few options which can be used with success. The challenging part often comes in getting Visual Studio to recognize the new DLL.
   1. The SFTP task is a control flow, not a data flow, item in the SSIS Toolbox.
   2. SQL Server 2012 stored the DDLs for control flow tasks in c:\Program Files (x86)\Microsoft SQL Server\120\DTS\Tasks.
   3. Your gacutil.exe should be located in C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\bin. If it is not, find gacutil.exe on your PC and copy it to this location.

**Note:** gacutil.exe can sometimes also be found in C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322

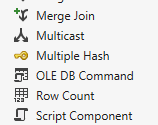
* 1. Instructions for adding a DLL to the VS SSIS toolbox
     1. Open a command window.
     2. Enter cd C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\bin or set PATH=%PATH%;C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin
     3. To add the assembly to the cash run gacutil /i "c:\Program Files (x86)\Microsoft SQL Server\120\DTS\Tasks\[the name of your SFTP DLL].dll"
     4. If later you want to remove the assembly from the cash run gacutil /u "c:\Program Files (x86)\Microsoft SQL Server\120\DTS\Tasks\[the name of your SFTP DLL].dll"
     5. Open VS 2012, and create or open a SLN (project solution file).
     6. In VS 2008 or SSDT right mouse click on the Toolbox and select ‘choose items’. Find the new SFTP on the SSSI control flow tab and mark it.

/or/

In VS 2012 select right mouse click on your SSIS Toolbox and choose ‘Refresh Toolbox’ as the ‘Choose Items’ option for the SSIS Toolbox has been removed.

1. **Multiple Hash**. If your source system does not have a way to indicate what rows are new, or recently updated, you may need to use a hash task in SSIS. One does not come packaged with SSIS, so you’ll need to tap into CodePlex once again. The Multiple Hash 1.6.3.x comes with a nice GUI install program so you do not need to go through the process of adding the Multiple Hash DLL to VS manually.

When complete, you will find MultipleHash2014.dll located in C:\Program Files (x86)\Microsoft SQL Server\120\DTS\PipelineComponents. Open up VS, and drill into a control flow object. Refresh your SSIS Toolbox, and you should now see ‘Multiple Hash’ under ‘Common’.



# Scripting

1. Remember that someone always comes behind you. There is no prize for complexity. There is an inherent reward for kindness.
   1. **Document your code.** Every t-SQL, VB or C# script (etc.) should indicate the purpose, the author, the date written, how to run the script, and anything else that will be helpful to the next developer. Example of opening t-SQL in UDF:

/\*

Date Created: 1/5/2011

Description: Parses multi-value parameters into an varchar list

Where Used: SSRS multi-valued parameters

ETL objects that parse multi-valued delimited columns

To run: SELECT \*

FROM dbo.udf\_SplitDelimitedList(';', (SELECT ColumnNameList

FROM TableSource

WHERE ColumnName = 2))

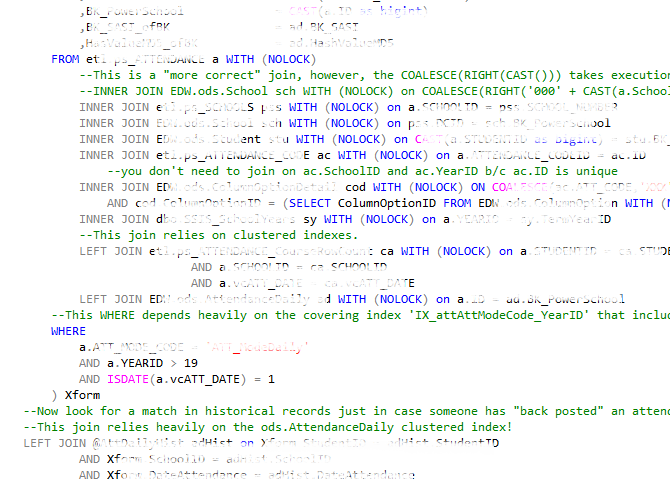
SELECT \*

FROM dbo.udf\_SplitDelimitedList(',', '123ABC, 1.25, 1, Column Value’)

Revision History:

\*/

Basically, leave quite a bit of “green text” in your t-SQL statements, or the equivalent SSIS script tasks. Focus on business rules, tell the ‘why’, and articulate anything that was assumed at time of script. Color representation of a well-documented SELECT statement:



* 1. **Format your scripts.** Indent. Keep it clean. Make is easy to read. You know the drill!

**Note:** Red Gate software sells a script formatting tool if you want to automate this effort.

1. **Use transaction processing.** There is explanation and multiple examples of transaction processing from MSDN and TechNet. The important thing is that you avoid partial commits by wrapping your INSERT, UPDATE, DELETE, ALTER, CREATE etc. statements in a transaction. Explicitly reporting on and handling errors, recording row counts impacted and controlling your work flow is a sign of mature software development and programming.
   1. BEGIN TRAN … END TRAN
   2. TRY... END TRY
   3. MERGE(). Even though SQL’s MERGE() statement is an “all or nothing” statement by design, you may still wish to consider wrapping it inside another transaction to perform some sort of action ‘On Error’.
   4. SSIS’s TransactionOption property. Read more on this under [Performance Impacts](#_Performance_Impacting_Consideration) for SSIS above.
2. **Trap for errors.** Although your Windows Application log, SSISDB database and SQL Agent report errors, explicitly trap for, report to an audit database, and handle your own errors. Take responsibility for your work flow and view Windows and SQL Server default error handling as a secondary level of information.
   * + 1. **Report and fail.**
          1. In SSIS, this is your typical error handling routine generally built into an ETL package through a package-level OnError event handler.
          - Always write the error Id and message to your audit database.
          - Pause the package (vs. fail) for a set time. This is helpful when row locking is causing the problem.
          - Automatically retry running the package beginning at point of failure.
          - Permanently fail the package reporting failure to the calling master package or scheduled job agent.
          1. In t-SQL, this is your ROLLBACK statement.
          - Always write the error Id and message to your audit database, even if this same error is being reported by SQL Agent job history and/or recorded in the SSISDB database.
          - Record in your audit database where to restart, especially if you have nested transactions.
   1. **Report and pass**
      * 1. In SSIS, when you are churning through hundreds of tables and terabytes of data in a daily ETL process, it is not always appropriate to stop the entire ETL process when a source system sends down an unexpected data integrity issue. For example, if your customer address package fails, would you rather have no invoices for the day, or at least obtain all your facts attributing what will now be “early arriving facts” to a -1 (unknown) customer address dimension Id? Most clients opt for their invoices! Keep these things in mind when making this decision:
           + This is a design decision in your ETL package, and development of a report and pass methodology takes programming time and additional testing, including the sending of automatic e-mails with report attachments.
           + The client has to be committed to acting upon the automated e-mail and fixing the issue in their source system(s), or this will eventually result in imbalanced analysis with too many fact records being attributed to unknown dimensions, or incomplete fact records with (possible) defaulted zero integer values.
           + Choose wisely! HP recommends that “report and pass” be implemented in the ETL packages only after testing shows where data errors may reoccur under special circumstances. It is not appropriate to “report and pass” after *every* INSERT, UPDATE or DELETE task.
           + “Report and pass” should never replace “report and fail”. Every ETL package should have a “report and fail” OnError Event because it is simply not possible to foresee every network, hardware and data occurrence that can cause a package to throw an error.
      1. In t-SQL with transaction processing, you can still COMMIT after an error. But again, the developer should take ownership of the work flow.
         * + Report meaningful information including PKs and BKs to your audit database so that the offending rows can be found, SELECTed into a report, and acted upon.
           + Do not overlook INNER JOIN statements and WHERE clauses that are often used to avoid data errors. This is effective error handling as well, but you still need to report data problems that you have explicitly avoided through well written SELECT statements. In a proper BI environment with ABC (audit, balance and control), reporting on this type of “error” becomes part of your data verification process – one of the final steps in data integration. Do not overlook it!

**Note:** It is *not advisable* that OnError event logic in individual packages, USPs, or UDFs send automated e-mails. As you are reporting errors to your audit database, **have a single audit database routine send out e-mails** based on rows written to your audit database error table.

1. **Collect metadata** on …
   1. DML row counts
   2. Execution begin and end times
   3. Both successful executions, and failures
   4. User IDs, even if a service login

At any given time in an ETL or ELT process, you should know the following:

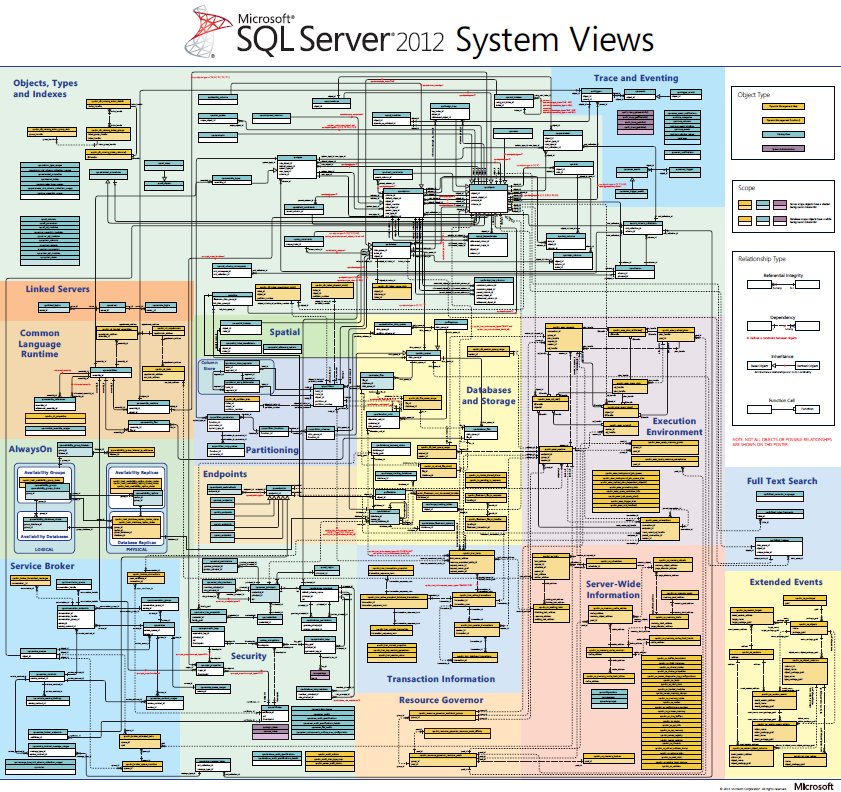
1. Who is performing the action?
2. What DMLs and /or SSIS packages are currently executing?
3. What ETLs ran on [This Date] for any database on any server? What was the total execution time?
4. What rows were added to [Table Name] and by what ETL package?
5. How many rows were deleted or marked for deletion between [Begin Date] and [End Date]?
6. **Are all rows SELECTed finding their way down an INSERT, UPDATE or DELETE path? Is my ETL business logic dropping data?**
7. What is my **row count trend** over the last 3 months?
8. Why is [SSIS package name] suddenly running 25% slower than it has been running over the previous month?
9. The client reports that the EDW is “wrong”, and your sustain team needs to show that the row in question has inserted on [This Day] and last updated on [This Day], and was sourced from [This Source System(s)].
10. The **ETL fails mid-run**, which rows were successfully loaded and which were not? At what point should the master package restart?

**Note:** Just because a SSIS package or DML statement did not report an error, does not necessarily mean there is no problem. By writing both structural (data bout the containers of data) and descriptive (data about data content) metadata to your audit database, you can **conduct some basic analytics and discover trends**. *Your metadata should help you to be proactive in managing your BI environment!*

1. **Understand the impact of the WITH (NOLOCK)** query hint in OLAP environments.
   1. This query hint may cause the query to read in-flight data that has not been committed and may still be changing or rolled back.
   2. A Snapshot transaction isolation level is an alternative, but understand that you will need to tune and closely monitor your TempDB as the Version Store is kept there.
   3. If your OLAP environment is near real time, the WITH (NOLOCK) query hint will be highly problematic and most like not appropriate.
2. **Use caution with the t-SQL MERGE() statement.** Microsoft’s MERGE() statement has made tremendous improvements since its appearance in SQL Server 2008. Before you make it a permanent fixture in your ETL process …
   1. Test it against a stand-alone INSERT and UPDATE statement and the in-memory capabilities of SSIS.
   2. Understand that MERGE() can be a way of handling SCD (slowly changing dimension) transforms.
   3. Explicitly identify the advantage(s) that you are trying to achieve by using a MERGE() statement in your ETL methodology.
3. **Never use ‘SELECT \*’.** This is just bad form under any context and is rarely disputed. Although it is more work to explicitly list / select all columns from a table, it is a layer of protection. New columns do get added and columns get rearranged. Using ‘Select \*’ simply ignores potential changes in your DDL. It also sets you up for pulling CLOB and BLOB data that 1.) may be completely unnecessary and 2.) should most likely be handled in a separate ETL process and with Hadoop.
4. **Use TRUNCATE instead of DELETE TABLE.** The short version of this best practice is that DELETE will fill up your transaction log faster than TRUNCATE. To use TRUNCATE, you may need to drop referencing table relationships, but again, take ownership of your data flow and referential integrity and govern the process.
5. **Take advantage of SQL’s** [**system views**](https://msdn.microsoft.com/en-us/library/ms177862.aspx)**.** Object Catalog Views and Dynamic Management Views (DMVs) are both extremely helpful when managing a SQL database. Use these built-in system views to …
   1. Select real time data dictionary metadata
   2. Perform an impact analysis on object dependencies
   3. Select real time performance statistics
   4. Easily find SQL property and version settings

If you are using SSAS, do not overlook [SSAS’s DMVs](https://msdn.microsoft.com/en-us/library/hh230820.aspx). They can be used to monitor server resources as well as provide information on SSAS structural content.

**Note:** You can [download](http://www.microsoft.com/en-us/download/details.aspx?id=39083) a system views map from Microsoft which groups the plethora of system views into 17 categories and will assist you in identifying their use. The following graphic is not meant to be readable, but rather to provide a pictorial idea of the number of system views available to you.



# Reporting with SSRS

## Best Practices for Most Reports

1. **Create and use a standard SSRS company template.** There is value to having all of your automated reports contain a similar look and feel. Examples:

* Your ‘printed date/time stamp’ for example should be in the same place on every report.

="Printed: " & Globals!ExecutionTime

* Your company logo should appear in the same place for each report
* If you use complex filtering, a ‘report subtitle’ parameter should be required

=Parameters!l\_Subtitle.Value

* The report parameters passed should always print on the report footer (not page footer).

=First(Fields!ReportFilter.Value, "RptDataSet")

* Your report URL should print on your report footer

=Globals!ReportFolder + "/" + Globals!ReportName

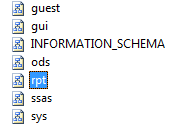
* Your report title and subtitle should be the same font on each report

Come up with your own template to include those features important to you and stick to it!

1. **View SSRS as a “report formatter”.** Push all analytic processing and aggregating back onto SQL Server views, stored procedures or your ETL process. Leave SSRS to only do formatting! Pulling 100K + rows into SSRS and then grouping and sorting via SSRS is a big performance hit.  Pass to SSRS the absolute minimum number of rows (!).
2. **Use shared Datasets.**

* You will have many “pick lists” for report filtering.  For example, create a single ‘customer’ pick list, and use this same dataset for all reports.
* If more than one report uses the same business logic, do not repeat it in each report, but maintain this shared logic in a shared dataset.

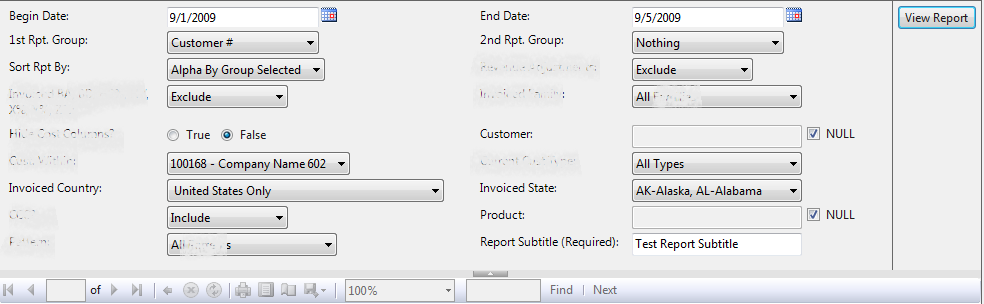
1. **Use a separate SQL Server schema for report objects**. Commonly, this is something like ‘rpt’ or ‘report’. Coincide your schema with current security methodology.



1. **All reports coming from the EDW should have a date/time ‘last updated’ stamp.**  If a report pulls from multiple fact tables, then minimally give the date/time of the last successful ETL master package execution.

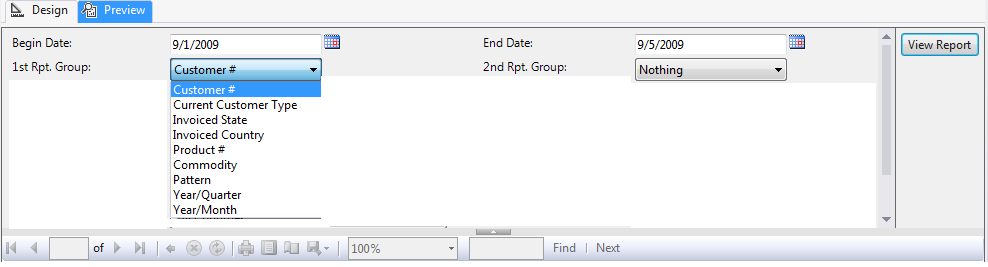
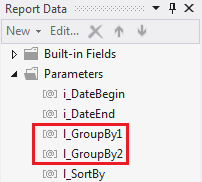
="Data Last Updated on: " & Parameters!l\_LastUpdate.Value

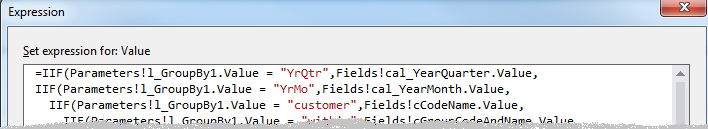
1. **Determine maximum wait time for your reports.**  This will impact your USP scripts.  You may need to use SSIS to denormalize frequently accessed data.
2. **Share detail reports** as much as possible. You do not want to have to spend the development time to write and maintain three “order detail” reports, for example. Create one “order detail” that contains a plethora of columns and filter parameters (see “Use complex filtering” below), and drill through to a single report from all relevant aggregated / summary reports.
3. **Use complex filtering** vs. multiple reports that have the same columns. If you need invoice information for “top 20 customers” and someone else needs these same columns for “category a, c and z”, create one report w/ filters to accommodate both.



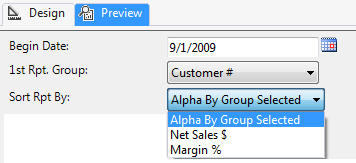
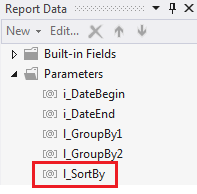
**Note:** It is good practice for complex filtering to also provide an ad hoc Report Subtitle. Although you are showing the report parameters on the report footer (!), end users also need to indicate what their report is about for their own reference.

1. **Use a ‘group by’ drop down pick list.**  Do not create multiple identical reports with the only difference being the grouping. This is one of SSRS best features!  Use appropriate parameters along with *Row Group* and *Text Box* expressions to save yourself valuable development time spent on replication.

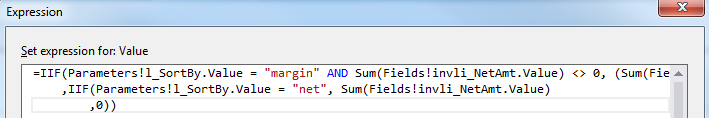




1. Use a ‘sort by’ drop down pick list. Although you can set interactive sort options for each text box, it is nice to give end users the ability to sort the underlying data as a report parameter. You use the same development methodology as dynamic ‘group by’ explained above.



Row Group Sort By Expression:



1. Allow your users to **interactively sort** report columns. Even if you provide a sort parameter, still allow users to sort columns. This is a SSRS feature that is easy to implement for non-expression based columns, and gives immediate value to users.



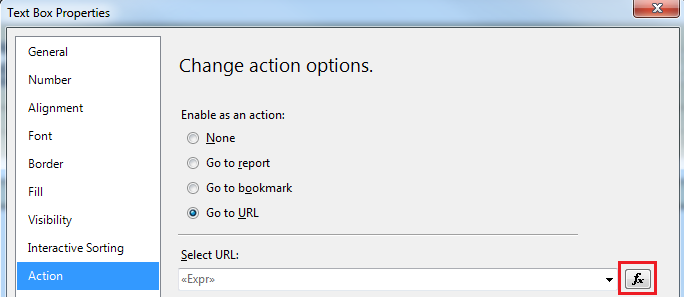
1. **Use an “export to Excel navigation URL” button** vs. two different reports. Without this ….

* Formatted SSRS reports export to Excel with merged columns making further data analysis very difficult.
* You end up having to maintain two separate RDL files – one formatted report that exports w/ merged cells, and one report that has limited to no SSRS formatting, but exports to Excel without merged cells.

Example Textbox on SSRS Report:



Textbox Action Expression:



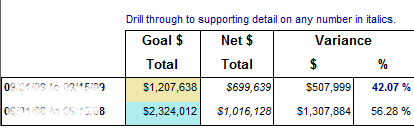
**Expression If Report has No Parameters:** =Globals!ReportServerUrl & "?" & Globals!ReportFolder & "/" & Globals!ReportName & "&rs:Command=Render&rc:Simplepageheaders=true&rs:format=excel"

**Expression If Report has Parameters:** =Globals!ReportServerUrl & "?" & Globals!ReportFolder & "/" & Globals!ReportName & "&rs:Command=Render&rc:Simplepageheaders=true&rs:format=excel&dtStartDate=" & Parameters!dtStartDate.Value & "&dtEndDate=" & Parameters!dtEndDate.Value

1. **Drill Through Report Best Practices:**
   1. Make sure that the **sum total found on the parent report is repeated and easily found** (via color or font etc.) on the child report. The parent report will most likely show the total on a detail line, while the drill through “child” shows the same total on a group or report footer line.
   2. **Rarely, if ever, pass aggregated totals** to a child report. Use shared datasets to provide multiple reports with the same summaries.
   3. Whenever possible, **source both the parent and drill through reports from the same EDW tables** or cube measure groups. Use a shared data set in your report project for both. (To avoid having SSRS group data, use a stored procedure in your shared data set that has a group by input parameter whereby returning different levels of granularity on demand.)
   4. Every drill-through report should be able to be **run independent of a parent**.  This means that all security and report parameters will need to be in both the ‘parent’ and the ‘child’ report. You lose significant ROI by only making ‘child’ reports available from a ‘parent’.
   5. Consider **additional filter parameters in child reports**.  By nature, they contain more detail and should not be restricted to the more aggregated parent filter parameters.
   6. If a report only shows an aggregated percentage, **the drill-through should provide both the numerator and denominator detail.**  Use security roles to protect PII data, but appropriate end users must be able to be able to see the ‘A’ and the ‘B’ that resulted in ‘C’.  If not, it will be the BI team who has to prove the report calculations are correct by ad hoc t-SQL data exports again … and again … and again.  Consider a subreport or additional tablix that is completely security-driven to hide/display based on user login. Examples:

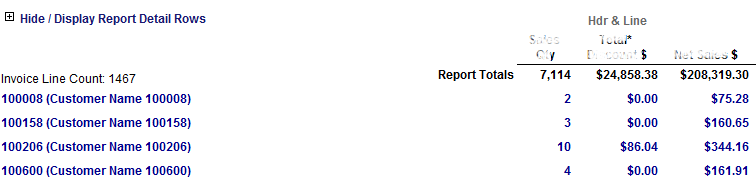
* Aggregated percentage of total hours billed to calculation of personnel hourly cost to company
* Aggregated percentage of total store hours open to total personnel hours worked
* Aggregated sum of total payroll providing drill through to employee PII data and subtotals.
  1. **Avoid toggling detail rows**. Toggling is a “hide / display” feature in SSRS. “Go to Report” is an action item. Both provide what most people think of as “drill through capabilities.” However, these two options are far from equal. “Go to Report” action item is generally much more efficient.

Example of *Go To Report* action. The detail is only brought from the server and formatted on demand, and only for the single row on which the user drills through.

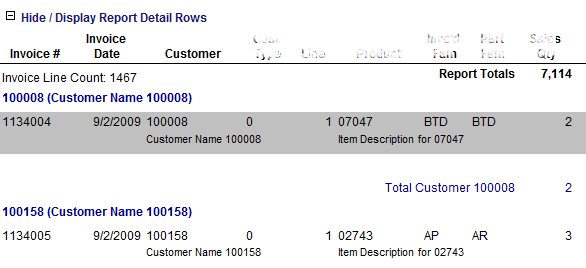


Clicking on a number is this column results in the instantiation of a 2nd SSRS report. Parameters are passed from the parent report and only the detail rows behind this single number are shown on the child report. The end user can at that time modify the child report parameters and continue working to find the information needed.

Example Toggle (Hide/display):

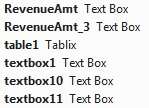


The “toggle” has actually caused SSRS to pull all of this data into the report, even if you don’t need the detail. This results in longer report formatting.

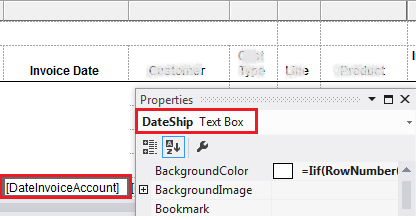


1. **Always provide a tablix group subtotals, tablix table grand totals, and report totals.**
   1. An end user shouldn’t be expected to sum things up in their head.
   2. For QA, the report can do the work of providing totals that would otherwise need to be computed by the QA team.
   3. Even if the total is redundant, visually seeing that ‘tablix1’ sums up to the same result as ‘tablix2’ provides validity to the detail.
      1. **Report object names are important.**  Someone always comes behind you and will have to maintain what you are written.  Take time to make ALL OBJECT NAMES meaningful and correct (!!). This includes project, shared data sources, shared data sets, actual report, parameter, image, text box, tablix, and built-in field names, to just name a few.

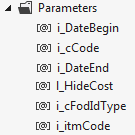
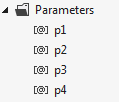
If you need to reference a text box in an expression, how meaningful is ‘textbox10’? If you need to reference revenue in an expression, what makes *RevenueAmt\_3* different from *RevenueAmt*?



**If you change the data source of a text box, be sure to also change the name of the text box.** “DateInvoiceAccount” data should not be in a textbox named “DateShip”.

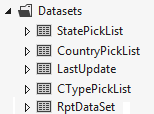


Looking only at the parameter names, which parameters are input parameters (i\_) to the report data source stored procedure, and which parameters are only used in the local report (l\_)?

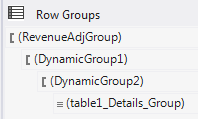


What is “p2” and which of these parameters are dates?

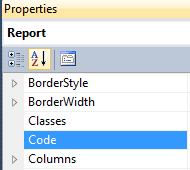
Looking at these report data sets, which ones are used for report parameter pick lists, and which one is the primary data set behind the report?



Meaningful names matter everywhere, even in Row Groups!



* + 1. **Use Public Shared Functions.** If you find yourself coping/pasting business logic in expressions, use a report level Public Shared Function. You will find these under the “Code” property of your Report object. As of SQL 2012, you can only write in Visual Basic and there is no IntelliSense, but this SSRS feature will allow you to write functions that can be called from anywhere in your report.



Example usages:

* Format a text box to empty or special string based on some input condition
* Return a -1, 1 or 2 integer value for KPIs

**Example VB Scripts:**

Public Shared Function FormatCurrency(ByVal input as NULLable(of decimal)) as string

RETURN IIF(IsNothing (input), “N/A”, Format(input, “##,##0.00;(##,##0.00)”))

End Function

Public Shared Function ShowTrendArrow(currValue as double, prevValue as double)

if (currValue = 0 or prevValue = 0) Then

ShowTrendArrow = 2

elseif (currValue <prevValue) Then

ShowTrendArrow = 1

else

ShowTrendArrow = - 1

end if

End Function

* + 1. **Use expressions in BackgroundColor property with caution.** Switching every-other report row from grey to white actually does look very nice and enhances readability; however, implement carefully. *This also results in noticeable performance degradation in large reports.*

=Iif(RowNumber(Nothing) Mod 2, "Silver", "White")

## Best Practices for non-cube Based Reports

1. **Never imbed SELECT statements in a report.** Having your report sources in SQL Server USPs or UDFs allows them to be easily searched via SQLs dynamic management views.  Having them embedded in each RDL requires an XML file search.
2. **Standardize on USPs for all report sources.** USPs provide the most functionality.  You can use views and UDFs some of the time, but you WILL USE stored procedures for complex analysis.  If you don’t’ standardize on USPs, then you are jumping between multiple object types.  This starts to get confusing – does that data already exist in a view?  Oh no, it’s in a UDF.  Well, I just created it again in a USP b/c I didn’t find it.  … So, things start to get messy.  Standardize on USP b/c they can do everything.
3. Use a standard **USP template** that includes a ‘header’ for change tracking

/\*

Author: DeveloperName

Date Created: TodaysFullDate

Description: SELECT for NameOfReport

Where Used: SSRS or SharePoint URL

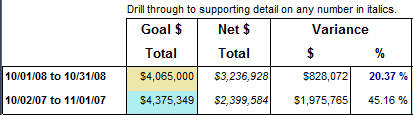
To run: EXEC spRPT\_MyRDLName Param1, Param2, Param3, Param4

Revision History:

\*/

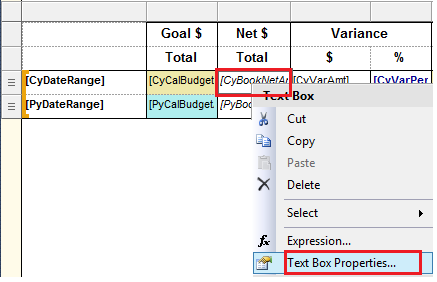
## SSRS Drill-Through Reports

First, unless every single row and column label is a drill-through, I would suggest deciding on some sort of indicator, like italics, to tell the end user that a particular column header or row number has drill-through capabilities. In the example below, only “Net $ Total” has drill-through capabilities.

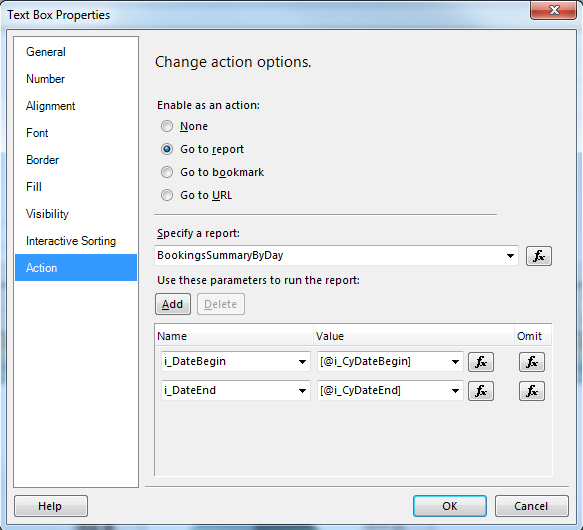


Keep in mind that every drill-through report is a stand-alone SSRS report deployed to your SharePoint environment. Users won’t always have to go through a “parent” SPOT report to run a detail report. Provide default report parameters, deploy and test each drill through on its own. (When you call the “child” report, you’ll be passing report parameters from the “parent” which will over-write the default.)

To create drill-through capabilities, in SSRS Design View, right mouse click on the column and choose “Text Box Properties”.



Next, go to the “Action” tab.



IF your “parent” and “child” reports are deployed to the same SharePoint site, you can simply specify the report name. You do not need to [use a URL](http://technet.microsoft.com/en-us/library/dd239346.aspx). The rest of the “Action” tab is pretty self-explanatory.

# Analytics with SSAS

* + - 1. Remember that your SSAS cube processing performance is directly related to the source table storage. A proper filegroup and file storage options supporting your EDW impacts not only view, USP and UDF response times, but your cubes as well.
         1. **Manage your EDW indexes.** What is the first t-SQL script you see SSAS executing when you process a dimension?

SELECT DISTINCT column1, column2, column3, column4, column…[n] FROM dbo.DIMTableName

If your EDW table does not have a clustered index, SSAS has to add a distinct sort to the query plan. This add time. Adding a column store index on the columns used in your dimension may improve performance; however, all objects have to work together. An index strategy for SSAS may not be the best solution for ad hoc end user queries being executed against the EDW for other purposes.

* + - * 1. **Create a semantic layer of views in your EDW.** SSAS is part of your information delivery layer, and many of the same best practices which apply to reporting, apply to cube design. Neither the SSRS nor SSAS DSV (data source view) should be accessing EDW tables directly. Always point to EDW views and …

Give yourself a layer of protection against structural changes in your EDW.

Add simple business calculations to views that can be shared out immediately to all objects (USPs, UDFs, reports, cubes and Excel workbooks) consuming the view.

**Note:** An example of a simple business calculation is a subtraction or addition that acts upon two existing EDW columns in the same table. Complex business logic should almost always be pushed back to your ETL process, USP or UDF.

* + - * 1. **Consider a NOLOCK or TABLOCK query hint.** If this is not already part of your EDW view logic, you can add these query hints in your SSAS DSV.
      1. **Avoid using your SSAS Data Source Views (DSVs) for any sort of data integration.** This is what your ETL/ELT process is designed to do! Although you can join tables from multiple data sources, create aggregated values and change data types in a SSAS DSV, this is rarely the appropriate place for such processes.

When a data integrity issue is identified either in an EDW table, a MultiD cube, a tabular model, a SSRS report or an Excel workbook, in helps a great deal to have all your business logic located in a single place – your ETL tool.

* + - 1. **Partition your measure groups.** By default, a single measure group will be placed in a single partition. Frequently SSAS partitions are date parameterized and only the current time (month, quarter, or year) is processed each day, and at the end of the time period, the current partition is merged with a historical partition and the cycle continues.

**Note:** Only partitions with the same storage mode and aggregation design can be merged.

* + - 1. **Be aware of your OleDB packet size property.** As with SSIS, you can increase performance when working with large data sets by increasing your OleDB packet size property. Increase the property value from 4096 to 32767 and work with your network administrator to correlate your SSAS connection manager packet size property with your Ethernet card and network switch properties.
      2. **You should always have a time dimension.** If EDW has even one date time column in a fact table, such as invoice, order, shipment or payment date, your EDW should also have a calendar dimension table. Although SSAS has the ability to create a time dimension for you, the appropriate place for this is in your physical and logical data model with data imported during the ETL process and updated via the ETL process when needed. Again, stick to your BI roadmap, even for tables, like DimCalendar, DimTime or DimDate, which are most likely sourced from a one-time imported Excel workbook.
      3. **Have at least one hierarchy per dimension.** MS BI visualization tools, like PPS (Performance Point Services) and Excel, are designed to work with hierarchies. If you have a dimension that has no natural hierarchies, consider denormalizing it into another dimension.
      4. At initial push to production, include only measures and dimension attributes that are clearly needed. Do not misinterpret this best practice with *removing* the unneeded measures and dimensions from the EDW or SSAS DSV. Quite the contrary. If the OLTP took the time to write it to disc, the BI roadmap should probably be taking the time to extract, transform and load the data into the EDW. The semantic layer of SQL Server views should take the time to expose all columns in a table. The SSAS DSV should select all columns. The SSAS dimensions created, however, should only expose a subset of attributes.

Including all attributes in every SSAS dimension in your first production push will result in end users being over-whelmed and frustrated with hundreds of attributes that they are not interested in, and quite possibly do not understand.

A BI project is a process and you want your users to feel a “win” right at the outset. 50% of the success of any BI implementation effort is the attitude of the end users, so “train them up” starting with a manageable, understandable cube which, behind the scenes, is designed to scale quickly to accommodate additional demands.

As your EDW is cleansed and you have not used your DSV for data integration, you only have to perform QA activities against your hierarchies, cube relationships and measure aggregates. Adding an existing EDW column that is in your existing SSAS DSV to an existing SSAS dimension involves very little development effort.

* + - 1. **Use** [**SSAS Dynamic Management Views**](https://msdn.microsoft.com/en-us/library/hh230820.aspx) (DMVs) via MDX to manage SSAS. If you are not comfortable free hand typing your own MDX, you can find several samples on the Internet and make them your own. Here are a few examples:

--find all SSAS DMVs

SELECT \*

FROM $SYSTEM.DBSCHEMA\_TABLES

WHERE TABLE\_SCHEMA = '$SYSTEM'

--find all tables that can be queried

SELECT \*

FROM $SYSTEM.DBSCHEMA\_TABLES

--find current activity

SELECT \*

FROM $SYSTEM.DISCOVER\_SESSIONS

ORDER BY SESSION\_LAST\_COMMAND\_START\_TIME DESC

--get last process date/time for a cube (not the partitions)

SELECT

[CATALOG\_NAME] as DatabaseName

,[CUBE\_NAME] as CubeName

,[LAST\_SCHEMA\_UPDATE] as LastDeployment

--this is the last DateTime that ANY data was processed in the cube

--this is NOT the last DateTime the CUBE was processed

--this timestamp may or may not be the same timestamp that you find in

-- SSAS DB --> Cubes --> CubeName --> Properties --> Last Processed

,[LAST\_DATA\_UPDATE] as LastPartitionProcessed

FROM $system.MDSCHEMA\_CUBES

WHERE LEFT([CUBE\_NAME],1) <> '$'

* + - 1. View your visualization tool as a “data formatter” and **push as much work as possible back onto your cube** and ETL process. Again, we are back talking about the BI roadmap and assigning a function for each software application in the roadmap. The primary purpose of a report writer is to turn data into meaningful information. The primary purpose of a MultiD cube or tabular model is to prepare that data optimized for reporting…
         * end-user friendly labels
         * in-memory secured data
         * transparent table relationships
         * pre aggregated measures
         * shared dimensions
         * business logic calculated
         * hierarchies determined
      2. **When MDX is needed, make sure it is clean, documented and efficient!** Remember that MDX calculations are not pre-processed, so cube calculations should be doing much more than executing a SUM() statement. Reserve MDX for business logic which must exist in a cube environment, but follow the same scripting best practices you use in SSIS and t-SQL:
         * If you find yourself coping and pasting script to reuse it, place that MDX in a function, then call the single function repeatedly.
         * Comment your code!
         * Format your code so it is readable and easily browsed.
      3. Estimate aggregations prior to production, but circle back and **reanalyze the actual aggregation usage** after a short period of time. Spend some time learning to navigate and effectively use the SSAS Aggregation Design Wizard.
      4. **Use BIDS Helper dimension and measure group health checks.**  Before you try to process your MultiD cube, be proactive and run the health checks supplied by BIDS Helper. If your health check reports errors, you can be certain that your cube will not process. BIDS Helper supplies helpful information and points you directly to the attribute or measure values that are causing the problem.
      5. **Tune memory and processor usage** before end users complain. Entire books have been written on this subject, but minimally use the *Rows read/sec* counter in Windows Performance Monitor. Certainly do not ignore long running reports and cube calculations – stay ahead of your end user pool and consciously be proactive.

# SharePoint Integration

The purpose of this section is to provide lessons learned from integration the MS BI tool stack with SharePoint 2013 in a Windows 2012 environment. It is not necessarily “best practices”, but hopefully “tips and tricks” that will save you installation and error handling time.

## Installing SharePoint

1. Check your Windows application log for errors after each installation step.
2. If you are in Win2012, run the prerequisiteinstaller.exe manually. This is supposed to come up in a flash screen, but it does not in Win2012.
3. After the prerequisites are installed, you should be able to successfully run SharePoint’s Setup.exe and install a ‘complete’ (vs. ‘stand-alone’) version of SharePoint to avoid having your SP databases placed in SQL Express.
4. Next, install SharePoint 2013 SP1 (KB2880552)

officeserversp2013-kb2880552-fullfile-x64-en-us.exe

1. It is not recommended that you install SharePoint with local accounts; however, if you are installing on your local VM or in a demo lab with no domain controller (aka “in stand-alone mode”), you may need to do so.

## MS BI Security Spaces

Security in SharePoint, SQL Server, SSRS, SSAS, and effective passing of credentials between all four tools, can be rather complex. Listed below are the security “touch points” for these tools.

|  |  |  |
| --- | --- | --- |
| BI Tool | Security Space | Location |
| SQL Server | Service Accounts | Start 🡪 All Programs 🡪 Microsoft SQL Server 2012 🡪 Configuration Tools 🡪 **SQL Server Configuration Manager** |
| SQL Server | Users / Logins | SSMS 🡪 Object Explorer 🡪 Security 🡪 **Logins**  /and/  SSMS 🡪 Object Explorer 🡪 [Database Name] 🡪 Security 🡪 **Users** |
| SQL Server | Credentials | SSMS 🡪 Object Explorer 🡪 Security 🡪 **Credentials** |
| SQL Server | Proxies | SSMS 🡪 Object Explorer 🡪 SQL Server Agent 🡪 **Proxies** |
| SQL Server | Database Mail Profiles | SSMS 🡪 Object Explorer 🡪 Management 🡪 **Database Mail** |
| SSRS | Shared Data Sources | Visual Studio 🡪 Solution Explorer 🡪 **Shared Data Sources** |
| SSAS | Server Administrators | SSMS 🡪 Analysis Services Tabular Instance 🡪 Properties 🡪 **Security** |
| SSAS | Tabular Model Data Connection | Visual Studio 🡪 Solution Explorer 🡪 Model.bim 🡪 **Existing Connections**  /or/  SSMS 🡪 Analysis Services Tabular Instance 🡪 Databases 🡪 Deployed model name 🡪 **Connections** |
| SSAS | Tabular Model Roles & Membership | SSMS 🡪 Analysis Services Tabular Instance 🡪 Databases 🡪 Deployed model name 🡪 **Roles** |
| SharePoint | Site Users and Groups | Home 🡪 Site Contents 🡪 Settings 🡪 **People and groups** |
| SharePoint | Site permissions | Home 🡪 Site Contents 🡪 Settings 🡪 **Site permissions** |
| SharePoint | Report Data Sources | Home 🡪 Site Contents 🡪 **Data Source Connections** |
| SharePoint | Site Collection Administrators | Central Administration 🡪 Application Management 🡪 **Site Collection Administrators** |
| SharePoint | SSRS E-mail Settings | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 SQL Server Reporting Services Application 🡪 **E-mail Settings** |
| SharePoint | SSRS Execution Account | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 SQL Server Reporting Services Application 🡪 **Execution Account** |
| SharePoint | SSRS Subscriptions & Alerts | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 SQL Server Reporting Services Application 🡪 **Provision Subscriptions & Alerts** |
| SharePoint | External data connections in Excel Calculation Services | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 Excel Services Application Web Service Application 🡪 Global Settings 🡪 **External Data** |
| SharePoint | Secure Store accounts | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 **Secure Store Service Application** |
| SharePoint | PPS Unattended Service Account | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 Performance Point Service Application 🡪 **Performance Point Service Application Settings** |
| SharePoint | Search Service default content access account | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 **Search Service Application** |
| SharePoint | Authentication to DB Server | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 Usage and Health Data Collection Service Application 🡪 **Logging Database Server** |
| SharePoint |  | Central Administration 🡪 Application Management 🡪 Manage service applications 🡪 User Profile Service Application 🡪 **Manage User Permissions** |
| SharePoint / SSRS | Credentials of admin on computer that hosts the report server | Central Administration 🡪 General Application Settings 🡪 **Reporting Services Integration** |
| SharePoint | External Content Types | SharePoint Designer 2013 🡪 [open site] 🡪 External Content Types |
| SharePoint | Data Sources | SharePoint Designer 2013 🡪 [open site] 🡪 Data Sources |

## Integrating SQL Server with SharePoint

1. SharePoint ribbons will be greyed out / disabled and CA menus missing until you disable UAC and restart the server

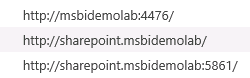
<http://epmgelist.wordpress.com/2013/06/14/central-administration-greyed-out-ribbons-and-missing-functions/>

1. If you do not have a DC (domain controller), you may want to install DNS for Windows 2012

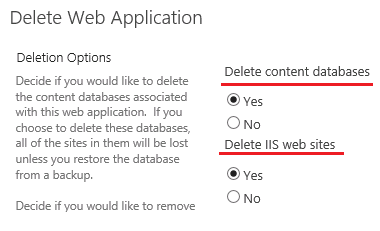
<http://cyrusbesharat.wordpress.com/2013/02/14/how-to-install-dns-on-windows-server-2012/>

Server Manager 🡪 Add Role and Features

Without working with a DNS, you cannot use the ‘Host Header’ property of your SharePoint web applications. Your URLs will be your server name and look like this …

vs. user-friendly names such as …. 

1. When you delete a web application in SharePoint, be sure to clean up everything.



Also, if Internet Explorer will not delete your web applications, try Mozilla.

1. Registering Managed Accounts:
   1. In a production environment, you should use a separate domain account for each SP application for best security.

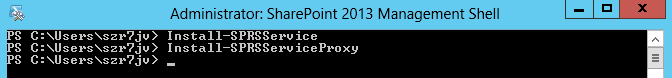
CA 🡪 Security 🡪 General Security 🡪 Configure Managed Account 🡪 Register Managed Account

* 1. Registering managed accounts is pretty straight forward. You can register your service accounts as you go (as you configure Excel, PPS, secure store etc.), or you can choose to get it all done at once.

1. Next, configure your secure store service. Use a smart name for your Secure Store database name. There is no advantage to having GUID-looking database names, and smart names make your SharePoint databases easier to manage.



1. Configure PPS (performance point services), Excel, Visio and SSRS Services. Again, use intelligent database and application pool names.
2. Configure your SSRS service after you create a web app, but before you create a site collection. Detailed instructions are given here with screen prints and this tends to be the place that causes the most difficulty.
   1. Run the PowerShell for SharePoint scripts to install service



Install-SPRSService

Install-SPRSServiceProxy

* 1. Start the service

CA 🡪 System Settings 🡪 Manage Services on Server



* 1. Register managed account from PowerShell (if you are using local accounts)

$cred = Get-Credential

USPNSVULM1057\LocalSPReportingSrv

New-SPManagedAccount –Credential $cred

* 1. Grant access to SharePoint databases from PowerShell

$w = Get-SPWebApplication -identity http://uspnsvulm1057:29460/

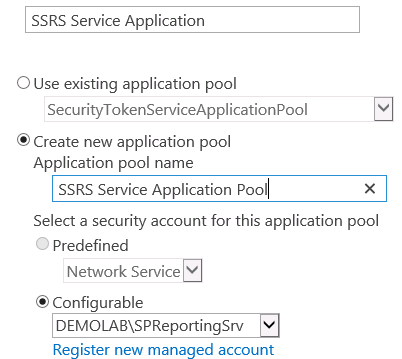
$w.GrantAccessToProcessIdentity("DEMOLAB\SPReportingSrv”)

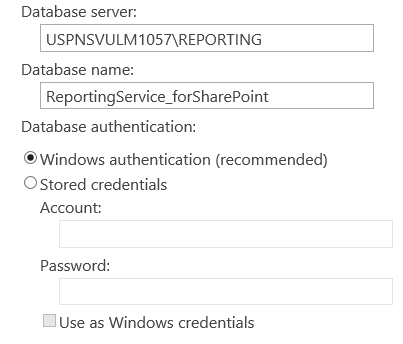
$w = Get-SPWebApplication -identity http://uspnsvulm1057:29460/

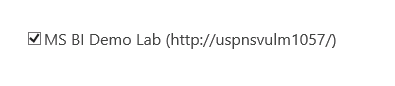
$w.GrantAccessToProcessIdentity("DEMOLAB\SPReportingUser”)

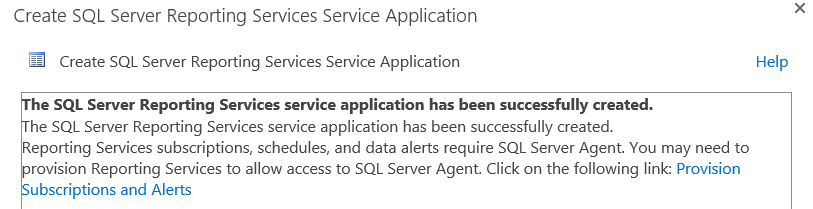
* 1. Create a new SSRS service application

CA 🡪 Application Management 🡪 Manage Service Applications 🡪 New 🡪 SSRS

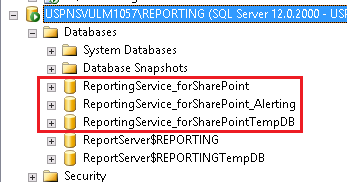








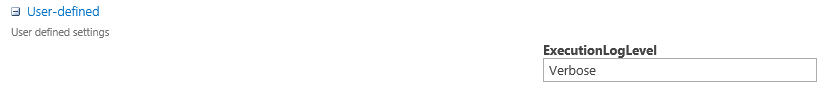
Resulting SQL Server databases 🡪



Note: **ReportServer $REPORTING** databases are for Native Report Server mode.

* 1. Last, change your SharePoint SSRS logging to **verbose**.

CA 🡪 Application Management 🡪 Manage Service Applications 🡪 [Your New SSRS Services Application] 🡪 Service Applications toolbar 🡪 Manage 🡪 System Settings



Note: The default is “normal”.

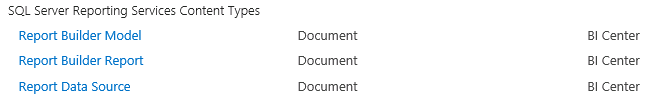
Read more about SSRS logging with SharePoint integration here 🡪 <http://msdn.microsoft.com/en-us/library/ms159110(v=sql.120).aspx#bkmk_sharepoint>

You are now ready to create your first web application! If you have already configured MS BI services, you can connect them at this time. If your service applications have not been created yet, you can easily connect them later by going to CA 🡪 Application Management 🡪 Manage Web Applications 🡪 [highlight your web application] 🡪 Web Applications ribbon 🡪 Service Connections.

## Creating a BI Site Collection

1. Create a Business Intelligence Center site collection. This process is well documented at <http://technet.microsoft.com/en-us/library/jj219656(v=office.15)>.
2. Confirm that SSRS site content is listed in your new site collection

[Your Site] 🡪 Site Contents 🡪 Settings 🡪 **Site Content Types**



1. Confirm that BI features are started in your new site collection. Make sure there is a blue <Active> box next to each of these features:

[Your Site] 🡪 Site Contents 🡪 Settings 🡪 **Site Features**

<http://uspnsvulm1057:40352/sites/BICenter/_layouts/15/ManageFeatures.aspx>

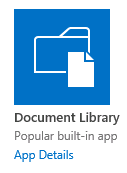
* Access App
* BICenter Data Connections Feature
* PerformancePoint Services Site Features
* Report Server File Sync
* SharePoint Server Enterprise Site Features
* SharePoint Server Standard Site Features

1. Confirm that your BI features are started in your new site collection. Make sure there is a blue <Active> box next to each of these features:

[Your Site] 🡪 Site Contents 🡪 Settings 🡪 **Site Collection Features**

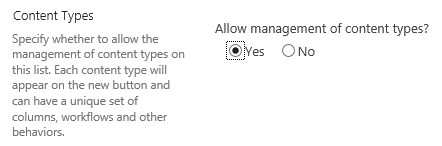
* PerformancePoint Services Site Collection Feature
* Power View Integration feature
* Report Server Integration Feature
* SharePoint Server Enterprise Site Collection Features

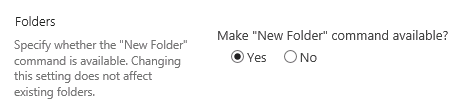
1. For SSRS reports, add a new ‘**Document Library’** application in your site



1. In your new document library, allow management of content types and enable the “New Folder” command

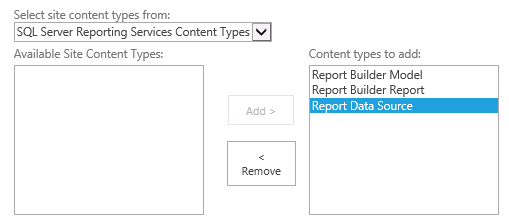
[Name of New Document Library] 🡪 Site Content 🡪 Settings 🡪 Advanced Settings





1. Add SSRS content types to your new document library

[Your new document library 🡪 Library ribbon 🡪 Library Settings 🡪 Add From Existing Site Content Types

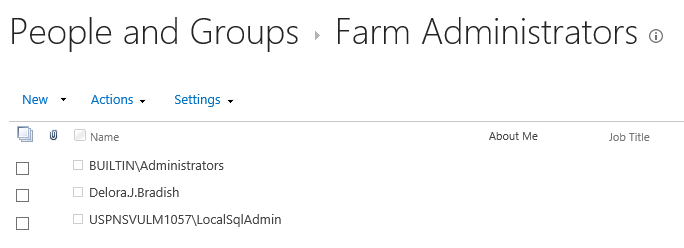


1. Assign User Permissions to the new site collection

Site Settings 🡪 Users and Permissions 🡪 People and groups

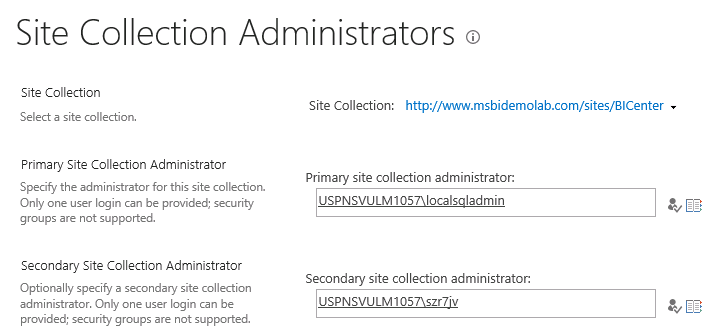
1. Add additional Farm Administrators, if applicable

CA 🡪 Security 🡪 Manage the Farms Administrators Group



1. Update your site collection administrators in

Central Administration 🡪 Application Management 🡪 **Site Collection Administrators**

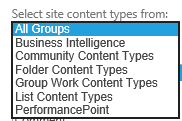


## Verify Your SSRS Configuration

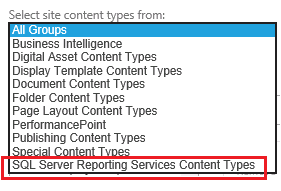
MSDN documents this well here <http://msdn.microsoft.com/en-us/library/ms143773.aspx#bkmk_sharepointmode>.

**CRITICAL NOTE!** You cannot add SSRS content types to “PerformancePoint Content”, or “Custom Lists” objects. You can add SSRS content types to “Document Library”, “Dashboard Library” or “Data Connection Library” objects. So before you think that your SSRS integration did not work, double check the type of SharePoint object with which you are testing!

In this screen shot, the SSRS content types are not available for custom lists.



In this screen shot, the SSRS content types ARE AVAILABLE because you are adding them to a Data Connection Library object.



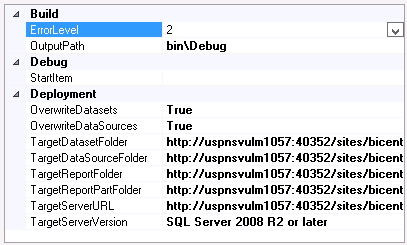
## Deploying SSRS Reports to SharePoint

### SharePoint Prerequisites

1. Business Intelligence-based site collection created
2. SSRS content types added to desired library
3. A data connection successfully added manually to the library
4. Windows application log devoid of SSRS errors

### Visual Studio Prerequisites

1. Reports successfully tested in the VS environment
2. Target URLs set in the project **Property Pages**



|  |  |
| --- | --- |
|  |  |
| Target Dataset Folder | http://uspnsvulm1057:40352/sites/bicenter/Standard Reports/Shared Report Objects/Data Sets |
| Target Data Source Folder | http://uspnsvulm1057:40352/sites/bicenter/Standard Reports/Shared Report Objects/Data Sources |
| Target Report Folder | http://uspnsvulm1057:40352/sites/bicenter/Standard Reports/Adventure Works Reports |
| Target Report Part Folder | http://uspnsvulm1057:40352/sites/bicenter/Standard Reports/Shared Report Objects/Report Parts |
| Target Server URL | http://uspnsvulm1057:40352/sites/bicenter |

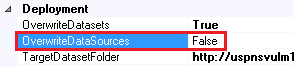
### Deploying reporting objects from Visual Studio

1. First, rebuild your project in Visual Studio and fix any errors.
2. To deploy to SharePoint, it is as simple as a right mouse click and **Deploy**. If your SSRS objects do not deploy…
   1. Check your URLs in your project property. Copy and paste them from your VS error message to test if they work.
   2. Check to make sure you have added SSRS content types to the library to which you are deploying
   3. Check to make sure all of your data connections, shared data sets and reports run without error in VS
   4. If you are *developing* in ONE domain, but trying to *deploy* in a SECOND domain, you will need to coordinate with your IT group to open the appropriate fire walls. Generally, this doesn’t work. You must deploy from inside the domain where your SharePoint site lives.
3. Verify that your data sources, shared data sets and reports are all now in your SharePoint environment.
4. Update your data sources to use **stored credentials**. Unless you are using Kerberos, you will have the well documented “double hop” issue. It is highly recommended that you create a one or more domain accounts for this purpose that have read only permissions to appropriate data sources.
5. Run a SSRS deployed report. If you report does not run…
   1. Check the credentials you are using to hop from SharePoint to the SSRS server and back to SharePoint. Does this account have read (and possibly execute) rights to all views, stored procedures and user defined functions used in the report? Does this account have appropriate access to the SharePoint report databases?

Read about security needed for SSRS SharePoint integration here 🡪 <http://technet.microsoft.com/en-us/library/bb283324(v=sql.105).aspx>

* 1. Can you successfully create a SSRS data source and Report Builder report inside of your SharePoint site library? If not, there is something wrong with your SSRS integration with SharePoint.
  2. Did you redeploy all objects needed by the report like shared data sets or report parts? You are SURE that SharePoint has all the same versions of ALL objects that you are being used successfully in your VS test/dev environment?

1. Back inside of Visual Studio, change your project properties to NOT overwrite your data sources. (You don’t want your SharePoint stored credentials to get over-written.)



## Change Diagnostic Logging

### PowerPivot Service

From <http://msdn.microsoft.com/en-us/library/ee210652.aspx>

“

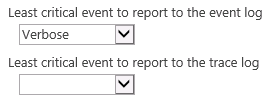
If you want log entries for routine server operations such as connections, requests, or query reporting, you must change the logging level to Verbose.

“

1. Go to CA 🡪 Monitoring 🡪 Configure Diagnostic Logging 🡪
2. Mark the PowerPivot app



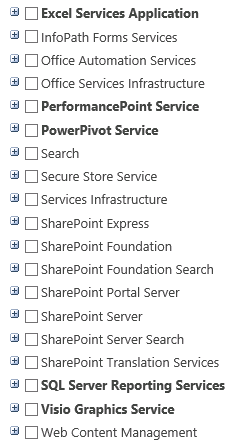
1. Change the “Least critical event to report to the event log” to **Verbose**



1. Click <OK> to save your changes

### Excel, PPS, SSRS & Visio Services

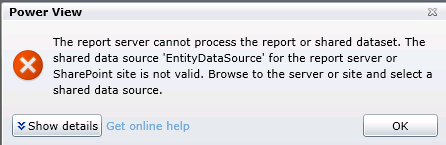
Follow the instructions above to make this same change to additional MS BI services. At least temporarily, having more informative logging is extremely helpful during the initial configuration and testing phase. Reconsider verbose logging after everything is running smoothly.



## Possible ShaerPoint Reporting Errors

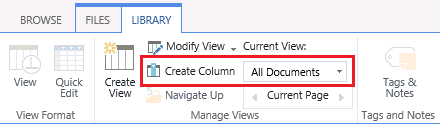
### Shared Data Source is Invalid

Error:

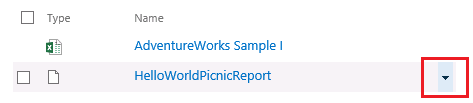


**Solution:** Have you just uploaded an XLSX or RDLX file that contains a data connection from a new source? Verify that the data connection behind the offending document is working.

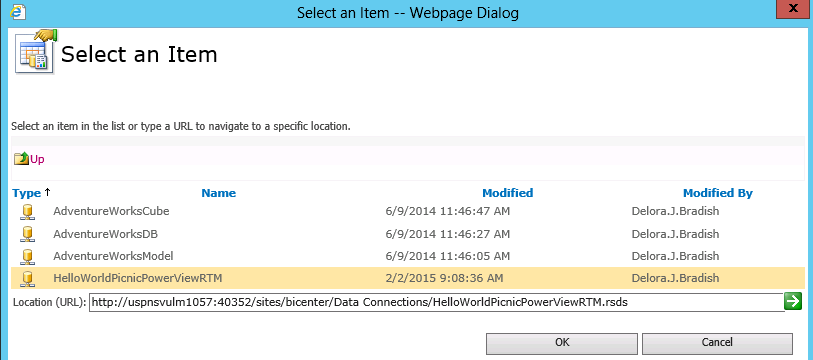
1. Go to your PowerPivot library ribbon and view your PP reports as a list.



1. Now click on the drop down menu of your RDLX file …



1. Go to “Manage Data Sources”
2. Click on the name of data source link and the BISM for PV connection UI should come up.
   1. At this point, if the connection doesn’t even exist in SharePoint, you will need to create it and point to your new “Shared Data Source”, or you can create a “custom data source” right here.
   2. It has happened that a link mysteriously disappears and the Data Source Link box is completely empty. To correct, simply click on the ellipse box and select the existing data connection



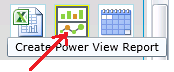
Your Data Source Link should now have a value. Click <OK>.

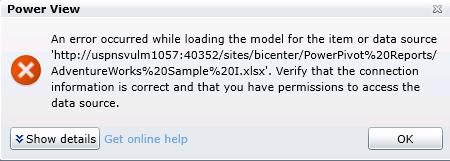


1. In your shared Data Connections, make sure you can successfully test the connection.

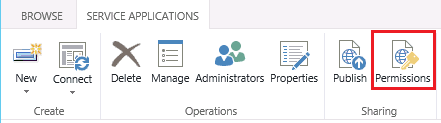
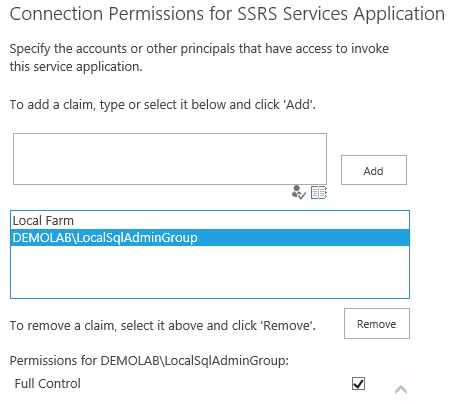
### Permissions to Access the Data Source

Error:



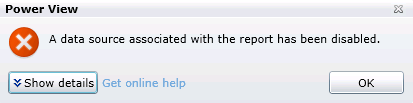


**Solution:** Even with my local domain account in a ‘LocalSqlAdminGroup’ that had ‘sysadmin’ permissions on EVERY SQL Server DB and SSAS instance, you still have to add this same domain group to the **SSRS Service Application** permissions list in CA. This cannot possibly be the best solution(!)?

### Data source has been disabled

**Error:**

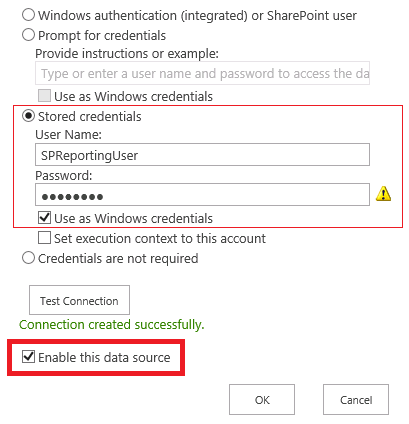


**Solution:**

1. Go to your data source connection and edit the data source definition.

BI Center 🡪 Data Connections 🡪 [Connection Name] 🡪 click ellipse 🡪 click ellipse 🡪 Edit Data Source Definition menu option

1. Re-enter the user Id and password and check “Enable this data source”



1. Test your connection
2. Click <OK> to save your changes and return to your Data Connections list

### BISM for PowerView support of query parameters

To replicate the error, try to create a new BISM for PV against an Excel workbook.

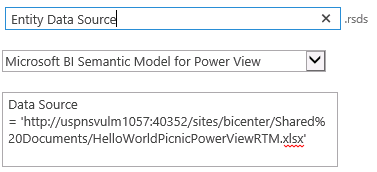
Error Message: The data source type doesn’t support query parameters.



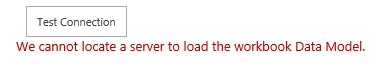
Possible solution: Your URL is wrong. Can you open your URL in IE *without opening Excel*?

### BISM for PowerView cannot locate a server

Replicate the error by following these steps: Create a new ‘Report Data Source’ object in the ‘Data Connection’ library choosing ‘Microsoft BI Semantic Model for Power View’ as your data source type, and an Excel workbook as your source. Click <Test Connection> to receive the error.

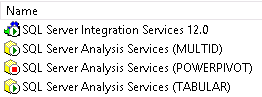


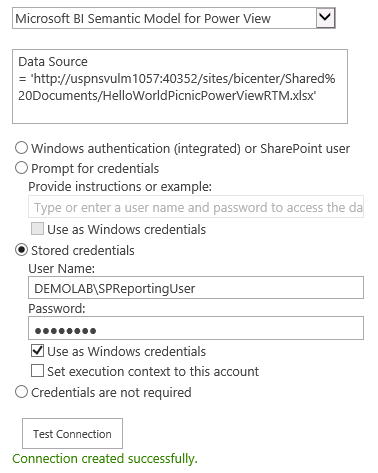
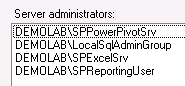
**Error Message:** We cannot locate the server to load the workbook Data Model



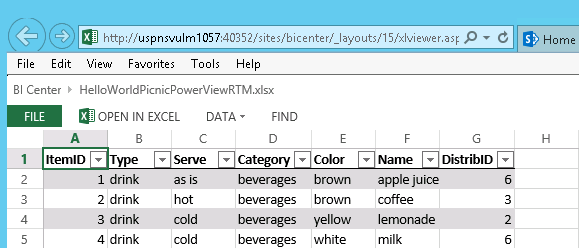
**Solution**:

1. Make sure your SSAS PowerPivot instance is running (!!) and that the stored credential you are using (if applicable) or the local login account (if not using a stored credential), and the SPExcelSrv account all have Administrative permissions in your PowerPivot instance.



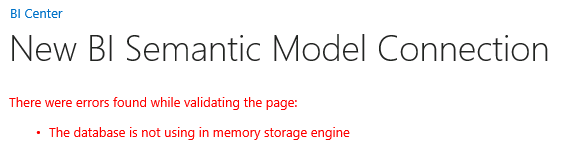


**Note:** Double check your Data Source URL. You should be able to copy/paste the URL to a new IE tab and have your workbook *open inside of IE* – not in the Excel UI.



### BISM In memory storage engine

Error Message: The database is not using in memory storage engine

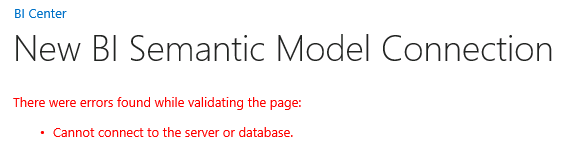


Cause: You are trying to establish a BISM connection to a multidimensional cube

Solution: Connect to a tabular model in an SSAS tabular instance

### BISM Cannot connect

Error Message: Cannot connect to the server or database.



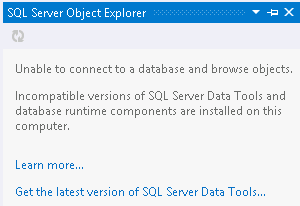
**Solution:** The PowerPivot service domain account must have read access to your tabular model. If you have not set up roles and corresponding memberships for your model, you will need to add the PowerPivot service domain account as an Administrator in your tabular SQL instance.

SSMS 🡪 Connect to your tabular SQL instance 🡪 right mouse click on the instance name 🡪 properties 🡪 security 🡪 Server administrators 🡪 <Add> button



### Visual Studio Incompatible versions

**Error Message:** Incompatible versions of SQL Server Data Tools and database runtime components are installed on this computer.

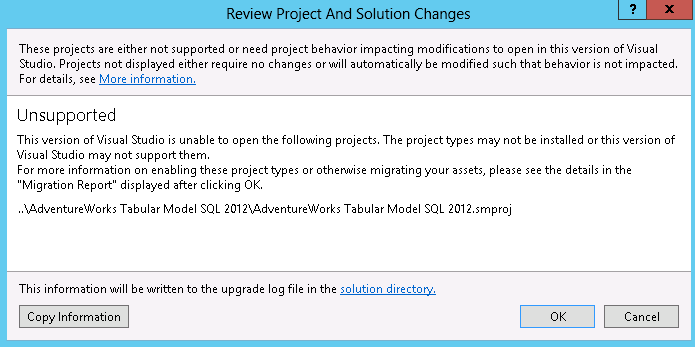


Visual Studio version being used:

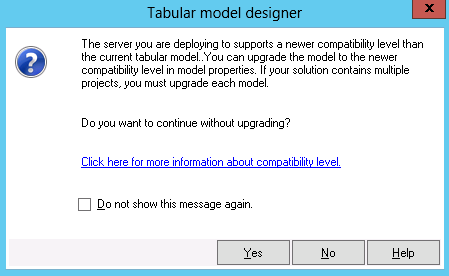


**Solution:**

1. Try to open the solution using Visual Studio 2013, however, Microsoft has not released the BI components for VS2013. You will get this error 🡪



1. Click the <Start> button in VS2012 and you will get this message. Click <Yes>.



# Team Foundation Server

Team Foundation Server (TFS) is the application lifecycle management framework from Microsoft based in Visual Studio. It is a project management tool that can handle version and source control, team collaboration, test case management, and deployment tracking.

1. Clearly indicate what objects should be part of the TFS repository:
   1. Visual Studio solution / project files
   2. Scripts
   3. Documentation
   4. Customer data, if any. (Calendar and time seeding files for date dimensions are often appropriate, but any sort of PII (personally identifiable information) and customer data in general is rarely appropriate.)

In a multi-developer environment, not everything should be checked into TFS, like Visual Studio’s BIN folder and OBJ folders

1. Ensure that all users set a valid local path (aka working folder) at the project root. This is important to keep from “loosing” artifacts stored in multiple places. Your solution architect should be able to tell team member where in the team project file system all TFS working folders should be pointing, or possibly to each local PC.

Remember that even if you map your working folder at the top-most level, each user does not need to “get latest” from the entire project. Team members can still just “get latest” only on the individual project folders they are needing.

1. Train end users in the basic TFS methodology. For source code control, minimally, cover these points:
   1. Connecting to TFS
   2. Entering a TFS server
   3. Setting a TFS working folder
   4. Getting the latest objects from TFS
   5. Checking objects in / out
   6. Undoing pending changes
   7. Reviewing history
2. For TFS Administrators, review these TFS methodologies:
   1. Adding new users to projects
   2. Undo checkout of another team member’s files through the use of TFS Power Tools
   3. Be aware of a possible configuration file fix for VS Users which corrects slowness when connecting to TFS.

There is a problem when connecting to TFS (especially when upgraded from VS2008 to VS2010) that causes extreme slowdown while working in TFS. Some investigation showed that it does not appear to be a TFS issue per se but rather something lower in the .NET Framework stack, having to do with automatic proxy discovery through the WPAD protocol.

You can work around the issue by turning off proxy autodiscovery on a per-application basis by using a .exe.config setting.

The .exe.config for Visual Studio is %VSINSTALLDIR%\Common7\IDE\devenv.exe.config and it already contains a section in the .exe.config for System.Net. After adding the defaultProxy element, that section would look like this:

   1:  <system.net>

   2:     <defaultProxy enabled="false" />

   3:     <settings>

   4:        <ipv6 enabled="true"/>

   5:     </settings>

   6:  </system.net>

This file is a system setting—not a per-user setting. So, it only needs to be updated in 1 place on the server.