cid:image001.jpg@01CA6144.728946B0

***SSIS Processing Standards***

***Microsoft Confidential***

***May 5, 2010***

Table of Contents

[Table of Contents 2](#_Toc260870727)

[Document History 3](#_Toc260870728)

[ETL Process Conventions 3](#_Toc260870729)

[1. General Guidelines for Automated Processes 3](#_Toc260870730)

[2. General Guidelines for SSIS Packages 3](#_Toc260870731)

[3. SSIS Package Naming Conventions 4](#_Toc260870732)

[4. Use of Configuration Files/Configuration Database 4](#_Toc260870733)

[5. Package Logging 5](#_Toc260870734)

[6. Package Deployment 5](#_Toc260870735)

[7. Package Design Guidelines 7](#_Toc260870736)

[8. Package Component Naming Conventions 8](#_Toc260870737)

[9. Operational Considerations: 10](#_Toc260870738)

|  |
| --- |
| Document History |

|  |  |  |
| --- | --- | --- |
| **Last Updated By** | **Last Updated Date** | **Comments** |
| Kelly Carpenter | 1/30/2010 | Draft |
| Kevin Johnston | 2/1/2010 | Draft |
| Dwight Mayeda | 5/6/2010 | Updates to config standards |
|  |  |  |

|  |
| --- |
| ETL Process Conventions |

## General Guidelines for Automated Processes

All automated processing will be performed via the SQL Server job engine.

All ETL processes will be defined and configured as SSIS packages.

## General Guidelines for SSIS Packages

SSIS packages are used for all ETL work.

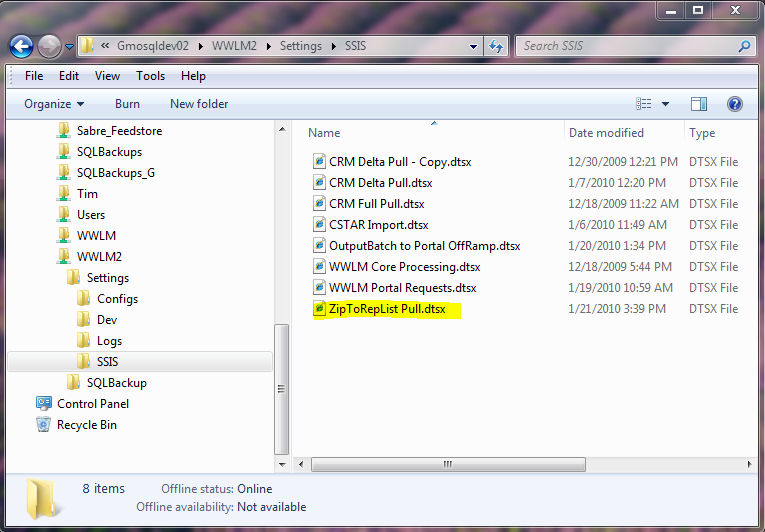
The scope of a particular SSIS package is limited to one subject area. A “subject area” typically refers to the application that a particular ETL set supports. That said, the amount and variety of data needed to support an application can range from simple to complex. An example of a simple application is one which is supported by data on one server and small number databases. A complex application can involve multiple servers, databases, and file systems.

Therefore, a smaller subject area can most likely be served by a single ETL; therefore, a single SSIS package. On the other hand, consideration must be given as to whether a large, complex subject area is better served by breaking the ETL into smaller pieces with corresponding SSIS packages.

Packages should be stored using source control in Visual Studio Team Foundation Server (TFS). TFS source control guidelines and standards are currently being created. Source control for SSIS packages will align with these guidelines.

## SSIS Package Naming Conventions

SSIS ETL package names consist of the target database followed by a description of the ETL process. For example:



In the example highlighted above, “ZipToRepList” is the name of the database, whereas “Pull” describes what the ETL does, in this case, data from the source server for ZipToRep is used as input to refresh a local copy.

## Use of Configuration Files/Configuration Database

SSIS packages should be designed in a way that a package definition (contained in the .dtsx file) can be deployed to any server to support the various application development stages without requiring any code change. In other words, the Testing, UAT, and Production version of the .dtsx code should be identical, even though the Connection Manager properties are different. This is accomplished by storing server, database, and any other components which may be different, in a separate centralized Configuration Database.

As part of standardized deployment of SSIS packages, the Operations Team will start implementing a centralized configuration database. This approach will have two components:

1. An environment variable that will be used as the bootstrap configuration location to point each package to the server where a centralized configuration database is located.
2. A centralized configuration database that will be used by all packages. This database will be standard in all environments.

The environment variable will point to different configuration database depending on which server the package is run: on production servers, it will point to production configuration database, on development servers to development configuration database. The environment variable will contain the OLE DB connection parameter for connecting to the central configuration database.

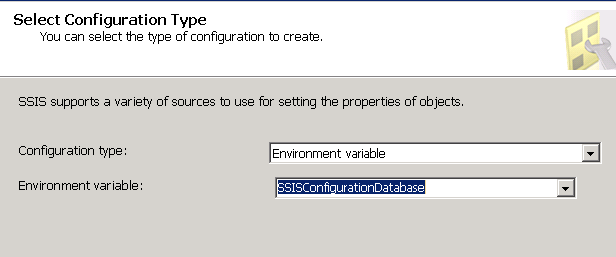
All other configuration elements will be stored in the configuration database.

The configuration database will have the same name and structure across all environments.

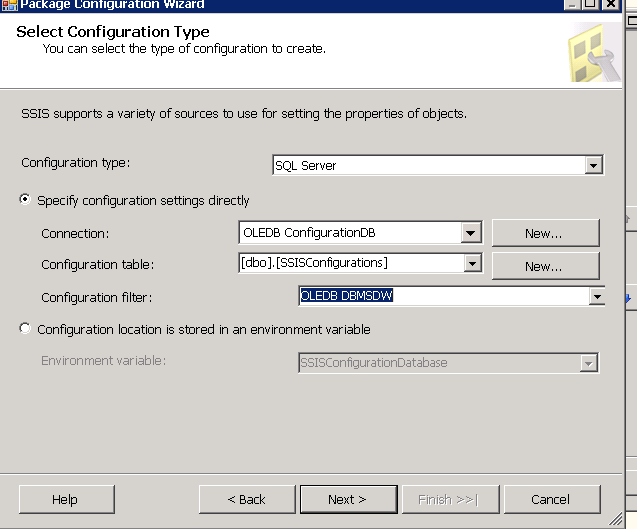
To configure a package two sets of configuration will be added: one based on environment variable and the rest based on centralized SQL Server configuration database.

As part of each deployment, the development team will be asked provide a script to populate the configuration database in production environment in addition to the package.

Validate configuration names against the centralized database before making additions/updates to the database.



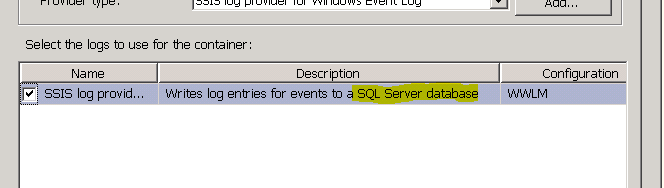
Setting up environment variable configuration



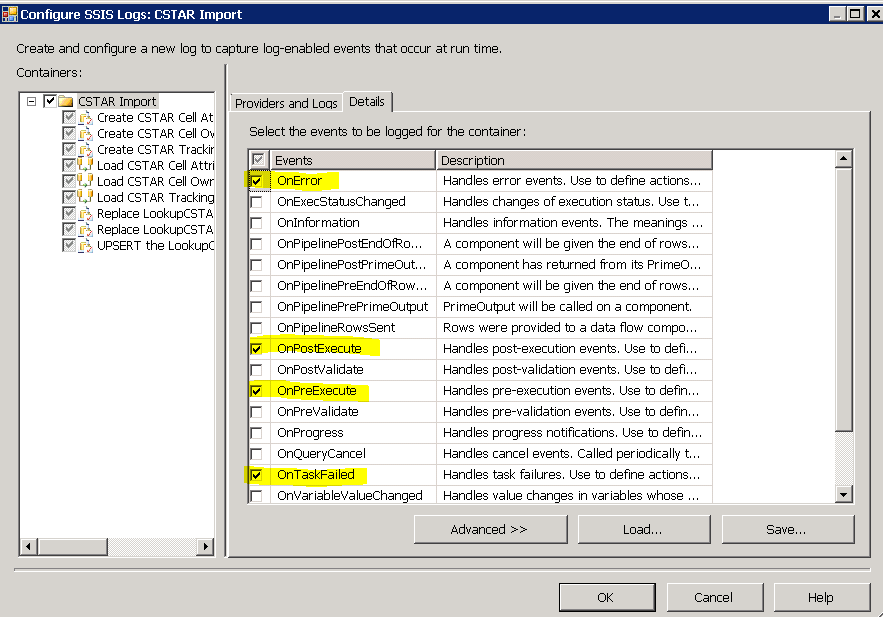
Setting up SQL Server database configuration

## Package Logging

All SSIS packages should have logging enabled. Logging should be configured to point to the same centralized SQL Server used for package configuration.

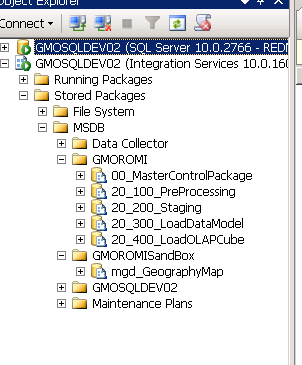


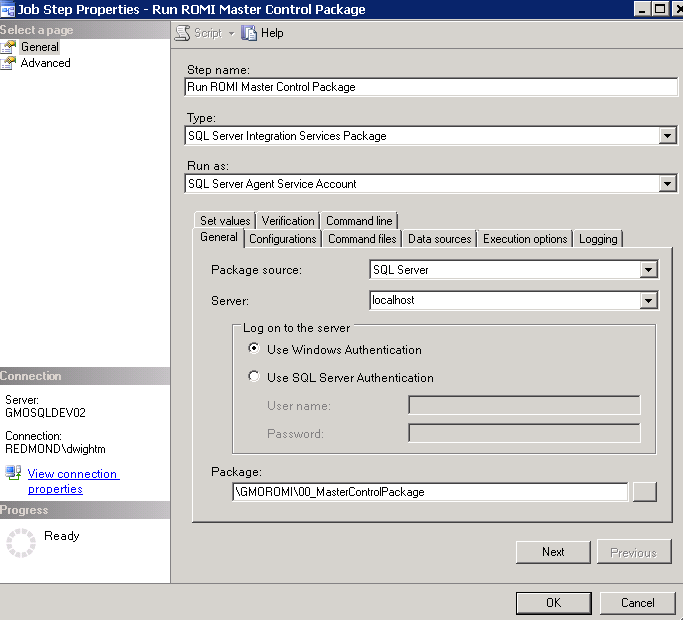
When packages are deployed to Production, the “OnError”, “OnPostExecute”, “OnPreExecute” and “OnTaskFailure”, events should be logged.



## Package Deployment

All packages will be stored in SQL Server DB when deployed in production. This will allow the backup of the packages with MSDB backups.





## Package Design Guidelines

Smaller, simpler packages are preferable to larger, complex ones.

Tasks that can be run in parallel should be placed in separate sequence containers.

Data transformations that can be run in parallel should be placed in the same Data Flow task.

All tasks should contain proper error handling and post to a logging mechanism.

Execute SQL tasks should not contain a large amount of code. Instead the code should be placed in stored procedures and executed from the Execute SQL task. Consequently, the code for all dependent stored procedures should be included in the same Visual Studio solution as the SSIS package.

Processes which perform ETL on tables in the same database are better served by stored procedures and executed by Execute SQL tasks.

Tasks involving multiple databases and servers should be defined in Data Flow tasks, employing Connection Manager Definitions for each database object involved, the specifics of which are prescribed in the Configuration database.

## Package Component Naming Conventions

This section prescribes best practices for naming the various objects in a package, such as sequence containers, execute SQL tasks, data flow tasks, the objects within a data flow task, such as source, destination, and transformation tasks, etc.

Set of recommended verbs:

|  |  |
| --- | --- |
| **Verb** | **Description** |
| Add or Insert | Insert a specific row. |
| Delete | Delete a specific row or set of rows. (Usually based on a column for the table). |
| Get | Select and return a specific row or a set of related rows. (Usually based on a column for the table). |
| List | Select and return all rows for a table. |
| Update | Update a specific row or set of rows. UPDATE ONLY processing is recommended for Set stored procedures. |
| Validate | Perform validation of rows to be operated |
| Log | Log changes (update, insert, delete) to a specific row or a set of related rows. |

Control Flow Objectscomponent

|  |  |  |
| --- | --- | --- |
| **Type** | **Control Flow Object** |  |
| Container |  |  |
| Container |  |  |
| Container |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |
| Task |  |  |

Data Flow C

|  |  |  |
| --- | --- | --- |
| **Type** | **Data Flow Component** |  |
| Data Flow Source | ADO NET Source | ADO\_SRC |
| Data Flow Source |  |  |
| Data Flow Source |  |  |
| Data Flow Source |  |  |
| Data Flow Source |  |  |
| Data Flow Source |  |  |
| Data Flow Source |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation | Cache Transform | CTR |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation |  |  |
| Data Flow Transformation | ADO NET Destination | ADO\_DST |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |
| Data Flow Destination |  |  |

## Operational Considerations:

1. ETL process grows in complexity very quickly.  Since package won’t be run from BIDS in production, the Operations Team needs visibility to help:
   1. Understand what went wrong when a package didn’t work as expected
   2. Monitor the performance of the package(s) to forecast performance to stay within a given timeframe
2. Packages should be kept as simple as possible. Rather than one big package, have smaller packages that can be run independently of each other. <If multiple packages need to share certain values, when a child package is executed from a master package, the parameters that are passed from the master package need to be configured in the child package. For this, use the ‘Parent Package Configuration’ option in the child package. But, for using the ‘Parent Package Configuration’, you need to specify the name of the ‘Parent Package Variable’ that is passed to the child package. If you want to call the same child package multiple times (each time with a different parameter value), declare the parent package variables (with the same name as given in the child package) with a scope limited to ‘Execute Package Tasks’. SSIS allows declaring variables with the same name but the scope limited to different tasks – all inside the same package!>
3. Consider the fact that keeping SSIS packages small and atomic will allow operations team to easily configure them production using SQL Agent jobs in multiple steps. This will provide easy rollback in case of a failure and logs will not grow too big.
4. One of the challenges of moving packages from development to production is configuration. To ease this process and minimize errors, all packages should utilize the centralized Configuration Database and do initial boot strapping of the Configuration Database using environment variables. These environment variable would be specific to each environment (production, UAT, development), and would point to the configuration database where the actual connection information would be held.
5. Logging will be configured for “OnError”, “OnPostExecute”, “OnPreExecute” and “OnTaskFailure” during deployment. Developers should add additional logging options if the package/tasks warrant it (e.g. Script Tasks). Please confer with the Operations Team before substantially increasing the amount of logging in a package.

As a side note, the Operations Team will be testing a different DtExec that provides more logging options.

1. All packages will be stored in SQL Server DB when deployed in production. This will allow the backup of the packages with MSDB backups.
2. Leverage “Transaction” and “Checkpoint” features for easy restartability and minimized data load repetitions. You can do the same across multiple tasks by using “RetainSameConnection” parameter set to true.
3. Leverage “FailParentOnFailure” to stop or continue the package execution.
4. Make sure Run64BitRuntime is set to true.
5. Leverage new caching methods in SSIS 2008 to minimize hits on the database server.
6. Always use DTExec for performance testing, do not rely on BIDS.
7. Do not use “SELECT \*” and pull all the fields from a table/view for Data Flow Sources. Use specific column names to efficiently use memory and IO.
8. Do not use the simple table name from the pull down menu for Data Flow Sources. Use an explicit “SELECT” statement; it is has better performance.
9. Do not sort unnecessarily. Always set IsSorted=TRUE on the Source component output (Advanced Editor) if you know the source data is sorted.
10. Parallel runs for data flows should be limited to 2 or 3 if it’s going to be run in a shared environment. If it’s an exclusive environment then it should be 1 less than CPU\*core numbers.
11. Make sure to drop indexes before large data loads and rebuild them afterwards.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |