

## Scale-Out SQL Server Integration Services Environment:

### A Sample Pattern

SQL Server Technical Article

**Summary:** This white paper illustrates how Microsoft SQL Server Integration Services (SSIS) can be used in conjunction with commodity hardware or virtual machines running in Microsoft Azure to build elastic, highly available, and highly scalable ETL solutions.

**Authors:** Joe Quint, Steve Howard, Patrick LeBlanc, Sreedhar Pelluru

**Technical Reviewers:** Carla Sabotta, Wee Hyong Tok

**Published:** TBD

**Applies to:** SQL Server 2014 SSIS, SQL Server 2012 SSIS

Copyright

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious.  No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2014 Microsoft. All rights reserved.

Contents

[Scale-Out SQL Server Integration Services Environment: 1](#_Toc403369936)

[A Sample Pattern 1](#_Toc403369937)

[Introduction 5](#_Toc403369938)

[Scale-out architecture and methodology 5](#_Toc403369939)

[Supervisor 8](#_Toc403369940)

[Worker nodes 8](#_Toc403369941)

[SSIS packages on worker nodes 8](#_Toc403369942)

[Determine workload granularity 10](#_Toc403369943)

[Sample implementation 10](#_Toc403369944)

[Configure a scale-out SSIS environment 11](#_Toc403369945)

[Restore ETLSupervisor database on supervisor machine 11](#_Toc403369946)

[Review projects in ScaleOutSSIS Visual Studio solution 13](#_Toc403369947)

[Review SSIS packages in ETLSolution project 14](#_Toc403369948)

[Deploy ETLSolution project to worker nodes 15](#_Toc403369949)

[Schedule SSIS packages on worker nodes 16](#_Toc403369950)

[Review SSIS package in CreateWorkflows project 20](#_Toc403369951)

[Deploy CreateWorkflows project to supervisor machine 21](#_Toc403369952)

[Schedule Create Test Workflow package on supervisor machine 21](#_Toc403369953)

[Enable SQL Server Agent jobs on supervisor and worker nodes 21](#_Toc403369954)

[Monitor workflows 22](#_Toc403369955)

[Deploy ReportingSolution project 23](#_Toc403369956)

[Work Flow Monitor Dashboard 23](#_Toc403369957)

[Maintenance 24](#_Toc403369958)

[Add and remove processing servers 24](#_Toc403369959)

[Hung jobs and timeout 25](#_Toc403369960)

[Archive work items 25](#_Toc403369961)

[Alternate implementations 25](#_Toc403369962)

[Appendix 27](#_Toc403369963)

[Stored procedures 27](#_Toc403369964)

[Runtime stored procedures 27](#_Toc403369965)

[Administrative stored procedures 30](#_Toc403369966)

[Reporting stored procedures 32](#_Toc403369967)

[Uncategorized stored procedures 33](#_Toc403369968)

[Table-valued functions 33](#_Toc403369969)

[rptGetCurrentWorkflowStatus 33](#_Toc403369970)

[rptGetWorkflowDuration 33](#_Toc403369971)

[Tables 33](#_Toc403369972)

[ETLNodes 33](#_Toc403369973)

[WorkItemStatus 33](#_Toc403369974)

[WorkItemTypes 34](#_Toc403369975)

[WorkItems 35](#_Toc403369976)

[WorkItemLocations 35](#_Toc403369977)

[Views 35](#_Toc403369978)

[WorkItemsMonitor view 35](#_Toc403369979)

# Introduction

Scalability refers to the ability of an application to continue to meet its performance objectives with increased load. As the load increases on the system, your Microsoft® SQL Server® Integration Services (SSIS)-based Extract, Transform, and Load (ETL) system may not satisfy the performance requirements of your organization, so you may need to scale your ETL system to handle the increased workload at an acceptable performance level. There are two main approaches to scaling: scale-up and scale-out.

**Scale-up approach:** You upgrade your existing hardware (such as CPU, memory, disks, and network adapters). An upgrade could also entail replacing existing servers with new servers. To support the scale-up approach, you also need to increase the amount of parallelism in your ETL system to increase overall throughput. SSIS has a number of built-in features that make it easy to design your packages for parallel processing. For more information, see [Designing your SSIS Packages for Parallelism](http://technet.microsoft.com/sqlserver/ff686759.aspx).

Scaling up is a simple option, and it can be cost effective because it does not introduce additional maintenance and support costs. However, you may want to consider the scale-out approach for reasons such as:

* Any single point of failure remains with the scale-up approach, which is a risk.
* Beyond a certain threshold, adding more hardware to the existing servers may not yield additional throughput or better performance.
* Parallelism is limited to what has been hardcoded into the package. For a more dynamic, scalable approach, parallelism needs to be introduced and coordinated at a higher level.
* When running in environments such as virtual machines running in Azure, scaling up may not be an option. In this case, scaling out to more virtual machines is the only option for improving throughput.

**Scale-out approach:** You add more servers to your system to spread the application processing load across multiple computers. Doing so increases the overall processing capacity of the system. You can take a server offline for maintenance with relatively little impact on the ETL system.

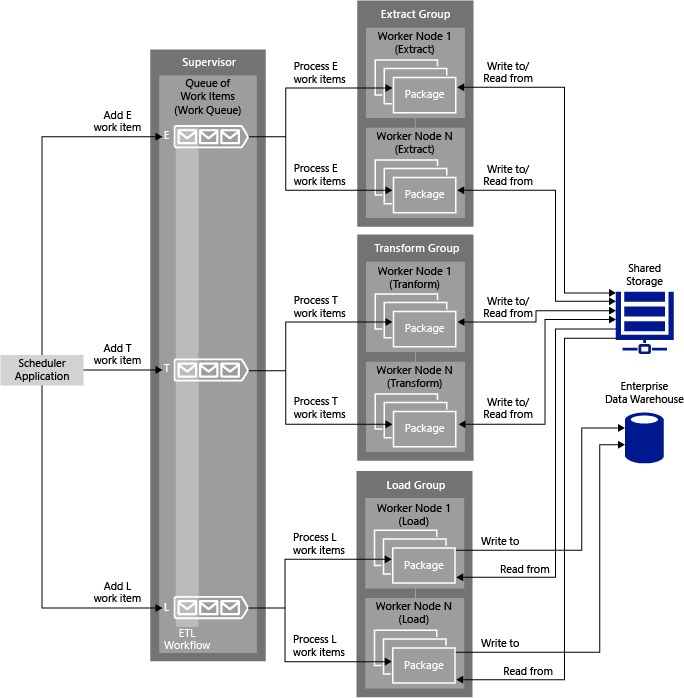
This whitepaper describes a proven architecture and methodology used to effectively scale-out SSIS processing across an arbitrary number of worker nodes. It provides a sample implementation of an architectural pattern in addition to supporting files that you can download to use for testing.

# Scale-out architecture and methodology

The core component of this scale-out architecture is a *supervisor* machine that maintains a centralized*work pile* or *work queue*. A work queue contains *work items*. An ETL farm with two or more *worker nodes* process work items in the work queue by using SSIS packages.

A *work flow* is a logical group of related work items in the work queue. It represents an ETL task, and it is divided into distinct extract, transform, and load operations so that these operations can run in parallel across servers.

The following diagram depicts the scale-out architecture with all the components in it.



This diagram shows the following work flow:

1. The scheduler application adds a work flow to the work queue on the supervisor machine. A work flow contains three types of work items: Extract, Transform, and Load.

The work queue is implemented as a table in a SQL Server database. A row in the table corresponds to a work item in the workflow. When a workflow is created, three rows are created in the table, one for each work item type.

1. There are three groups of worker nodes, one for each type of work item (Extract, Transform, and Load). There can be one or more worker nodes in each group.

A SQL Server Agent job on a worker node runs an SSIS package every minute or as per the schedule defined by the job. The SSIS packages process appropriate types of work items. For example, packages running on a worker node in the Extract group process Extract work items.

1. The SSIS package running on a worker node in the **Extract** group checks out a work item of the Extract type from the queue, processes the work item, and then checks in the work item to the database with the processing status.

To process the work item, the package reads data from a flat file from an input folder, transforms the data, and then writes the output data to a file in the shared storage.

1. The SSIS package running a worker node in the **Transform** group checks out a work item of the Transform type from the queue, processes the work item, and then checks in the work item to the database with the processing status.

To process the work item, the package reads the output file that was stored in shared storage by the Extract package in Step 3, transforms the data, and then writes the output data to a file in the shared storage.

1. The SSIS package running on a worker node in the **Load** group checks out a work item of the Load type from the queue, processes the work item, and then checks in the work item to the database with the processing status.

To process the work item, the package reads the output file that was stored in shared storage by the Transform package in Step 5, transforms the data, and then writes output data to a file in the destination data store, typically a data warehouse.

When multiple workflows are being processed, the work items in different workflows may be at different stages of processing at a given time. For example, a workflow may be at the load (L) stage of processing while another workflow is at the extract (E) stage. However, within a workflow, work items are processed in the sequence you define, for example, E, T, and then L. You may have tens, hundreds, or thousands of workflows running in a production environment.

In the sample implementation provided with this article, a workflow has three work items: one for extract, one for transform, and one for load. Each work item (E, T, or L) is processed by an instance of a package that is dedicated to processing work items of this specific type. Whether processing large amounts of data or small sets of files from multiple sources, this solution works for any data integration issue that can be broken into an isolated set of work items.

## Supervisor

The supervisor runs the whole show. If the supervisor is down, no work gets done because the work cannot be coordinated. For an analogy, the supervisor can be compared to a superintendent at a construction site. The superintendent has a clipboard of activities that are similar to work items in the scale-out architecture. The clipboard represents the supervisor schema, and the construction workers represent the SSIS packages running on worker nodes.

The construction workers form a single-file line (because anything else would be mayhem), and one-by-one, they check out jobs. After a construction worker has completed the job, he returns to the line and checks-in the job. If the job has been outstanding for too long, the supervisor will reassign the job to a new construction worker.



Supervisor

Server 3

Package A

Server 2

Package B

Server 4

Package B

Server 1

Package B

Server 1

Package A

Coffee Break 3 minutes

At this point, one could make an argument that the single threaded nature of checking in and out jobs could cause a bottleneck within the system. Although this statement is true, the time to check in and check out a job should be less than 50 milliseconds, and the time for a job to complete should be in the minutes. Therefore, the percentage of time in management overhead is miniscule compared to the overall processing.

## Worker nodes

Each worker node in the architecture requires an installation of SQL Server Integration Services (SSIS) and a SQL Server instance. Each worker node may contain an identical set of SSIS packages and SQL Server Agent jobs. The SQL Server Agent jobs running on worker nodes start SSIS packages every minute or as per the schedule defined on the jobs. These packages poll the supervisor for available work items.

### SSIS packages on worker nodes

Each worker node in the architecture should be able process any type of work item irrespective of the group (Extract, Transform, or Load) to which it belongs. Therefore, the same set of packages should be deployed across all nodes. This gives you the flexibility to switch worker nodes between the groups based on the performance of each group.

A SQL Server Agent job running on a worker node invokes the main SSIS package, which in turn may optionally invoke one or more child packages. The main task of the package is to find the next available work item in the queue and process the work item by itself or by invoking a child package. The package is responsible for updating the work item’s status on the supervisor node after processing the work item.

For robustness, the package may also contain its own timeout logic. If the package cannot process a work item before the timeout expires, the package should check whether it still owns the work item. If the work item has been picked up by another node because the timeout has expired, the package should discard the results and not update the status on the supervisor node.

#### Package design principles

Packages in this solution must be able to run on any node. The consecutive phases (for example, E >T >L) within a workflow might not run on the same node. Packages performing different stages of processing on a work item may need to share information or resources.

Packages must be designed to not modify shared resources until they have completed their processing. Packages should also check if they still own the work item before committing any changes.

#### Staged data

Some forms of ETL processing may require data to be staged between phases. Because the consecutive work item phases may be performed on different nodes, the data needs to be staged in a shared resource.

There are three common ways to stage data with SSIS:

|  |  |  |
| --- | --- | --- |
| **Resource** | **Method** | **Advantages** |
| Central storage | Flat file | * Readable by humans * Readable outside of SSIS |
| Raw file | * Slight performance advantage over flat file * Embedded metadata * Smaller file sizes |
| Azure blob storage | * Best option if you are using virtual machines running in Azure to set up the scale-out environment. |
| Shared SQL Server | SQL Server tables | * Metadata * Can be queried * Easy to divide data into shared tables by using batch ID or time stamp columns |

Staging data between data flows is a common design pattern. If a lot of data must be staged, the time it takes to stage on a shared resource may outweigh the benefit of scaling out the processing. You might want to consider staging the data locally and performing all processing within a single package or phase instead.

## Determine workload granularity

A key aspect for success in any parallel design solution is choosing the right granularity of work. Unfortunately, there is no one right answer for this issue. The goal is to have the system equally busy across all nodes (a skew with a difference of 0%).

The skew is calculated by using the following equation:

|  |
| --- |
|  |

Figure 2: Skew Comparison shows that a skew of more than 10 denotes an imbalance of work. The larger the difference, the larger the issue, which means less efficiency for the hardware that is being used.

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

Figure 2: Skew comparison

On the other extreme, a too small unit of work will increase the overhead of the system. A method to gauge this overhead is to compare the setup time required for the task with the time it takes to perform the task. For example, within an SSIS dataflow, the system will pre-cache data for a lookup transformation. If the time required for the pre-cache is equal to or greater than the time spent transforming the data, the unit of work is too small or the pre-caching strategy is flawed.

# Sample implementation

Let us look at one way of implementing the scale-out methodology described in this whitepaper by using a sample.

First, download the zip file for this sample from the [Microsoft Download Center](http://go.microsoft.com/fwlink/?LinkId=518250). This zip file (ScaleOutSSISSolutions.zip) contains the following files:

* Database backup (.bak) file for the database: ETLSupervisor
* A Visual Studio 2013 solution that consists of two SSIS projects and an SSRS project: [SQL Server Data Tools – Business Intelligence for Visual Studio 2013](http://www.microsoft.com/en-us/download/details.aspx?id=42313)

Extract files from the package to your computer and copy the ETLSupervisor.bak file to the supervisor machine as follows, depending on your version of SQL Server:

* SQL Server 2014:   
  C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\Backup
* SQL Server 2012:   
  C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\Backup

Next, we will look at step-by-step instructions for configuring a scale-out SSIS environment, and then we will see how to start workflows and monitor them.

## Configure a scale-out SSIS environment

To configure the scale-out SSIS environment for this sample, you need to perform the following steps:

1. **Identify machines to be used as supervisor and worker nodes**. Even though you could have a single machine to act as both a supervisor and work node, we recommend that you test the sample with at least three machines (one supervisor machine and two worker nodes).
2. **On all the machines**:
   1. Install SQL Server 2014 or SQL Server 2012 with Integration Services.
   2. Create the SSIS Catalog. For step-by step instructions, see [Create the SSIS Catalog](http://msdn.microsoft.com/en-us/library/gg471509.aspx).
3. **On the supervisor machine:** 
   1. Restore the ETLSupervsior database (see the [Restore ETLSupervisor database on supervisor machine](#RestoreETLSupervisor) section).
   2. Deploy the CreateWorkflows SSIS project**.** The project includes the Create Test Workflow package, which creates an ETL workflow when it runs.
   3. Schedule a SQL Server Agent job to run the Create Test Workflow package every minute. This automates the creation of workflows for testing purposes.
4. **On worker node machines:** 
   1. Deploy the ETLSolution project (see the [Deploy ETLSolution project to worker nodes](#DeployWorkerNodes) section). This project contains packages that process work items from the work queue in the supervisor machine.
   2. Schedule three SQL Server Agent jobs (one for each SSIS package in the ETLSolution project) to run every minute.

### Restore ETLSupervisor database on supervisor machine

The first step in configuring your SSIS scale-out environment is to restore the ETLSupervisor database. This database stores all the information needed to drive the process of scaling out your SSIS deployment. In addition, it stores all the functionality required to view the SSRS-based reports that are included in this sample.

To restore the ETLSupervisor database:

1. On the supervisor machine, launch **SQL Server Management Studio**.
2. Expand **<server name>,** right-click **Databases**, select **Restore Database**. You should see the **Restore Database** dialog box.
3. In the **Restore Database** dialog box, select **Device** for **Source**, and then click **…** (ellipsis).
4. In the **Select backup devices** dialog box, click **Add**.
5. Navigate to the folder where the **ETLSupervisor.bak** file is saved, select the file, and click **OK**.
6. Click **OK** to close the **Select backup devices** dialog box.
7. Click **OK** to close the **Restore Database** dialog box.
8. Confirm that you see the **ETLSupervisor** database under **Databases**.

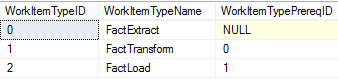
The most important tables in this database are: **WorkItemTypes**, **WorkFlows**, and **WorkItems**. As the name suggests, the WorkFlows table contains information about workflows that ran or are currently running in the ETL system. The WorkItems table contains work items within these workflows that ran or are currently running in the ETL system. There will be three work items for each workflow. The WorkItems table implements the work queue concept in the scale-out pattern that is described in the [Scale-out Architecture and Methodology](#_Scale-Out_Architecture_and) section.

Let’s review the WorkItemTypes table first. We will look at WorkFlows and WorkItems tables later. Please see the [Appendix](#_Appendix) for detailed information for all the tables, stored procedures, functions, and views in the ETLSupervisor database.

#### Review WorkItemTypes table

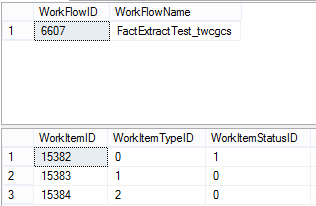
This table contains the types of work items processed by the ETL system. It also defines a workflow by specifying the order in which work items of these types are processed.

In the following example (partial values from the WorkItemTypes table in the ETLSupervisor database), there are three work item types defined: FactExtract, FactTransform, and FactLoad. FactExtract does not have a prerequisite work item type, FactTransform has a prerequisite set to 0 (FactExtract), and FactLoad has a prerequisite set to 1 (FactTransform). This means that within a workflow, a FactExtract work item is run first, a FactTansform work item is run next, followed by a FactLoad work item.



When you invoke the CreateWorkItem stored procedure with a workflow name, and a work item type, the procedure creates a workflow and three work items (one for each work item type). The workflow may correspond to an input file to be processed by an ETL system.

The following example shows the WorkFlows table and the WorkItems table. The WorkItems table also includes the WorkFlowID column to indicate which workflow each work item is associated with.



You can easily define a different type of workflow, for example, ELT. To set up an ELT workflow, you set the prerequisite for the FactLoad work item type (WorkItemTypePreReq column) to FactExtract, and you set the prerequisite for the FactTransform work item type to FactLoad.

Another example would be to load dimensions and facts in a specific order. See [WorkItemTypes](#_WorkItemTypes) in the Appendix for more details.

### Review projects in ScaleOutSSIS Visual Studio solution

Open the Visual Studio (.sln) file from the ScaleOutSSIS folder in [SQL Server Data Tools – Business Intelligence](http://www.microsoft.com/en-us/download/details.aspx?id=42313) for Visual Studio 2013. You will see three projects in the file:

* **CreateWorkflows** project: This SSIS project contains an SSIS package that can be used to generate workflows for testing the ETL solution.

|  |  |
| --- | --- |
| **Package** | **Description** |
| Create Test Work Flow | Creates a work flow that provides three work items of the FactExtract, FactTransform, and FactLoad types |

* **ETLSolution** project: This SSIS project contains three SSIS packages (one for each work item type in the workflow).

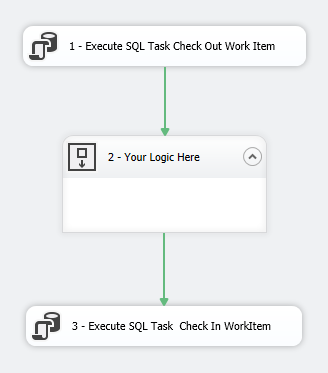
|  |  |
| --- | --- |
| **Package** | **Description** |
| Process Extract Work Item | Processes work items of the FactExtract type |
| Process Transform Work Item | Processes work items of the FactTransform type |
| Process Load Work Item | Processes work items of the FactLoad type |

* **ReportingSolution** project: This SQL Server Reporting Services (SSRS) project contains a report and a dashboard to monitor all the workflows in the ETL system.

Let’s review the three packages in the ETLSolution. We will review the package in the CreateWorkflows project later.

### Review SSIS packages in ETLSolution project

Open The Process Extract Work Item, Process Transform Work Item, and Process Load Work Item packages in Visual Studio. You will notice that the first and last tasks in these packages are similar, as shown in the following image:



The [Execute SQL Task](http://msdn.microsoft.com/en-us/library/ms141003(v=sql.110).aspx) Check Out Work Item checks out a work item for processing by running the CheckOutWorkItemByType stored procedure.

This stored procedure has two required parameters: WorkItemTypeID and ETLNodeName. If you look at the parameter mapping for the Check Out Work Item task, you will notice that these parameters are mapped to two variables:

* User::ETLServerName
* User::WorkItemTypeID

The ETLServerName variable uses the expression @[System::MachineName] to get the name of the machine on which the SSIS package runs.

The WorkItemTypeID is determined by the type of work item that the package will be processing. The WorkItemTypeID variable is set to 0 (FactExtract) for the Process Extract Work Item package, 1 (FactTransform) for the Process Transform Work Item package, and 2 (FactLoad) for the Process Load Work Item package. See [CheckOutWorkItemByType](#_CheckOutWorkItemByType) in the Appendix to understand what this stored procedure exactly does.

The stored procedure returns a list of values. In all of the processing packages, a [Precedence Constraint](http://msdn.microsoft.com/en-us/library/ms141261(v=sql.110).aspx) uses the **ReadyToRun** value from the list to determine the order of the how the final tasks will run in the Data Flow. Double-click the arrow that connects **Execute SQL Task Check Out Work Item** and **WaitFor10to20 Seconds**. You’ll see the expression: **@[User::ReadyToRun] == 1**. When **ReadyToRun** is 1 (run), the subsequent tasks run; otherwise, the package terminates.

In the sample, the Check In Work Item task in all the processing packages invokes a stored procedure that waits for 10-20 seconds (by using a WaitFor Delay statement).

The Check In Work Item task in the processing packages checks in the work item by using the **CheckInWorkItem** stored procedure. This stored procedure will set the status of the work item in the database, based on processing status.

The CheckInWorkItem stored procedure requires three parameters: WorkItemID, ETLNodeName, and Success (status). The sample packages set the work item ID to the ID it received from the CheckOutWorkItemByType stored procedure. The packages set the status to 1 (success).

There are three optional parameters: input rows, output rows, and error rows. The sample packages set these three parameters to -1 in this release because they do not really process any input files. See the [CheckInWorkItem](#_CheckInWorkItem) section in the Appendix for more details.

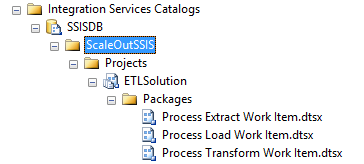
### Deploy ETLSolution project to worker nodes

Use Visual Studio or the Integration Services Deployment Wizard (ISDeploymentWizard.exe) command-line tool to deploy packages in the ETLSolution project to all worker nodes. If you are using Visual Studio, right-click **ETLSolution** project, and select **Deploy**.

Follow these instructions if you use the Integration Services Deployment Wizard:

1. Click **Next** on the **Welcome** page of the **Integration Services Deployment Wizard**.
2. On the **Select Source** page, confirm that the correct ISPAC file is selected, and click **Next**.
3. On the **Select Destination** page, do the following:
   1. Select **Browse** next to the **Path** field.
   2. In the **Browse for Folder or Project** dialog box, select **SSISDB** from the tree view, and click the **New Folder** button.
   3. In the **Create New Folder** dialog box, type **ScaleOutSSIS** for the name, and click **OK**.
   4. Click **Next** on the **Select Destination** page.
4. Review the settings, and click **Deploy** on the **Review** page.
5. Review the results, and click **Close** to close the wizard.

Confirm that the packages have been deployed successfully by using SQL Server Management Studio. The menu should look like this:



#### Configure Connection Manager

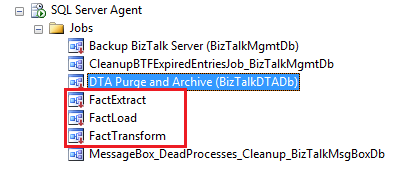
The connection manager in these three packages points to a local server, so you need to update the connection to point to the supervisor machine or worker node.

To update the connection:

1. In **SQL Server Management Studio**, expand **Integration Services Catalogs** > **SSISDB** > **ScaleOutSSIS** > **Projects**.
2. Right-click **ETLSolution**, and click **Configure**.
3. In the **Configure – ETLSolution** dialog box, click the **Connection Managers** tab in the right pane.
4. Click **…** (ellipsis) next to **ServerName**.
5. In the **Value** section, select **Edit Value**, enter the name of the supervisor machine that has the ETLSupervisor database, and click **OK**.
6. Click **OK** to close the **Configure – ETLSolution** dialog box.
7. Repeat the Steps 2-6 for the **ETLSolution** project under the **Projects** node to configure the project Connection Manager that points to the ETLSupervisor database.

### Schedule SSIS packages on worker nodes

After deploying SSIS packages to worker nodes, you need to schedule the packages on the worker nodes. Create a SQL Server Agent job for each SSIS package on each worker node and configure scheduling information for the packages. The menu will look similar to this:



If you want to remotely enable or disable SQL Server Agent jobs on worker nodes from the supervisor machine, you need to create three stored procedures with the same names as the work item types (see the [WorkItemTypes](#_WorkItemTypes) table in the ETLSupervisor database). For the purpose of this sample, create three SQL Server Agent jobs named FactExtract, FactTransform, and FactLoad. If you want to enable or disable SQL Server Agent jobs on each node in a different manner, you can use any names.

To use a worker node as an Extract server, you enable only the FactExtract job on the node. Similarly, to use a worker node as Transform server, you enable only the FactTransform job on the node, and so on. You could have a worker node with one, two, or all three SQL Server Agent jobs enabled, based on your requirements.

**Important**   Disable all the jobs before taking a server offline.

#### Create SQL Server Agent jobs on worker nodes

Create a SQL Server Agent job for each SSIS package on each worker node and configure scheduling information for the packages. For detailed steps, see [Schedule a SSIS Package by using SQL Server Agent](http://msdn.microsoft.com/en-us/library/gg471507.aspx).

1. In SQL Server Management Studio, connect to the instance of SQL Server on the worker node.
2. Expand the **SQL Server Agent** node in Object Explorer.
3. Right-click **Jobs**, and then click **New Job**.
4. On the **General** page, enter **FactExtract** for the job name, select an owner and job category, and optionally, provide a job description.
5. To create a job step for the package, click **Steps** in the left pane, and then click **New**.
6. Enter a name for the job step.
7. Select **SQL Server** **Integration Services Package** for the job step type.
8. In the **Run as** list, select **SQL Server Agent Service Account**, or select a **proxy account** that has the credentials that the job step will use. For detailed steps, see the [Configure proxy account for SQL Server Agent jobs](#_Configure_proxy_account) section.
9. In the **Package Source** list box, select **SSIS Catalog**.
10. Enter the name of the worker node for the server.
11. Click **…** (ellipsis) next to the **Package** text box.
12. Expand **Integration Services Catalogs =** > **SSISDB** = > **ScaleOutSSIS** = > **ETLSolution**.
13. Select **Process Extract Work Item** package, and click **OK**.
14. Click **OK** in the **New Job Step** dialog box.
15. Click the **Schedules** page in the **New Job** dialog box.
16. Click **New** to create a new schedule.
17. In the **New Job Schedule** dialog box:
    1. Enter **ScaleOutSSIS Schedule** for the name.
    2. Select **Daily** for **Occurs** in the **Frequency** panel.
    3. Select **Occurs every** in the **Daily frequency** panel.
    4. Change **Hours** to **Minutes**.
    5. Click **OK** to close the dialog box.
18. Click **OK** the **New Job** dialog box.

Repeat these steps to create **FactTransform** and **FactLoad** jobs with the following settings:

* For the FactTransform job, use the Process Transform Work Item package.
* For the FactLoad job, use the Process Load Work Item package.
* For the FactTransform and FactLoad jobs, use the ScaleOutSSIS Schedule.

#### Configure proxy account for SQL Server Agent jobs

The SSIS packages running on worker nodes read from or write to the ETLSupervisor database on the supervisor machine. These packages access the database by using the account under which the SQL Server Agent is running. This is NT Service\SQLSERVERAGENT by default. Therefore, you need to create and use a proxy that has access to the ETLSupervisor database on the supervisor machine to run the SSIS packages.

Using a proxy account instead of the SQL Server Agent Service Account may resolve common issues that can occur when running a package that uses the SQL Server Agent. For more information about these issues, see [article 918760](http://support.microsoft.com/kb/918760) in the Microsoft Knowledge Base.

To create a proxy, you first need to create a credential and make sure that the user account has appropriate access to the SSISDB database and permissions to run SQL Server Agent jobs on the worker node. Here are the detailed steps you need to perform on the supervisor machine and worker node machines.

**Supervisor machine**

##### **Create a sign-in account**

If you do not have a domain account that already has the access to the database, you need to create a sign-in account. The SQL Server Agent jobs that are running on worker node machines use this account to access the ETLSupervisor database.

1. On the **supervisor** machine, launch **SQL Server Management Studio**.
2. Expand **<machine name>** and **Security**.
3. Right-click **Logins**, and click **New Login**.
4. In the **Login – New** dialog box, enter a sign-in name (such as mydomain\user).
5. Click the **User Mapping** tab in the left pane.
6. Select the **ETLSupervisor** check box from the list at the top of the page.
7. Select **db\_owner** from the list at the bottom of the page.
8. Click **OK** to close the **Login – New** dialog box.

**Worker node machines**

1. Create a sign-in account for the domain account and give it access to the SSISDB and MSDB databases.
2. Create a credential name and password.
3. Create a proxy that uses the credential.
4. Configure SQL Server Agent jobs to use the proxy.

##### To create a login for the domain account and give it the access to SSISDB and MSDB databases:

1. Launch **SQL Server Management Studio**.
2. Expand **<machine name>** and **Security**.
3. Right-click **Logins**, and click **New Login**.
4. In the **Login – New** dialog box, enter a sign-in name (such as mydomain\user).
5. Click the **User Mapping** tab in the left pane.
6. Select the **SSISDB** check box from the list at the top of the page.
7. Select **ssis\_admin** from the list at the bottom of the page.
8. To give this user the appropriate access to create, start, and stop SQL Server Agent jobs, select **MSDB** from the list at the top of the page.
9. Select **SQLAgentOperatorRole** from the list at the bottom of the page.
10. Click **OK** to close the **Login – New** dialog box.

##### To create a credential name and password:

1. In **SQL Server Management Studio**, expand **<machine name>**, and expand **Security**.
2. Right-click **Credentials** and click **New Credential**.
3. In the **New Credential** dialog box, enter a **name** for the credential (for example: domain\user name).
4. Click **…** (ellipsis) next to **Identity**.
5. In the **Select User or Group** dialog box, enter the **domain user** who has access to the **ETLSupervisor** database on the supervisor machine.
6. Type the **password** and confirm it.
7. Click **OK**, to close the **New Credential** dialog box.

For more details, see [How to: Create a Credential (SQL Server Management Studio)](http://technet.microsoft.com/en-us/library/ms190703(v=sql.100).aspx) or [Create Credential (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms189522(v=sql.100).aspx).

##### To create a proxy that uses the credential:

1. In **SQL Server Management Studio**, expand **<machine name>**, and expand **SQL Server Agent**.
2. Right-click **Proxies**, and select **New Proxy**.
3. Specify a **Proxy name**.
4. Click **…** (ellipsis) next to **Credential name**.
5. In the **Browse for Objects** dialog box, Click **Browse**, select the credential you created earlier, click **OK**.
6. Click **OK** on the **Select Credential** dialog box.
7. For the **Active on the following subsystems**, select **SQL Server Integration Services Package**, and then click **OK**.

For more details, see [How to: Create a Proxy (SQL Server Management Studio)](http://technet.microsoft.com/en-us/library/ms190698(v=sql.100).aspx) or [sp\_add\_proxy (Transact-SQL)](http://msdn.microsoft.com/en-us/library/ms188763.aspx).

##### To configure SQL Server Agent jobs to use the proxy:

1. In **SQL Server Management Studio**, expand **Jobs** under **SQL Server Agent**.
2. Right-click the **FactExtract** job, and click **Properties**.
3. In the **Job Properties – Fact Extract** dialog box, click the **Steps** tab in the left pane.
4. Click **Edit** at the bottom of the **Steps** page.
5. In the **Job Step Properties – Process Extract Work Item** dialog box, for **Run as**, select the proxy name you created earlier, and click **OK**.
6. Click **OK** in the **Job Properties – Fact Extract** dialog box.
7. Repeat steps 2 – 6 for the **FactLoad** and **FactTransform** jobs.

#### Enable or disable SQL Server Agent jobs from the supervisor machine

You can enable and disable SQL Server Agent jobs by right-clicking the job and selecting **Enable**/**Disable** in SQL Server Management Studio on each worker node machine (the supervisor machine and worker node machine can be the same). However, in a scale-out environment, you may want to enable or disable these jobs from the supervisor machine.

To do so, you need to add all worker nodes to the ETLNodes table manually and set their status to Active (1). This setting is used by EnableETLNodeJobs and DisableETLNodeJobs stored procedures to remotely enable or disable SQL Server Agent jobs on worker nodes. This table is not used for any other purpose in this release of the sample.

Next, confirm that the name of the SQL Server Agent jobs match the names of work item types in the WorkItemTypes table in the ETLSupervisor database. Then, you can use the [EnableETLNodeJobs](#_EnableETLNodeJobs) or [DisableETLNodeJobs](#_DisableETLNodeJobs) stored procedures to enable or disable SQL Server Agent jobs on worker nodes from the supervisor machine, assuming you have permissions to do so.

### Review SSIS package in CreateWorkflows project

Open the **Create Test Workflow** package in Visual Studio. [Execute SQL Task](http://msdn.microsoft.com/en-us/library/ms141003(v=sql.110).aspx) Check Out Work Item creates a workflow by running the CreateWorkItem stored procedure.

The CreateWorkItem procedure has two required parameters:

* **WorkItemTypeID**   The ID of the first work item type from a work flow. In our sample, it is FactExtract (0).
* **WorkFlowName**   A unique name of a workflow.

The CreateWorkItem stored procedure performs the following important tasks:

1. Creates a workflow in the ETL system by inserting an entry into the WorkFlows table by using the second parameter in the procedure.
2. Creates a work item for each work item type in the work flow. In our sample, there is a work item for each of the work item types: FactExtract, FactTransform, and FactLoad.

For more details about the [CreateWorkItem](#_CreateWorkItem) stored procedure, see the Appendix.

### Deploy CreateWorkflows project to supervisor machine

Use Visual Studio or the Integration Services Deployment Wizard (ISDeploymentWizard.exe) command-line tool to deploy the Create Test Workflow package in the CreateWorkflows project to the supervisor machine.

If you are using Visual Studio, right-click **CreateWorkflows** project, and select **Deploy**.

Follow these instructions if you use the Integration Services Deployment Wizard:

1. Click **Next** on the **Welcome** page of the **Integration Services Deployment Wizard**.
2. On the **Select Source** page, confirm that the correct ISPAC file is selected, and click **Next**.
3. On the **Select Destination** page, do the following:
   1. Select **Browse** next to the **Path** field.
   2. In the **Browse for Folder or Project** dialog box, expand **SSIS Catalog**, select **ScaleoutSSIS**, and then click **OK**.
   3. Click **Next** on the **Select Destination** page.
4. Review the settings and click **Deploy** on the **Review** page.
5. Review results and click **Close** to close the wizard.

### Schedule Create Test Workflow package on supervisor machine

Create a SQL Server Agent job for the Create Test Workflow on the supervisor machine and configure scheduling information for the package. You may not need to create a proxy account for SQL Server Agent jobs.

## Enable SQL Server Agent jobs on supervisor and worker nodes

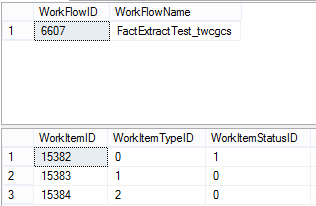
To start testing the sample, first enable the SQL Server Agent jobs:

1. Enable the SQL Server Agent job on the supervisor machine to create workflows.
2. Enable SQL Server Agent jobs on all worker nodes.

If the names of SQL Server Agent jobs match the names of work item types in the WorkItemTypes table (in the ETLSupervisor database), you could do this by running [EnableETLNodeJobs](#_DisableETLNodeJobs) (assuming that you have appropriate permissions).

The SQL Server Agent job on the supervisor machine runs the **Create Test Workflow** package every minute. This package creates work flows by inserting rows in the WorkFlows table of the ETLSupervisor database. It also inserts three rows in the WorkItems table, as shown in the following example:

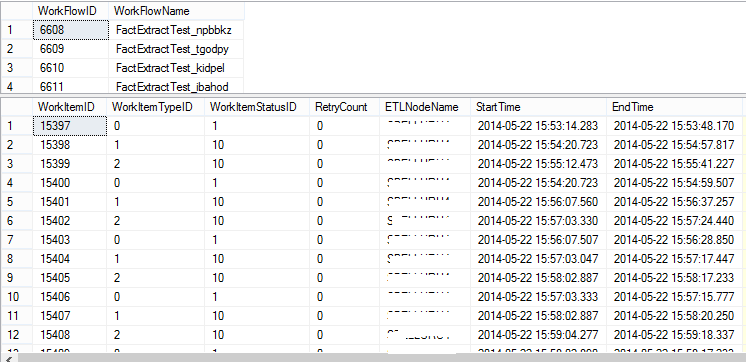
* select \* from workflows
* select \* from workitems



The workflows and work items are created continuously, based on the schedule you configured for the SQL Server Agent job.

The FactExtract, FactTransform, and FactLoad SQL Server Agent jobs on worker nodes launch corresponding Process Extract Work Item, Process Transform Work Item, and Process Load Work Item packages every minute. These packages check out, process, and check in work items with the status information.

You can run a SELECT query against the WorkItems table to see the worker node that processed the work item in the ETLNodeName column, in addition to the start and end times for the work item processing.



## Monitor workflows

As with any software product, the value is measured by the data. How can you effectively know if the system is behaving as expected without a means of viewing or monitoring the data? In this sample, we have included a set of SQL Server Reporting Services (SSRS) reports that will assist you in monitoring your scaled-out deployment of SSIS. The starting point for the reports is a dashboard that provides a high-level overview of your existing environment.

You also have the ability to drill into certain aspects of this report to view data at a more granular level. This allows you to gain more insight as to how any individual work item is performing overtime. This approach provides you with a means to proactively check for slowness that may overtime lead to bottlenecks and other unforeseen issues.

### Deploy ReportingSolution project

Perform the following steps on a machine with SQL Server Reporting Services (SSRS) installed:

**Note** You may need to run Visual Studio with administrator privileges (Run as administrator) to successfully deploy the solution to a report server.

1. In SQL Server Data Tools (in Visual Studio), right-click **Reporting Solution** in the **ScaleOutSSIS** solution, and click **Properties**.
2. Enter **ScaleOutSSIS-ReportingSolution** for the **Target Report Folder**.
3. Enter the URL (for example, <http://localhost/ReportServer>) for the **TargetServerURL**.
4. Click **OK** to close the dialog box.
5. Right-click the **Reporting Solution** project, and then click **Deploy** to deploy the reporting solution to the configured report server.

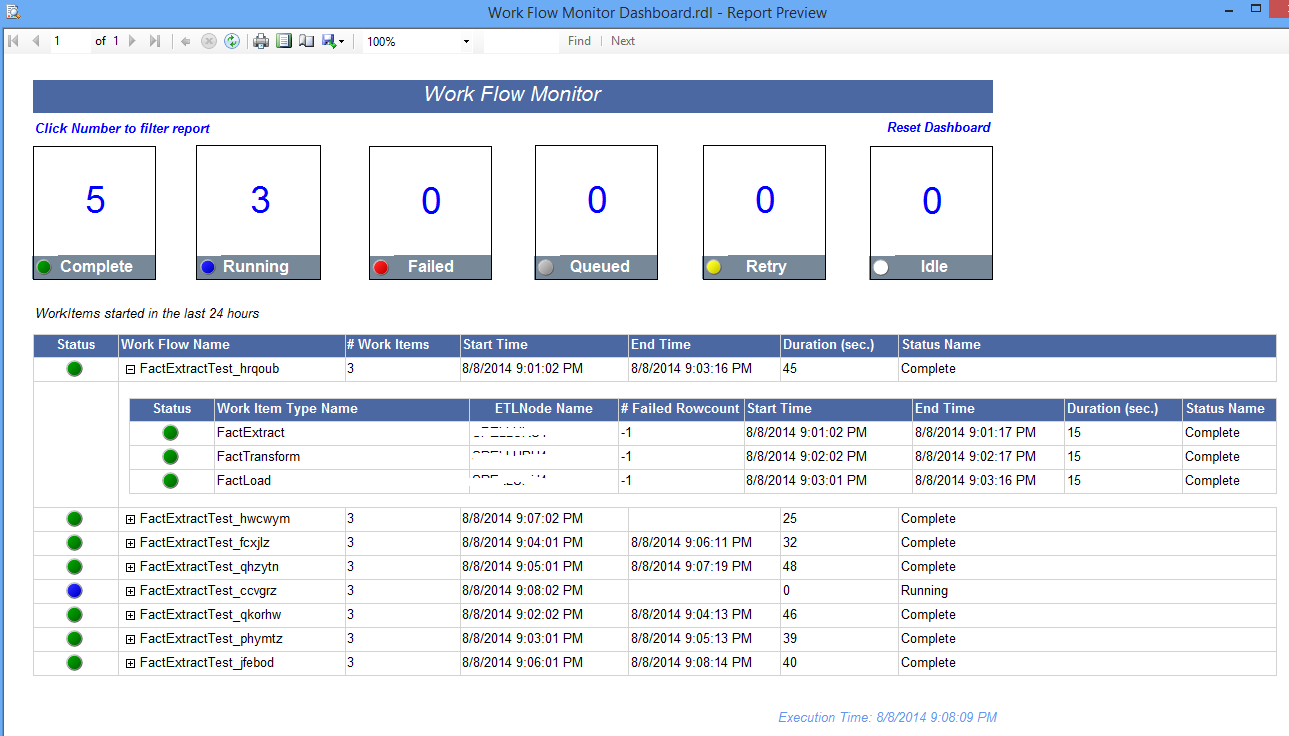
### Work Flow Monitor Dashboard

To launch the Work Flow Monitor Dashboard:

In Visual Studio, right-click **Work Flow Monitor Dashboard.rdl** and click **Run** (or navigate to <http://localhost/ReportServer>), click the **ScaleOutSSIS-ReportingSolution** folder, and then click **Work Flow Monitoring Dashboard.rdl**. This dashboard provides you with a quick overview of the scaled-out SSIS environment.

**Note** You may need to run the Visual Studio or Internet Explorer with administrator privileges (Run as administrator) to successfully the access the report on the report server and run it.

The following screenshot shows an example report preview.



Towards the top of the report is an aggregation of each possible work item status. Directly under the status is a table that lists work flows that started within the last 24 hours. Each value in the aggregation listing that is greater than zero has an action to access detailed data about the work flows and the corresponding status.

In addition, you can expand each work flow name in the table to view the corresponding Work Items. You can see the information about each work item, such as the status and the name of the node that ran the work item.

You will see that a workflow runs in the order of Extract (E), Transform (T), and Load (L), which is defined in the WorkItemTypes table. You will also see the worker nodes processing different work items at different phases (E, T, or L) of different workflows. You need to manually refresh the report to see the changes in the dashboard.

## Maintenance

It is common to take a server offline for maintenance or to add servers to increase the processing capability of an ETL system. You can easily add or remove worker nodes in an ETL system. It is also possible to analyze the ETL system for any hung or timed-out jobs.

### Add and remove processing servers

To add a worker node, follow the instructions in the [Deploy SSIS packages to worker nodes](#_Deploy_Packages_in) section. To deploy and schedule SSIS packages on the new worker node, follow the instructions in the [Schedule SSIS packages on worker nodes](#_Schedule_SQL_Agent) section. When you enable SQL Server Agent jobs on a worker node, the node starts processing work items.

If you want to enable or disable these SQL Server Agent jobs remotely from the supervisor machine, follow the instructions in [Enable or disable SQL Server Agent jobs from the supervisor machine](#_Enable/disable_SQL_Agent).

To temporarily take a worker node offline, you disable the SQL Server Agent jobs manually on the node or remotely from the supervisor machine.

### Hung jobs and timeout

Planning for job failure in a complex system is very important. For a typical failure scenario, it is desirable for the ETL processes to automatically restart; however, if a process fails a threshold many times, it is best to set that task aside for manual investigation. The WorkItems table can be queried for work items that have failed. If the package was properly designed, a detailed error message should accompany the record. In the event of multiple failures, you should aggregate these error messages to see if you can recognize a pattern.

Depending on the frequency of the work items (for example, hourly, daily, or weekly), the WorkItems table should be inspected for hung jobs. A best practice is to set up an alerting job that periodically inspects the table for issues and then sends a message—for example through System Center Configuration Manager (SCCM), email, or SMS text. This will free the monitoring staff to watch for greater issues, such as planning for capacity issues and managing customer data requests.

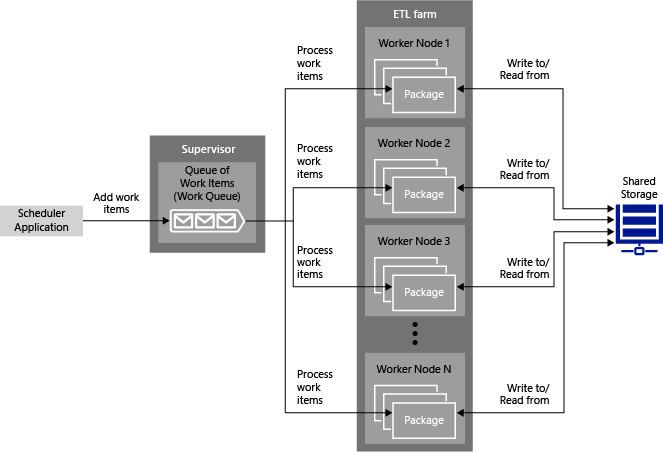
### Archive work items

A common mistake is to overload the supervisor with too many requests. The symptom will be excessive transactions on the supervisor, and the processor will likely be running at 100%. It is best to throttle the requests in the SSIS package by increasing the delay between attempts to the supervisor when the last attempt returns no task assignment. A reasonable time interval for requesting jobs to the supervisor is 1 minute, and then you can increase the time from there.

Another common mistake is not regularly cleaning the WorkItems table. Keeping only the most recent history in the WorkItems table is important to maintaining the speed of the system. You should periodically archive old data from the WorkItems table into an archive table to ensure that the size of the WorkItems table does not affect the performance of the ETL system.

# Alternate implementations

You could customize the architecture as shown in the following diagram so that you have only one type of work item, and all the worker nodes process all work items.



In this case, SSIS packages on worker nodes perform all the stages of the processing (extract, transform, and load).

You could also have different types of workflows. For example, you could have a work flow with the following sequence of operations:

1. Extract, transform, and load DimRegion data. (DimRegion data refers to data from a country/region.)
2. Extract, transform, and load DimCustomer, DimPart, and DimSupplier in parallel.
3. Extract, transform, and load DimPartSupplier.

Each instance of this workflow contains six work items, and they have to be processed in the sequence.

# Appendix

This appendix includes details about:

* [Stored procedures](#_Stored_Procedures)
* [Table Value Functions](#_Table_Value_Functions)
* [Tables](#_Tables)
* [Views](#_Views)

## Stored procedures

### Runtime stored procedures

#### CreateWorkItem

The **CreateWorkItem** stored procedure creates a workflow and a work item for each work item type in the workflow.

For example, let's say you defined three work item types: FactExtract(0), FactTransform (1), and FactLoad(2). You set the following prerequisite work item types:

* FactTransform (1) is set to FactExtract (0)
* FactLoad (2) is set to FactTransform (1)

This forms a workflow with the following sequence of work item types: FactExtract => FactTransform => FactLoad.

When you invoke the CreateWorkItem stored procedure with 0 as the work item type and “testwf” as workflow name, the stored procedure creates a work flow entry in the WorkFlows table. It also creates three work items in the work items table for each work item type: 0,1, and 2.

The CreateWorkItem stored procedure takes the following parameters:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| WorkItemType | Specifies the work item type. This type must be in the WorkItemTypes table and the first type in the workflow. In the previous example, it would be 0 (FactExtract). This parameter is mandatory. |
| WorkflowName | Specifies a name for the workflow. This parameter is mandatory. |
| FileName | Specifies the input file name. This parameter is optional. |
| InputPath | Specifies the path for the input file. This parameter is optional. |
| OutputPath | Specifies the path for the output file. This parameter is optional. |

Here is an example:

CreateWorkItem 0, ‘Test Workflow’

CreateWorkItem 0, ‘Test Workflow’, ‘myfile.txt’, ‘c:\input’, ‘c:\output’

This procedure performs the following steps:

1. Creates an entry in the WorkFlows table by using the WorkflowName parameter.
2. Based on the WorkItemType parameter (for example, 0), gathers all subsequent work item types (for example, 1 and 2). The work item types should be in the work flow based on prerequisite type ID.
3. For each work item type from the list prepared in the previous step, this example creates an entry in the WorkItems table. WorkItemLevel in the WorkItems table is calculated based on WorkItemPrereqID values in the WorkItemTypes table.

* For level = 0, the input file name is the same as the parameter, and the output file name is: input file name\_workitemtypeid.<extension from inputfile>.   
  For example, c:\output folder\test\_0.txt.
* For level = 1, the input file name is the output file of the level 0, and the output file name is: original input file name\_workitemtypeid.<extension from inputfile>.   
  For example, c:\output folder\test\_1.txt.
* For level = 2, the input file name is the output file of the level 1, and the output file name is: original input file name\_workitemtypeid.<extension from inputfile>.   
  For example, c:\output folder\test\_2.txt.

The input path parameter represents the input path for E, and the output path parameter represents the final output path for L. The intermediate locations (the output location for E and the input and output locations for T) automatically use a round robin algorithm. Specifically, they use the GetRoundRobinWorkItemLocation stored procedure.

#### QueueWorkItems

When you create work items by using the CreateWorkItem stored procedure, the work items are in an idle state (WorkItemStatusID=0). In the current release, CheckOutWorkItemsByType checks out work items even though they are in an idle state. You could modify the stored procedure to not pick up idle work items and have an additional step to queue work items by invoking the QueueWorkItems stored procedure after creating the work items by using the CreateWorkItem stored procedure.

#### CheckOutWorkItemByType

The main SSIS package running on each node in the group invokes the CheckOutWorkItemByType stored procedure to check out a work item to process. The package passes the type of work item (FactExtract, FactTransform, or FactLoad) that it processes and the name of the machine as parameters to this stored procedure. This stored procedure performs the following tasks:

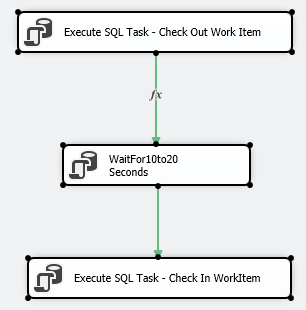
1. From all the work items in the WorkItems table, it gets work items with a status of 0 (Idle), 1 (queued), 77 (retry), or 5 (running but timed out).
2. Sorts the list by their creation times and selects the first work item after sorting the work items in the ascending order.
3. Sets the start time to the current time.
4. Sets ETLNodeName to the name of the node passed to this procedure for the work item.

The main SSIS packages on each node implement the following pattern:

CheckOutWorkItemByType => Actual Logic for Processing Work Item => CheckInWorkItem

(Execute SQL Task) (Tasks to implement processing logic) (Execute SQL Task)

The package may optionally call child packages. In the following example, the package waits for 10-20 seconds.



#### CheckInWorkItem

The master SSIS package that is running on each node in the group invokes the CheckInWorkItem stored procedure to check in a work item after it is done processing. The package passes the ID of the work item that it processed, the name of the machine or node, and the processing status (success or failure). You can optionally pass in the number of input rows, output rows, and error rows. The stored procedure performs the following tasks:

1. Ensures that this work item was checked out by the ETL node by looking at ETLNodeName for the work item in the WorkItem table. If it was checked out by the ETL node, it starts the check in process.
2. If the work item has completed successfully (success parameter is set to 1), it does the following:
   1. Updates the work item in the WorkItems table with a status of 10 (complete), sets the end time to the current time, adds the input row count, output row count, and failed row count to the values that were passed as parameters by the SSIS package that processed the work item.
   2. Checks for the next work item in the workflow based on the work-item level (for example: 1 after 0, and 2 after 1), and sets the status of work item to 1 (queued) so that the work item gets picked up by a free ETL node.
3. If the work item did not complete successfully, it sets the status to 77 (retry) and end time to the current time.

#### GetRoundRobinWorkItemLocation

The GetRoundRobinWorkItemLocation stored procedure is invoked by the CreateWorkItem stored procedure. It selects a location for the output files in a round-robin fashion to avoid all packages writing to the same location to generate output files.

#### WaitFrom10to20Seconds

WaitFrom10to20Seconds provides a test procedure that is used by the SSIS packages in the sample. It runs the “waitfor delay” SQL statement with a value that is randomly generated and is between 10 and 20.

#### ParseFileName

The ParseFileName stored procedure is invoked by the CreateWorkItem stored procedure. It parses a file name that includes the path to separate the file name and extension. For example, from c:\test\file1.txt, it extracts file1 and .txt.

### Administrative stored procedures

#### CreateWorkItemTypes

The first step in using the sample provided in this whitepaper is to populate the WorkItemTypes table, which contains the types of work items in your ETL workflow. For example, if your ETL workflow supports three work item types: Extract, Transform, and Load, there are three SSIS packages associated with each work item type.

You can use the CreateWorkItemTypes stored procedure to create work items in this table. The stored procedure has three required parameters—one is optional and two have default values. The following table provides a description of each parameter.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Description** | **Required** |
| WorkItemTypeName | Name of the work item type. You may want to name a work item type with the same name as the package that is associated with this type. | Yes |
| WorkItemTypeDesc | Description of the work item type. | Yes |
| WorkItemRetryLimit | Number of times the package that corresponds to this type will be run in the event of a failure. | Yes |
| WorkItemTypePrereqID | Integer value that ensures the packages are run in a specific order. | No |
| Active | Boolean value that specifies whether a type will be run. The default value is 1 (Yes). | No |
| TimeOutSec | Integer value that determines how long to attempt running a work item before it is stopped. The default value is 60 seconds. | No |

The following code shows a sample run of the CreateWorkItemTypes stored procedure:

EXEC dbo.CreateWorkItemTypes

@WorkItemTypeID = 0,

@WorkItemTypeName = 'FactGrade',

@WorkItemTypeDesc = 'Loads incoming data into the Grade Fact',

@WorkItemRetryLimit = 5

You need to run this stored procedure for each work item type in the work flow. For example, if you want to build a process that loads two dimensions and one fact, you need to run the stored procedure three times.

#### EnableETLNodeJobs

There is a one-to-one relationship between work item type, SSIS package, and a SQL Server Agent job. A work item type corresponds to an SSIS package. A SQL Server Agent job is created to schedule the SSIS package to automate the processing of work items of that type. A node may have one or more SQL Server Agent jobs depending on the types of work items that you want that node to process.

This stored procedure enables SQL Server Agent jobs on active ETL nodes (entries in the ETLNodes table with the Active flag set to True).

**Important**

You need to ensure the following for the EnableETLNodeJobs stored procedure to work:

* Name of the work item type matches the name of the SQL Server Agent job running the package that processes work items of that type. For example, if the name of the work item type is FactExtract, the name of the agent job must be FactExtract.
* Enable xp\_cmdshell. For more information, see [xp\_cmdshell Server Configuration Option](http://msdn.microsoft.com/en-us/library/ms190693.aspx).

Update the user name and password in the definition of the stored procedure before using it.

#### DisableETLNodeJobs

The DisableETLNodeJobs stored procedure disables SQL Server Agent jobs on all active ETL nodes so that the nodes can process work items. For more information, see [EnableETLNodeJobs](#_DisableETLNodeJobs).

#### StopETLNodeJobs

This StopETLNodeJobs stored procedure stops and then disables SQL Server Agent jobs on active ETL nodes. For more information, see [EnableETLNodeJobs](#_DisableETLNodeJobs).

### Reporting stored procedures

#### rptGetETLNodes

The rptGetETLNodes stored procedure returns a list of active ETL nodes (IDs and names) that will be used by various reports.

#### rptGetWorkflowDetails

The rptGetWorkflowDetails stored procedure returns a list of workflows and their statuses, which is displayed on a reporting dashboard. It specifically returns the following information:

* Workflow ID
* Workflow name
* Number of work items in the workflow
* Workflow start time (start time of the first work item in the workflow)
* Workflow end time (end time of the last work item)
* Duration of the workflow
* Workflow status.

#### rptGetWorkItemsDetails

The rptGetWorkItemsDetails stored procedure returns a list of work items and their statuses for a specific workflow. This procedure is used by a reporting dashboard. It returns the following specific information of work items in a workflow:

* Workitem ID
* Name of the ETL node on which the work item was processed
* Workitem type
* Start time for the work item
* End time for the work item
* Duration of the work item
* Input location path
* Output location path
* Input row count
* Output row count
* Difference between input row count and output row count
* Failed row count
* Work item status.

#### rptGetWorkItemsStatusCounts

The rptGetWorkItemsStatusCounts stored procedure returns a list of work items and their statuses, which will be used to populate counts at the top of the monitoring dashboard. For example, for each ETL node, the dashboard displays the number of work items in the following states: Complete, Idle, Queued, Running, Failed, and Retry. The stored procedure returns the status name, ETL node name, and number of work items that have the status.

### Uncategorized stored procedures

#### QueryWorkItemTypeBreakDown

The QueryWorkItemTypeBreakDown returns a list of all work item types, based on their prerequisites and levels. For example, if you pass 0 and Fact Extract as the input parameter, you get a table with three rows: 0, 1, and 2 levels and their names as output values.

## Table-valued functions

### rptGetCurrentWorkflowStatus

This function is used by the rptGetWorkflowDetails stored procedure. It takes the workflow ID as a parameter and returns ID, status ID, the name of the workflow, and the ETL node on which the last work item of the workflow ran.

### rptGetWorkflowDuration

This function is used by the rptGetWorkflowDetails stored procedure. It takes the workflow ID as a parameter and returns the ID, start time of the workflow, and duration of the workflow (end time of the last level item minus the start level of the first level item (for example, the end of the L (Load) operation minus the start of the E (Extract) operation in our example).

## Tables

The WorkFlows and WorkItems tables are updated when you create work items by using the CreateWorkItem stored procedure. Work items are processed by using SSIS packages (the CheckOutWorkItemByType and CheckInWorkItem stored procedures).

The ETLNodes, WorkItemStatus, and WorkItemTypes need to be maintained manually by you by inserting, updating, and deleting entries in these tables.

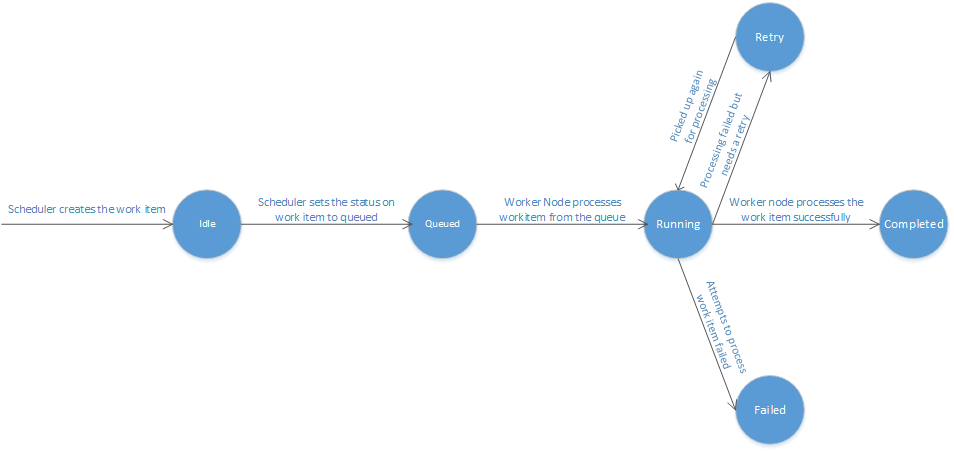
### ETLNodes

You register all your ETL nodes by inserting node names into this table. You set the Active status to 1 (Active) or 0 (Inactive). When you run EnableETLNodeJobs or DisableETLNodeJobs, the stored procedure looks for ETL nodes that are active at that time and enables or disables SQL Server Agent jobs on these nodes.

### WorkItemStatus

The WorkItemStatus table contains the following statuses of a work item in a work flow.

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Description** |
| 0 | Idle | Work item is idle and not ready to be placed into the queue |
| 1 | Queued | Work item is in the queue |
| 5 | Running | Work item is assigned and running on an ETL node |
| 10 | Complete | Work item has been completed |
| 77 | Retry | Work item attempted to run and failed somewhere |
| 99 | Failed | Work item retry limit has been reached and will no longer be retried |



### WorkItemTypes

The WorkItemTypes table is very important. You start defining the workflow by using this table. For example, you can define three work items (Extract, Transform, and Load), with Extract as a prerequisite for Transform, and Transform as a prerequisite for Load. This iss shown in the following example.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Prerequisite ID | Notes |
| 0 | FactExtract | NULL | No prerequisite for E in ETL workflow |
| 1 | FactTransform | 0 | E is the prerequisite for T in ETL workflow |
| 2 | FactLoad | 1 | T is the prerequisite for L in ETL workflow |

You can easily define a different type of workflow, for example, ELT workflows. The WorkFlows table contains information (ID and name) about workflows that ran or currently running in the ETL farm. You don’t insert entries directly into this table. Instead, you use the CreateWorkItem stored procedure, which automatically inserts a workflow entry in this table and creates a work item for each work item type in the workflow. Work item type are defined by using the WorkItemTypes table.

### WorkItems

The WorkItems table contains information about work items in work flows that ran or are currently running in the ETL system. The CreateWorkItem stored procedure creates three work items for each invocation, one for each work item type. CheckOutWorkItemByType updates the work item with the node name and start time. CheckInWorkItem updates the work item with a status of processing and end time. There are several columns, including Input Location Path, Output Location Path, Input Row Count, Output Row Count, and Failed Row Count, which are not used in this sample. You can use these columns when you customize the sample to process an input file and generate an output file.

### WorkItemLocations

The WorkItemLocations table contains locations for storing the intermediate data within a work flow. For a work flow, there is an input path, and an output path is specified, but there may be intermediate files generated. For example, in an E-T-L workflow, the Extract operation writes the output to an intermediate store, the Transform operation picks up data from this store, performs transformations, and stores the result data in the intermediate store. The Load operation picks up the data from the intermediate store and loads it into an output file (for example, a flat file or a database).

## Views

### WorkItemsMonitor view

The WorkItemsMonitor view is used to build an SSRS-based report. You can run this view by using SQL Server Management Studio to see consolidated information about all the workflows and the work items in the workflows.

Did this paper help you? Please [send us your feedback](mailto:spelluru@microsoft.com?subject=White%20Paper%20Feedback:%20[Scale-Out%20SQL%20Server%20Integration%20Services%20Environment]). On a scale of 1 (poor) to 5 (excellent), how would you rate this paper? Also, please tell us why have you given it this rating, for example:

* Are you rating it high because it has good examples, excellent screen shots, or clear writing?
* Are you rating it low because it has poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of the white papers we release.