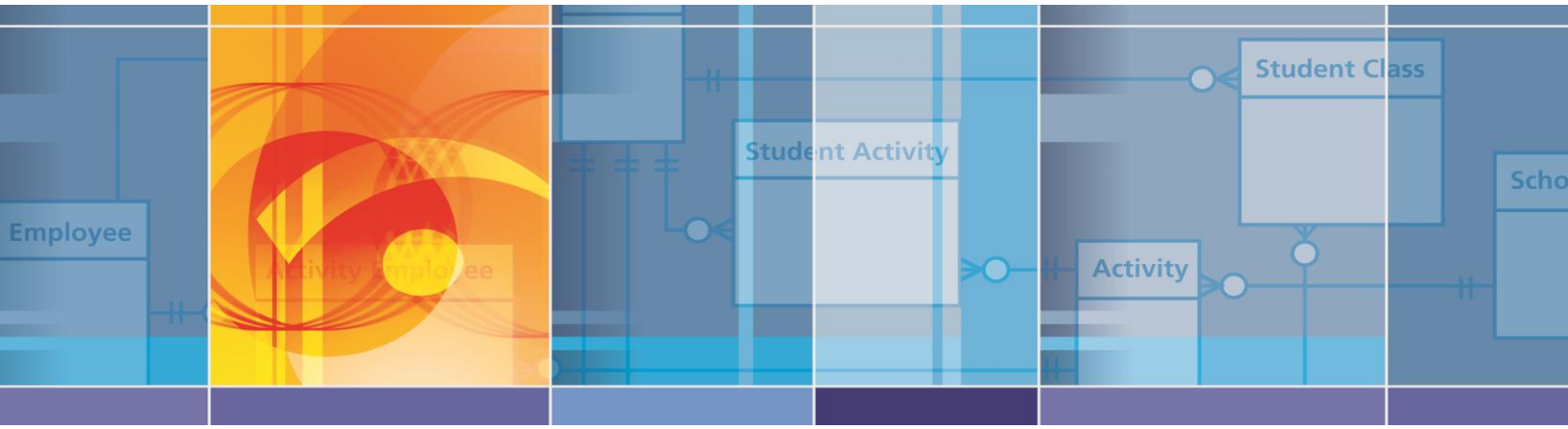


EDFacts Shared State Solution

Deployment Guide

ESP Solutions Group

Authors: Steven King
Date: 9 July 2016
Version: 0.1

Change History Log

Date	Summary of Changes	Version	Authorized By
5-Jan-2012	Initial Draft	v0.1	Steven King

Table of Contents

Preface

This document describes the hardware, software requirements and the steps required to install and configure a new EDFacts Shared State Solution implementation.

Target Audience

This document is targeted at the state IT and ESP staff charged with installing, configuring, and implementing an EDFacts Shared State Solution environment. The solution includes information for database administrators, database programmers, web administrators, web developers, user and system account and security staff.

Acronyms

ES3	EDFacts Shared State Solution
AD	Active Directory
App Pool Account	
BIDS	Business Intelligence Development Studio (SQL Server 2008 R2)
IIS	Internet Information Server
MSDB	
SQL Server	
SSDT	SQL Server Data Tools (SQL Server 2012 and beyond)
SSIS	SQL Server Integration Services
SSIS DB	SSIS database or catalog where SSIS packages can be stored in SS Versions 2012 and newer
SSIS Repository	

Related Documents

ES3 Development Guide

ES3 Web Management App Users Guide

ES3 Overview

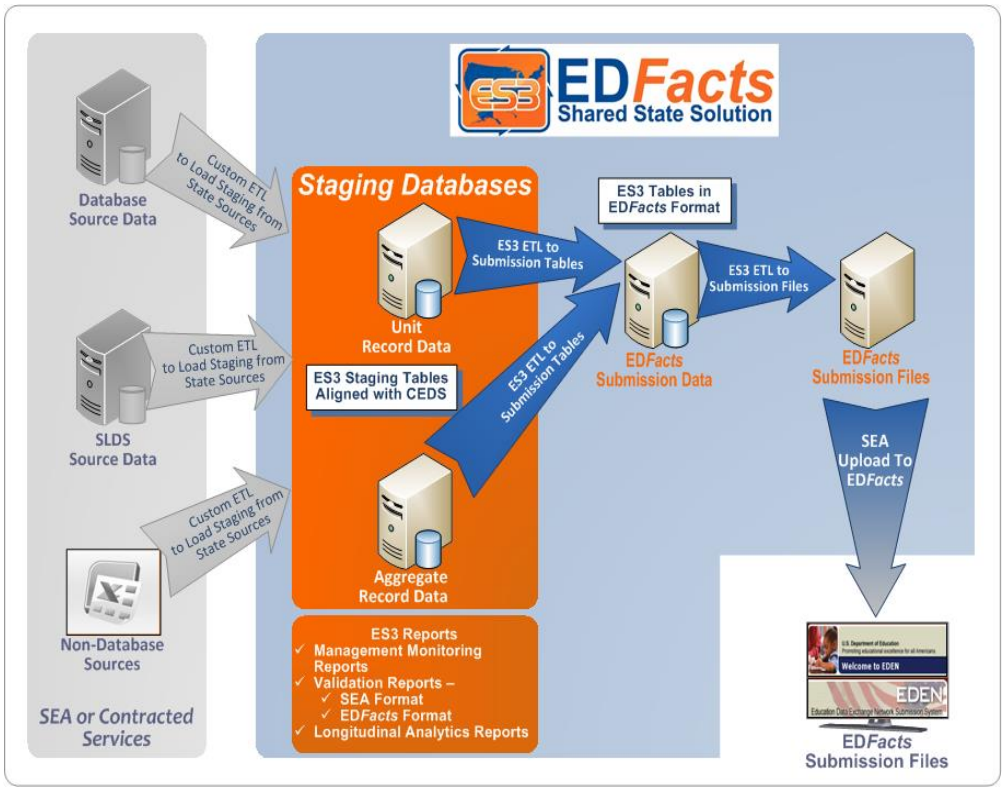
The ED*Facts* Shared State Solution (ES3) is a multi-state collaborative effort to minimize duplicate state effort in reporting ED*Facts* data to the US Department of Education (ED). ED revolutionized state-to-federal reporting with the ED*Facts* system. Every state is mandated to submit data in the same format. Most of the core processes are duplicated within every SEA.

Several states are coming together to develop a shared solution that is easily configurable to a particular state and maximizes the common processes while minimizing duplicate effort. ESP Solutions Group is managing this effort among states to share a common approach to reporting for ED*Facts*.

System Components and Interactions

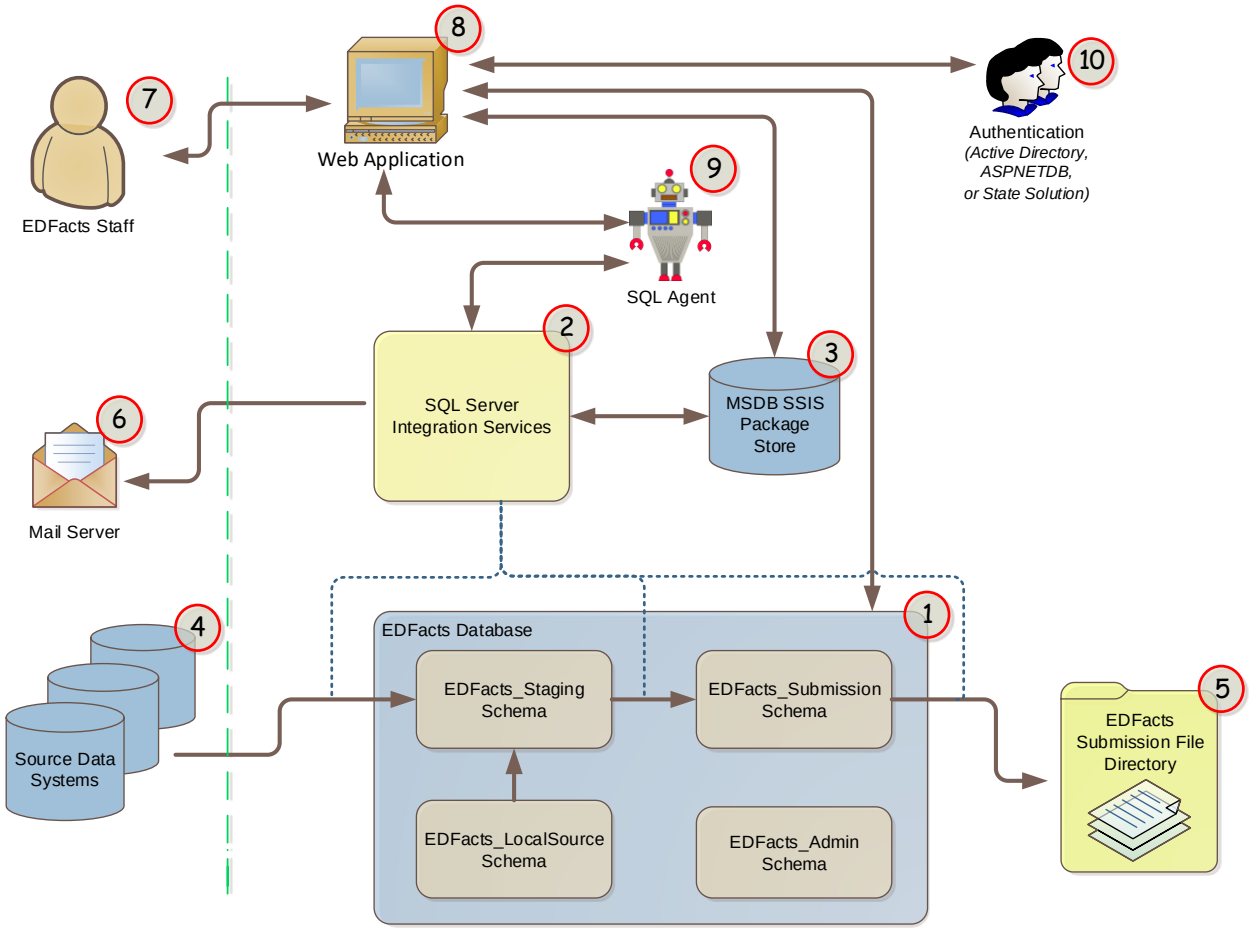
The ES3 consists of:

- **Staging Tables**: A set of SQL tables that store staged source data,
- **Submission Tables**: A set of SQL tables that hold the ED*Facts* data in the ED*Facts* submission file format,
- **Stage Loading SSIS Packages**: SQL Server Integration Services (SSIS) ETL packages that load the staging tables
- **Submission Loading SSIS Packages**: SSIS ETL packages that take the staged state data and load the ED*Facts* submission tables and then create the ED*Facts* submission files,
- **Stored Procedures**: SQL stored procedures that do most of the ETL work and are called by the SSIS packages,
- **SSIS Repository**: The SSIS packages are stored in the SQL Integration services repository in an EDFacts subdirectory
- **Configuration Tables**: A set of SQL tables that hold state configuration information,
- **Logging and Audit Tables**: A set of SQL tables that hold process history and logging/auditing information,
- **Web Management Interface**: A web application that gives the ED*Facts* coordinator access to operating and managing the solution.

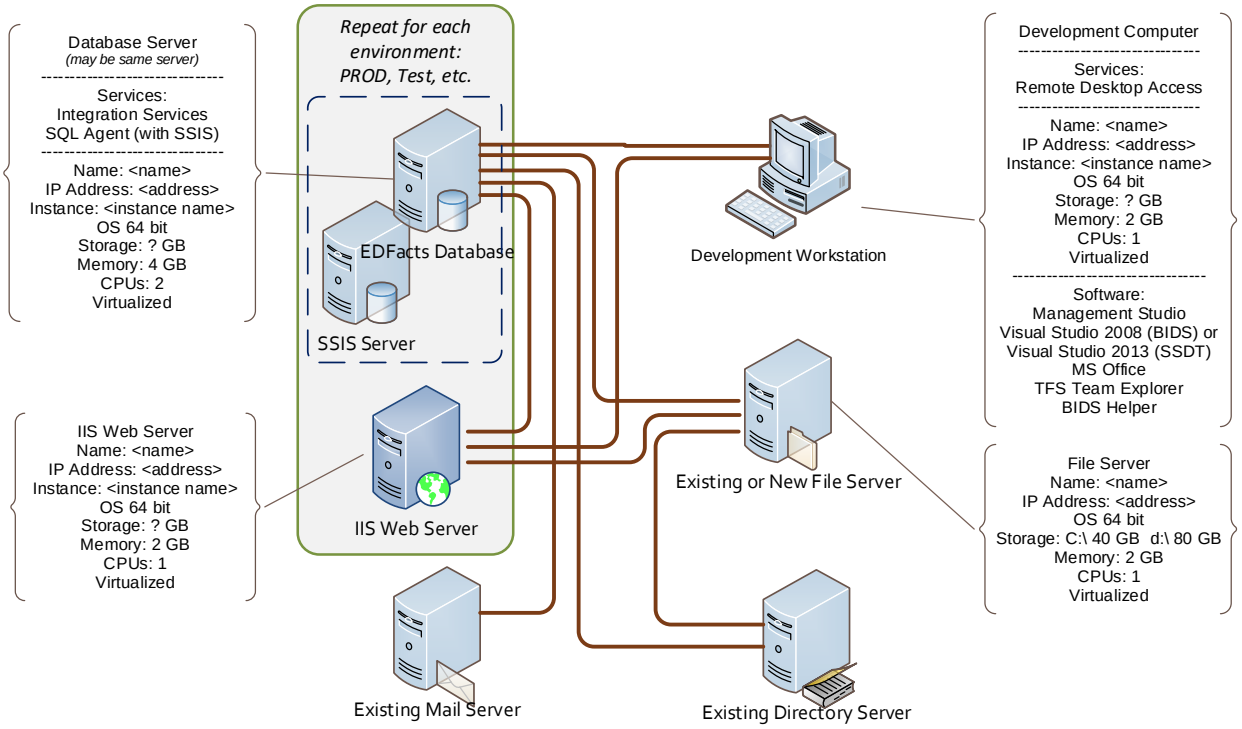


System Architecture

Conceptual System Architecture Model



Physical Implementation



Hardware and Software Requirements and Recommendations

The required/recommended virtualized hardware and software components for ES3 are as follows:

Server Role	Recommended Hardware	Required Software
Database Server(s): • Database Server • ETL (SSIS) Server • SQL Agent The EDFacts database and SSIS server can be together or separated. SQL Agent has to be running on the SSIS Server	OS: 64 bit CPU: 2 quad core RAM: 4 GB Disk: 100GB	Windows Server 2008 R2 or Newer SQL Server 2008 R2 • Integration Services OR SQL Server 2014 • Integration Services
File Server: • File Storage	OS: 64 bit CPU: 1 quad core RAM: 2 GB Disk: 100GB	Windows Server 2008 R2

Server Role	Recommended Hardware	Required Software
Development Workstation	OS: 64 bit CPU: 2 quad core RAM: 4 GB Disk: 150GB	Windows SQL Server Management Studio Microsoft Office (Excel and Word) Team Explorer Windows PowerShell SQLPS module ----- Either ----- SQL Server Business Intelligence Development Studio (BIDS) VSTS 2008 Forward Compatibility Update BIDS Helper ------ Or ------ SQL Server Data Tools (vs 2013)

The servers can all be virtualized. All three roles can be combined on a single server. Many installations place the EDFacts database on an existing database server and use existing file servers for the EDFacts file system.

The Development workstation must be available to the contractor via a remote desktop connection assuming appropriate VPN connections.

Component Configuration and Deployment

In order to set-up and configure the ES3 solution, the state will need to provide the environment within which the solution will function.

Workstation Set-up

The ESP staff will need a workstation/workspace inside the state firewall that has access to all the data and systems used by ES3. This can be a virtual machine or rights on one of the servers. Most states prefer virtual work spaces.

This arrangement means all state data can remain within the security confines of the state firewalls. ESP will not store student data outside the state environment.

ESP may have multiple staff supporting a particular client and each should have their own space so they can work in parallel.

The basic system requirements are:

- **VPN/RDC**: some method of remotely accessing the workstation. Each state has its own preferred mechanism for VPN and Remote Desktop access.
- **Data tools (BIDS or SSDT)**: the workstation should have the appropriate version of SQL Server business intelligence tools. These are licensed to

the state as part of SQL Server. The proper tool depends on the version of SQL Server to be used:

  - SQL Server 2008r2 – use Business Intelligence Development Studio (BIDS). This is installed from under Client Tools on the SQL Server installation disk. We will need to install BIDSHelper. This includes a Visual Studio 2008 shell.
  - SQL Server 2014 – use SQL Server Data Tools (SSDT). This is installed from the Client Tools section of the SQL Server install disk, or directly from Microsoft. This install a Visual Studio 2013 shell.

- **SSMS**: the appropriate version of SQL Server Management Studio
- **Team Explorer**: the appropriate version of team explorer for the Visual Studio used by the data tools. This will give access to the ESP TFS where the source code is kept.
- **Set-up TFS Workspace(s)**: each ESP staff member will need a place to store and work on the source code – a local TFS workspace. This can be a network folder or a local workstation folder (if space allows). Using the TFS connection from above, SEP will copy the source code to the workspace for edits and deployment.
- **Microsoft Office (Word and Excel)**: to facilitate data exchange with EDFacts staff, the workstations should have access to Word and Excel.
- **Browser**: the workstation will need a browser that can access the internal web site where the ES3 Management app is deployed, and the internet in general (port 80). Our TFS connection is through port 80.

ESP Staff will work with state IT staff to get the workstations set up and configured appropriately. Depending on the security rights on the workstation, ESP can do most of the configuration, or will have to rely on state staff to do the configuration with ESP guidance.

Windows Active Directory Group and Service Accounts

Various active directory accounts will need to be created in the state directory tree

EDFacts Group

The ESP staff will need accounts that have access to the source databases, and access to file system directories as outlined below. Our recommendation is to create and EDFacts or ES3 security group, and assign the ESP staff accounts to that group. State EDFacts staff can be members of this group as well.

IIS App Pool Service Account

The ES3 Web Management application will need a network account under which to run. That account will need access to read source data, read/write EDFacts database tables, create and execute SQL agent jobs, monitor ES3 Agent jobs, and write to the EDFacts system directory.

EDFacts File System Directory

We will need file system access into which the SSIS packages can write the EDFacts submission files. This can be a single directory location, or multiple locations – one for each environment, Dev, UAT, Prod, etc. The root folder for each environment will need to be accessible via a network share and UNC map. These locations will need to be accessible by the EDFacts coordinator. ES3 needs to be able to store/have access to:

- Production submission files
- Dev/test submission files
- Exported staging data
- Non-database source data – text files, etc.

A typically single directory structure looks like:

```
EDFacts
    2014-2015
        Export
        S002
        S004
        ...
    2015-2016
        Export
        S002
        S004
        ...
    ...
    UAT
        2014-2015
            Export
            S002
            S004
            ...
        2015-2016
            Export
            S002
            S004
            ...
        ...
    Source data
```

This structure has a single EDFacts root directory. Within it are folders for each reporting period. Each reporting period contains a folder to exported stage data and a folder for each submission file.

If there are additional environments—Dev, Test, or UAT—they have their own folders and mirror the structure of the root production folder. These could easily

be their own directories separate from Production rather than with the production directory.

There is a folder for non-database source data or for the exchange of other files between ESP Staff and State EDFacts staff.

The root folder for each reporting period and the directory path for its "export" folder are stored in the EDFacts_Admin.State_Config table – one record for each reporting period. The subfolder (S002, s004,etc.) used for each submission file is stored in the EDFacts_Admin.SubmissionFileCharacteristic table.

SQL Server EDFacts Database

We need a SQL Server database named "EDFacts." The database can be relatively small 10-15GB. It can be on a stand-alone server, its own VM, or shared with other databases. The number of simultaneous connections will likely be in the single digits.

There are scripts and SSIS routines to completely rebuild the database, so back-up and recovery requirements are minimal. The database does not need to keep transaction history – i.e. we don't need to know what the DB looked like last week.

Schemas

In the EDFacts Database we create the following schemas: EDFacts_Admin, EDFacts_Staging, EDFacts_Submission, and EDFacts_Validation. If we are comparing our results with legacy submissions, then we need an EDFacts_Compare schema.

```
use EDFacts
go
create schema EDFacts_Admin
go
create schema EDFacts_Compare   -- optional
go
create schema EDFacts_Staging
go
create schema EDFacts_Submision
go
create schema EDFacts_Validation
```

EDFacts_Admin Tables

EDFacts_Admin Table Data

EDFacts_Admin Utility Procedures and Functions

EDFacts_Submission Tables

EDFacts_Submission ETL Procedures

Submission Validation Procedures

EDFacts_Staging Tables

EDFacts Validation Tables and Procedures

SSIS and SQL Agent

SQL Server Integration Services must be installed. It is most convenient if SSIS is installed on the same server as the EDFacts database, but it can be installed on a separate server. If on a separate server, the SSIS packages that run on this server must be able to connect to EDFacts database.

The SQL Agent must be installed and running on the same server as SSIS.

We will store the SSIS packages in the database, not in the file system. There are different storage location possibilities depending on the version of SQL Server in use.

In SQL server 2008 r2, the only option is to store them in the MSDB system database. Beginning with SQL Server 2012, a separate SSISDB catalog is available, and is the preferred storage location going forward. Each environment is described in detail below.

In either case we will need to:

- Set up SSIS repository and folder structure
- Create EDFacts Job Category for SQL agent
- Create views against SSIS repository that only expose ES3 packages
- Create views against job monitoring system tables that only expose ES3 jobs
- Deploy stored procedure that creates a job and runs it for the selected package

Refer to one of the following two sections as appropriate based on where the SSIS packages are deployed; either an SSISDB or into the MSDB system repository.

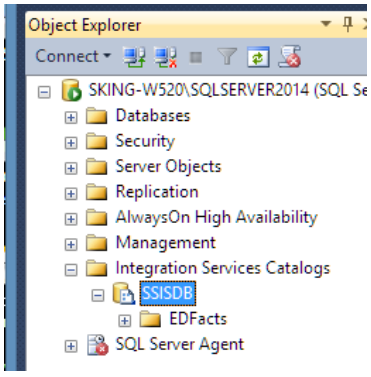
SSISDB (SQL Server 2012 +)

In SQL server 2012 and later versions, the preferred approach is to deploy SSIS packages into an SSIS "catalog" database. This is a separate database so there is not a need to store data in the MSDB system database.

SSIS and its catalog may be deployed to a different server than the one upon which stores the EDFacts database. In most instances, the two databases are on the same server.

Create SSISDB catalog

In SSMS, you can see the current Integration Services catalogs in the Object Explorer window.

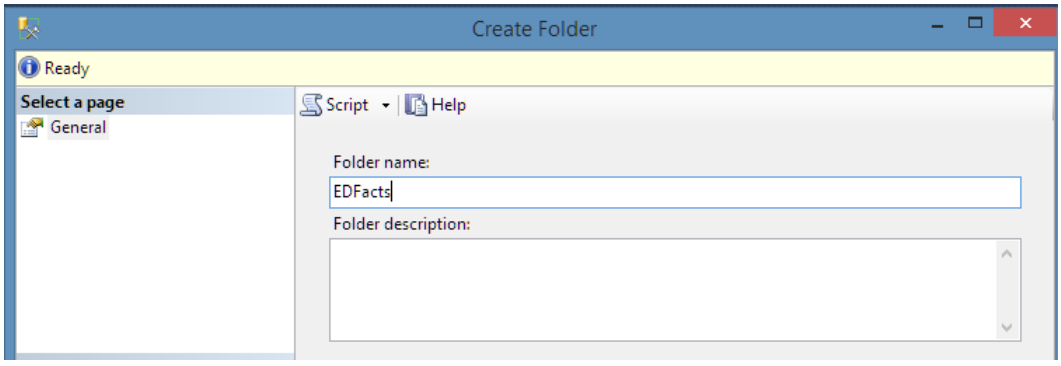


Object Explorer in SSMS

If the SSISDB catalog does not exist, right-click "Integration Services Catalogs" and select the "Create Catalog…" option. Use the default settings.

Create EDFacts folder

To create the EDFacts folder in the SSISDB catalog, right-click the catalog name in the SSMS Object Explorer and select the "Create Folder…" option.



Create Folder window in SSMS Object Explorer

Set the folder name to "EDFacts" being careful to get the spelling and capitalizations exactly as shown.

The projects will get added under the EDFacts folder when the projects are deployed from SSDT.

Create EDFacts_Admin Schema in SSISDB

We will be creating some views and stored procedures in the SSISDB. To keep the ES3 footprint neat and clean, we create a schema in the SSISDB and deploy our vies and procedures there.

```
use SSISDB
create schema EDFacts_Admin
```

This is a one-time operation.

Create Deployed Project and Package Views

The web app will read the deployed projects and packages "on the fly" allowing the user to select a package and run it. So that we don't expose all the deployed SSIS packages, we create views that only show projects and packages that are contained in the EDFacts folder created above.

```
use [SSISDB];
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[ef_Packages]'))
  begin
    drop view [EDFacts_Admin].[ef_Packages];
  end
GO
if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[ef_Projects]'))
  begin
    drop view [EDFacts_Admin].[ef_Projects];
  end
GO

set  ansi_nulls on;
GO
set  quoted_identifier on;
GO

create view EDFacts_Admin.ef_Projects
as
  select '1' as level,
          f.[name] as folderName,
          j.[name] as projectName,
          j.project_id as projectIdentifier ,
          isnull (nullif(j.[description],''), j.[name]) as projectDescription
   from catalog.folders f join [catalog].projects j
     on (j.folder_id = f.folder_id)
   where f.[name] = N'EDFacts'
GO
```

```
create view EDFacts_Admin.ef_Packages
as
     select p.package_id as packageIdentifier,
          p.[name] as packageName,
          isnull(nullif(p.[description],''),p.[name]) as description,
         p.package_guid,
          f.folderName,
          f.projectName,
          p.project_id as projectIdentifier
   from [catalog].packages p
        join EDFacts_Admin.ef_Projects f
     on (f.projectIdentifier = p.project_id)
GO
```

This script can be found in TFS in:

\ES3 Web Management App vs2012
\Support SQL
\ef_SSISViews_SSISDB.sql

This script creates two views ef_Projects and ef_Packages. ef_Projects is the list of projects deployed into our EDFacts SSISDB folder created above. ef_Packages is a list of the packages found in the projects in ef_Projects.

Any projects or packages in SSIS that are not in our EDFacts folder cannot be seen by these views, and therefore cannot be seen by the ES3 Web Management App.

Create the EDFacts Shared Solution Jobs category

To keep our jobs separate from other SQL Agent jobs we create a Job category and assign all of our jobs to that category. This is a one-time activity, but it does require system administrator privileges.

```
declare @returnCode int
declare @jobCategories table (
  category_id  int,
  category_type sysname,
  category_name sysname)

insert into @jobCategories (
      category_id,
      category_type,
      category_name)
   exec msdb.dbo.sp_help_category
          @class='JOB',
          @type='LOCAL',
          @suffix=1

set @returnCode = (select count(*)
                     from @jobCategories
                     where category_name = N'EDFacts Shared Solution Jobs')

if @returnCode = 0
  begin
    exec msdb.dbo.sp_add_category
            @class = 'JOB',
            @type = 'LOCAL',
            @name=N'EDFacts Shared Solution Jobs'
  end

select @returnCode
```

This script can be found in TFS in:

\ES3 Web Management App vs2012
\Support SQL
\Add Job Category.sql

If you don't have sufficient rights, have a system administrator run this script on the server where SSIS and the SQL Agent are running.

Create Job Status Views

Job execution status is monitored by creating views against some system views. Our views filter the system views so we only expose jobs in our job category.

The SQL agent is running on the server where the SSISDB catalog lives, so we will create our views in the SSISDB database. We created the EDFacts_Admin schema above.

```
use [SSISDB];
GO

set  ansi_nulls on;
GO
set  quoted_identifier on;
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysCategories]'))
  begin
    drop view [EDFacts_Admin].[efSysCategories];
  end
GO

create view EDFacts_Admin.efSysCategories
as
     select distinct category_id,
                     category_class,
                     category_type,
                     [name]
     from msdb.dbo.syscategories
     where name = N'EDFacts Shared Solution Jobs'
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysJobs]'))
  begin
    drop view [EDFacts_Admin].[efSysJobs];
  end
GO

create view EDFacts_Admin.efSysJobs
as
     select j.job_id,
            j.originating_server_id,
            j.[name],
            j.enabled,
            j.[description],
            j.start_step_id,
            j.category_id,
            suser_sname(j.owner_sid) as job_owner,
```

```
            j.notify_level_eventlog,
            j.notify_level_email,
            j.notify_level_netsend,
            j.notify_level_page,
            j.notify_email_operator_id,
            j.notify_netsend_operator_id,
            j.notify_page_operator_id,
            j.delete_level,
            j.date_created,
            j.date_modified,
            j.version_number
     from msdb.dbo.sysJobs j
          join EDFacts_Admin.efSysCategories c
            on j.category_id = c.category_id
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysJobActivity]'))
  begin
    drop view [EDFacts_Admin].[efSysJobActivity];
  end
GO

create view EDFacts_Admin.efSysJobActivity
as
     select ja.session_id,
            ja.job_id,
            ja.run_requested_date,
            ja.run_requested_source,
            ja.queued_date,
            ja.start_execution_date,
            ja.last_executed_step_id,
            ja.last_executed_step_date,
            ja.stop_execution_date,
            ja.job_history_id,
            ja.next_scheduled_run_date
     from msdb.dbo.sysJobActivity ja
          join EDFacts_Admin.efSysJobs j on ja.job_id = j.job_id
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysJobHistory]'))
  begin
    drop view [EDFacts_Admin].[efSysJobHistory];
  end
GO

create view EDFacts_Admin.efSysJobHistory
as
     select jh.instance_id,
            jh.job_id,
            jh.step_id,
            jh.step_name,
            jh.sql_message_id,
            jh.sql_severity,
            jh.[message],
            jh.run_status,
            jh.run_date,
            jh.run_time,
            jh.run_duration,
            jh.operator_id_emailed,
            jh.operator_id_netsent,
            jh.operator_id_paged,
            jh.retries_attempted,
            jh.[server]
     from msdb.dbo.sysJobHistory jh
          join EDFacts_Admin.efSysJobs j on jh.job_id = j.job_id
```

```
GO
```

This script is in TFS in

\ES3 Web Management app vs2012
\SupportSQL
\WebUISysViewsSSISDB.sql

This script creates four views in the SSISDB.EDFacts_Admin schema: efSysCategories, efSysJobs, efSysJobActivity, and efSysJobHistory. These query from and mirror the structure of msdb.dbo.sysCategories, msdb.dbo.sysJobs, msdb.dbo.sysJobActivity, and msdb.dbo.sysJobHistory, respectively. These views filter the results to only those jobs in our job category. Again, we do not expose any other SQL Server Agent jobs through our ES3 Web Management app.

Job Creation Stored Procedure

ES3 uses SQL Agent to actually run the SSIS packages. The SQL agent must be running on the server where the SSIS packages are deployed – which may be different than where the EDFacts database resides.

The "On the Fly Job Creation" stored procedure used by the ES3 Web Management app uses the following steps to execute a package:

1. Delete existing jobs named the same as the selected package (left over from previous run of this package).
2. Create a job named the same as our package
3. Add a job step to run the SSIS package
4. Add the job to the current server (the one running SSIS)
5. Schedule the job to run right away

The script is in TFS in

\ES3 Web Management app vs2012
\SupportSQL
\ef_sp_OnTheflyJobCreationSSISDB.sql

MSDB (SQL Server 2008r2)

Create EDFacts folder

Create Deployed Project Folder and Package Views

The web app will read the deployed project folderss and packages "on the fly" allowing the user to select a package and run it. So that we don't expose all the deployed SSIS packages, we create views that only show project folders and packages that are contained in the EDFacts folder created above.

```
use [EDFacts];
GO
```

```
if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[ef_Packages]'))
  begin
    drop view [EDFacts_Admin].[ef_Packages];
  end
GO
if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[ef_ProjectFolders]'))
  begin
    drop view [EDFacts_Admin].[ef_ProjectFolders];
  end
GO

set  ansi_nulls on;
GO
set  quoted_identifier on;
GO

create view EDFacts_Admin.ef_ProjectFolders
as
  with folderTree (level,
                   folderId,
                   parentFolderId,
                   parentFolderName,
                   folderName)
       as (   select 0 as level,
                     folderid,
                     parentfolderid,
                     cast ('root' as sysname)
                       collate SQL_Latin1_General_CP1_CI_AS
                         as parentfolderName,
                     foldername
              from msdb.dbo.sysssispackagefolders
              where foldername = N'EDFacts'
           union all
              select p.level + 1 as level,
                     f.folderid,
                     f.parentfolderid,
                     cast(p.folderName as sysname)
                       collate SQL_Latin1_General_CP1_CI_AS
                         as parnetFolderName,
                     f.foldername
              from msdb.dbo.sysssispackagefolders f
                   inner join folderTree p
                     on f.parentfolderid = p.folderid)
     select level,
            folderId,
            parentFolderId,
            parentfolderName,
            folderName
     from folderTree
GO

create view EDFacts_Admin.ef_Packages
as
     select p.[name] as packageName,
            p.id,
            isnull(nullif(p.[description],' '),p.[name]) as description,
            p.folderid,
            f.folderName

     from msdb.dbo.sysSSISPackages p
          join EDFacts_Admin.ef_ProjectFolders f
            on p.folderid = f.folderid
```

This script can be found in TFS in:

\ES3 Web Management App vs2012
\Support SQL
\ef_SSISViews_2008.sql

This script creates two views ef_ProjectFolders and ef_Packages. ef_ProjectFolders is the list of project folders deployed into our EDFacts MSDB folder created above. ef_Packages is a list of the packages found in the folders in ef_ProjectFolders.

Any projects or packages in SSIS that are not in our EDFacts folder cannot be seen by these views, and therefore cannot be seen by the ES3 Web Management App.

Create the EDFacts Shared Solution Jobs category

To keep our jobs separate from other SQL Agent jobs we create a Job category and assign all of our jobs to that category. This is a one-time activity, but it does require system administrator privileges.

```
declare @returnCode int
declare @jobCategories table (
  category_id  int,
  category_type sysname,
  category_name sysname)

insert into @jobCategories (
      category_id,
      category_type,
      category_name)
   exec msdb.dbo.sp_help_category
          @class='JOB',
          @type='LOCAL',
          @suffix=1

set @returnCode = (select count(*)
                     from @jobCategories
                     where category_name = N'EDFacts Shared Solution Jobs')

if @returnCode = 0
  begin
    exec msdb.dbo.sp_add_category
            @class = 'JOB',
            @type = 'LOCAL',
            @name=N'EDFacts Shared Solution Jobs'
  end

select @returnCode
```

This script can be found in TFS in:

\ES3 Web Management App vs2012
\Support SQL
\Add Job Category.sql

If you don't have sufficient rights, have a system administrator run this script on the server where SSIS and the SQL Agent are running.

Create Job Status Views

Job execution status is monitored by creating views against some system views. Our views filter the system views so we only expose jobs in our job category.

```
use [EDFacts];
GO

set  ansi_nulls on;
GO
set  quoted_identifier on;
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysCategories]'))
  begin
    drop view [EDFacts_Admin].[efSysCategories];
  end
GO

create view EDFacts_Admin.efSysCategories
as
     select distinct category_id,
                     category_class,
                     category_type,
                     [name]
     from msdb.dbo.syscategories
     where name = N'EDFacts Shared Solution Jobs'
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysJobs]'))
  begin
    drop view [EDFacts_Admin].[efSysJobs];
  end
GO

create view EDFacts_Admin.efSysJobs
as
     select j.job_id,
            j.originating_server_id,
            j.[name],
            j.enabled,
            j.[description],
            j.start_step_id,
            j.category_id,
            j.owner_sid,
            j.notify_level_eventlog,
            j.notify_level_email,
            j.notify_level_netsend,
            j.notify_level_page,
            j.notify_email_operator_id,
            j.notify_netsend_operator_id,
            j.notify_page_operator_id,
            j.delete_level,
            j.date_created,
            j.date_modified,
            j.version_number
     from msdb.dbo.sysJobs j
          join EDFacts_Admin.efSysCategories c
            on j.category_id = c.category_id
GO

if exists
     (   select *
```

```
        from sys.views
        where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysJobActivity]'))
  begin
    drop view [EDFacts_Admin].[efSysJobActivity];
  end
GO

create view EDFacts_Admin.efSysJobActivity
as
     select ja.session_id,
            ja.job_id,
            ja.run_requested_date,
            ja.run_requested_source,
            ja.queued_date,
            ja.start_execution_date,
            ja.last_executed_step_id,
            ja.last_executed_step_date,
            ja.stop_execution_date,
            ja.job_history_id,
            ja.next_scheduled_run_date
     from msdb.dbo.sysJobActivity ja
          join EDFacts_Admin.efSysJobs j on ja.job_id = j.job_id
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysJobHistory]'))
  begin
    drop view [EDFacts_Admin].[efSysJobHistory];
  end
GO

create view EDFacts_Admin.efSysJobHistory
as
     select jh.instance_id,
            jh.job_id,
            jh.step_id,
            jh.step_name,
            jh.sql_message_id,
            jh.sql_severity,
            jh.[message],
            jh.run_status,
            jh.run_date,
            jh.run_time,
            jh.run_duration,
            jh.operator_id_emailed,
            jh.operator_id_netsent,
            jh.operator_id_paged,
            jh.retries_attempted,
            jh.[server]
     from msdb.dbo.sysJobHistory jh
          join EDFacts_Admin.efSysJobs j
            on jh.job_id = j.job_id
GO



if exists
     (   select *
         from sys.views
         where object_id =
           OBJECT_ID (N'[EDFacts_Admin].[efSysSSISPackageFolders]'))
  begin
    drop view [EDFacts_Admin].[efSysSSISPackageFolders];
  end
GO

create view EDFacts_Admin.efSysSSISPackageFolders
as
  with folderTree (folderId,
```

```
                    parentFolderId,
                    folderName)
        as (   select folderid,
                      parentfolderid,
                      foldername
               from msdb.dbo.sysssispackagefolders
               where foldername = N'EDFacts'
            union all
               select f.folderid,
                      f.parentfolderid,
                      f.foldername
               from msdb.dbo.sysssispackagefolders f
                    inner join folderTree p
                      on f.parentfolderid = p.folderid)
     select folderId,
            parentFolderId,
            folderName
     from folderTree
GO

if exists
     (   select *
         from sys.views
         where object_id = OBJECT_ID (N'[EDFacts_Admin].[efSysSSISPackages]'))
  begin
    drop view [EDFacts_Admin].[efSysSSISPackages];
  end
GO

create view EDFacts_Admin.efSysSSISPackages
as
     select p.[name],
            p.id,
            p.[description],
            p.createdate,
            p.folderid,
            p.ownersid,
            p.packagedata,
            p.packageformat,
            p.packagetype,
            p.vermajor,
            p.verminor,
            p.verbuild,
            p.vercomments,
            p.verid,
            p.isencrypted,
            p.readrolesid,
            p.writerolesid
     from msdb.dbo.sysSSISPackages p
          join EDFacts_Admin.efSysSSISPackageFolders f on p.folderid =
f.folderid
GO
```

This script is in TFS in

\ES3 Web Management app vs2012
\SupportSQL
\WebUISysViews.sql

This script creates four views in the EDFacts.EDFacts_Admin schema: efSysCategories, efSysJobs, efSysJobActivity, and efSysJobHistory. These query from and mirror the structure of msdb.dbo.sysCategories, msdb.dbo.sysJobs, msdb.dbo.sysJobActivity, and msdb.dbo.sysJobHistory, respectively. These views filter the results to only those jobs in our job category. Again, we do not expose any other SQL Server Agent jobs through our ES3 Web Management app.

Job Creation Stored Procedure

ES3 uses SQL Agent to actually run the SSIS packages. The SQL agent must be running on the server where the SSIS packages are deployed – which may be different than where the EDFacts database resides.

The "On the Fly Job Creation" stored procedure used by the ES3 Web Management app uses the following steps to execute a package:

1. Delete existing jobs named the same as the selected package (left over from previous run of this package).
2. Create a job named the same as our package
3. Add a job step to run the SSIS package
4. Add the job to the current server (the one running SSIS)
5. Schedule the job to run right away

The script is in TFS in

\ES3 Web Management app vs2012
\SupportSQL
\ef_sp_OnTheflyJobCreation.sql

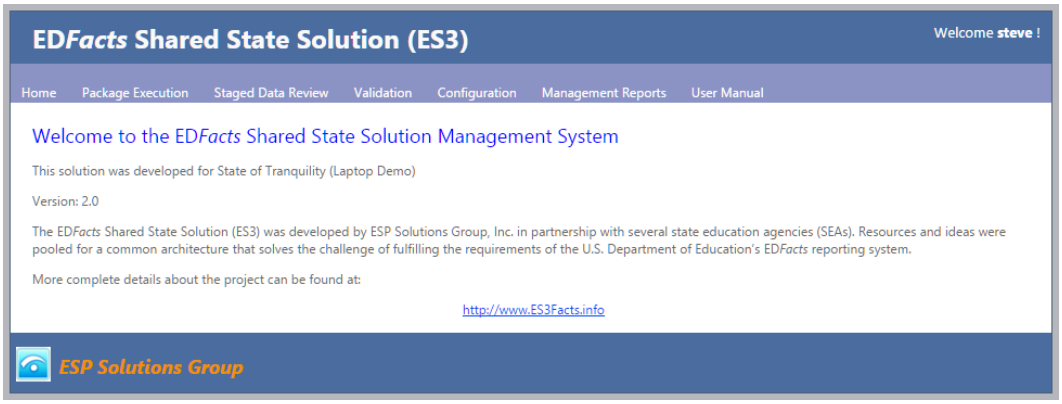
Access to an Email Server for Notification

The ES3 system notifies various parties at the conclusion of package execution through email. The solution will need the name of an email server that can send these messages.

We also will need to identify an email account that can be the FROM: line on the emails. This can be an unmonitored address, i.e. is NOT expected to receive any email.

Mail Server address:	Mail.state.gov
Enable SSL	Yes/No
Windows Authentication	Yes/No
Email From: Account	EDFacts@EDU.state.gov

ASP.NET Web Forms Application



There are five core functions in version 2 of the ES3 web application as represented by the top level menu options:

- SSIS Package Execution
- Staged Data Review and Edit
- Data Validation Reports
- System Configuration
- Management Reports

The bottom four are straight forward Web forms that allow the user to review and, in some cases, edit data for ES3. Only tables within the ES3 solution are exposed in the web application. That is, the web application's data review and edit screens do not require nor expose any access outside of the ED*Facts* database.

SSIS Package Execution Section

The SSIS Package Execution functions are the core components of the web application. It is the SSIS Packages that perform all of the data extract, transform and loading (ETL) operations that move the data from source through staging and into the submission files.

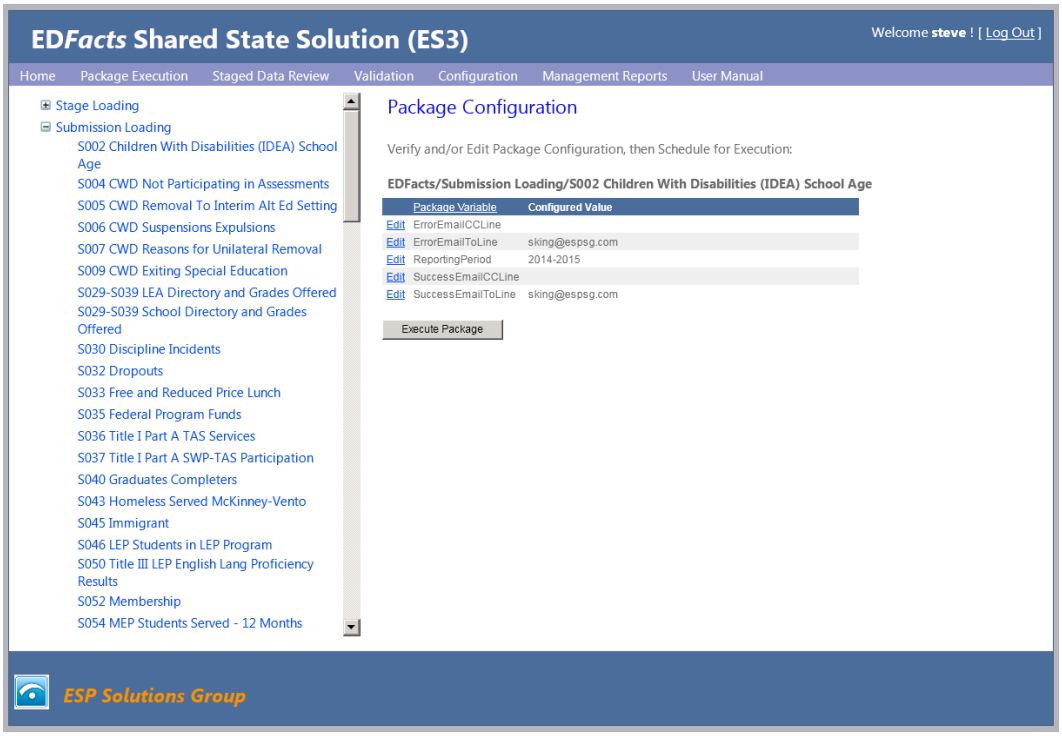
SQL Server used one approach for deploying and storing SSIS packages in SQL Server 2008 R2 and a new approach for later SQL Server versions.

In 2008R2, SSIS packages are stored the MSDB system database and they are configured by storing configuration settings in a special SSIS configurations table. The SSIS packages have to be executed by creating a SQL Agent job to run the package.

In later versions of SSIS, a separate database – SSISDB –is created to store the SSIS packages. Packages now have parameters that can be set at run time. The packages are still executed by creating and firing off a SQL Agent Job.

The two environments can, therefore, be distinguished by the type of repository they use to store the SSISS package: MSDB type or SSISDB type. In either environment the selection, configuration, and triggering of a package is the same.

The tree of available packages is built dynamically by reading a set of views against the SSIS Package repository on the Integration Services server. This occurs when the "Package Execution" menu option is selected.



MSDB Repository Package Configuration

Package names are used to identify the run-time configuration settings in the SSIS_Configuration table that users can set for the selected package.

The stored procedure EDFacts_Admin.ef_Utility_GetConfigurations is the routine that reads and returns the contents from this table.

In this example, the user selected "S052 Membership" and can change the school year to be processed as well as set who should receive package generated emails.

When the [Execute Package] button is selected, the system then:

1) Checks if an existing SQL Job is running for this package
2) If not, deletes any previously defined job
3) Creates a new SQL Agent Job with the same name as the package in an "EDFacts Shared Solution Jobs" job category.

4) Adds a single task of running the selected SSIS package
5) Adds the job to the "Local" server
6) Starts the Job

These steps are handled by the stored procedure EDFacts_Admin.ef_sp_OnTheFlyJobCreation.

The ED*Facts* coordinator can monitor the job execution and status through a couple of the reports under the Management Reports menu options

SSISDB Repository Package Configuration

When using the SSISDB repository, SSIS packages now have parameters and don't use the SSIS_Configuration table. All the package settings are exposed in a couple of system views.

We have an ES3 view against these catalog views that extract the package name and parameter name when the parameter name begins with "ES3_". In the package execution screen we will also replace underscores in the parameter name with spaces: a parameter named "ES3_Success_Email_To_Line" will be shown as "Success Email To Line" in the Web interface.

When the [Execute Package] button is clicked, the system calls a stored procedure to run the package.

Application Folder on IIS Server

Web.Config Edits

Connection Strings

Security and Access

IIS Configuration

Application Pool Account

Create EDFactsAppPool account for the ES3 web app (we can use the same domain account in both test and production)

Set the ES3 web app's web.config file to use Integrated Security (Windows authentication) for our connection to the EDFacts and SSISDB databases

Create SQL Server Credential that is tied to the EDFactsAppPool Account (added this account to the EDFacts group)

Grant the SQLAgentOperator role to the EDFactsAppPool account (by granting to the EDFacts group)

Create SQL Agent Proxy account tied to the EDFacts Credential from above and tied to the SSIS sub system

"EDFacts Shared Solution Jobs" SQL Agent Job Category created

Integrated ASP.NET account Security

Install the Database tables and routine

Windows\Micosoft.NET\Framework\<versionNumber>\aspnet_regsql.exe

Check list