Uploading DB and Web application

Technical Design Report

2014-03-26

Version 1.0

Document Information

Document

|  |  |
| --- | --- |
| **Location:** |  |
| Created by: | Dmitriy Rybalkin |
| Owner: | Dmitriy Rybalkin |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Updated By** | **Reason for Issue** |
| 1.0 | 2014-03-26 | Dmitriy Rybalkin | WEB and DB specification |
|  |  |  |  |
|  |  |  |  |

Reviewed By

|  |  |  |
| --- | --- | --- |
| **Reviewer** | **Business Unit** | **Date** |
|  |  |  |
|  |  |  |
|  |  |  |

Approved By

|  |  |  |
| --- | --- | --- |
| **Approver** | **Business Unit** | **Date** |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[1 Overview 4](#_Toc383611423)

[1.1 Glossary 4](#_Toc383611424)

[2 Implementation Issues 6](#_Toc383611425)

[2.1 Known issues and Usage Restrictions 6](#_Toc383611426)

[2.2 Compliance and Standards Issues 6](#_Toc383611427)

[3 Technical Design Summary 7](#_Toc383611428)

[3.1 Inter-component Dependencies 7](#_Toc383611429)

[3.2 Component Layering 7](#_Toc383611430)

[3.2.1 Components general specification 8](#_Toc383611431)

[4 Configuration Summary 10](#_Toc383611432)

[5 Database 11](#_Toc383611433)

[5.1 Tables 11](#_Toc383611434)

[5.2 Stored Procedures 13](#_Toc383611435)

[5.3 Backup and Restore policy 14](#_Toc383611436)

[6 Installation 15](#_Toc383611437)

[7 System Requirements 16](#_Toc383611438)

[Record of Review 18](#_Toc383611440)

Table of Tables

Table 1 : Glossary of Terms 5

Table 2 : List of Known Issues and Restrictions 6

Table 3 : Summary of Dependencies 7

Table 4 : Architectural Structure 8

Table 5 : Component general specification 9

Table 6 : Configuration Summary 10

Table 7 : Review Record 18

1. Overview

This document provides an overview of the technical design for the integral system for automation of daily and primary data uploading procedures. The current system provides 2 main functions implemented in namespace UploadingData.Web:

1. Uploading primary file to table PRIMARY\_LOAD of DB UploadingDB
2. Uploading daily file to table DAILY\_LOAD of DB UploadingDB

The main core subsystems and modules are grouped in 3 layers and compound the framework, known as ‘’Reunion’’:

1. Core Layer:
   1. Core.DAL – Data Access Layer for retrieving/processing/storing data from DBs
2. Business Layer:
   1. BusinessLayer.UploadingData – Include all reusable components for web application (client layer)
3. Client Layer – WEB application being implemented on basis of ASP.NET MVC 4 patterns.
   1. Glossary

|  |  |
| --- | --- |
| **Term** | **Description** |
| GUI | Graphical user interface is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators. |
| MVC | Model–view–controller (MVC) is a software pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. The central component, the model, consists of application data, business rules, logic, and functions. A view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants. The third part, the controller, accepts input and converts it to commands for the model or view.  Although originally developed for desktop computing, Model View Controller has been widely adopted as an architecture for World Wide Web applications in all major programming languages. Several commercial and non-commercial application frameworks have been created that enforce the pattern. These frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server.  Early web MVC frameworks took a thin client approach that placed almost the entire model, view and controller logic on the server. In this approach, the client sends either hyperlink requests or form input to the controller and then receives a complete and updated web page (or other document) from the view; the model exists entirely on the server. As client technologies have matured, frameworks such as JavaScriptMVC and Backbone have been created that allow the MVC components to execute partly on the client (see also AJAX). |
| DB | Is considered as Relational Database and assumed to be one of the following RDBMS (Relational Database Management System) types: MS SQL Server, Oracle, Postgre SQL. |
| Stored Procedure | A stored procedure is a subroutine available to applications that access a relational database system. A stored procedure (sometimes called a proc, sproc, StoPro, StoredProc, sp or SP) is actually stored in the database data dictionary.  Typical use for stored procedures include data validation (integrated into the database) or access control mechanisms. Furthermore, stored procedures can consolidate and centralize logic that was originally implemented in applications. Extensive or complex processing that requires execution of several SQL statements is moved into stored procedures, and all applications call the procedures. One can use nested stored procedures by executing one stored procedure from within another |

Table 1 : Glossary of Terms

1. Implementation Issues

This section provides a summary of all assumptions that have been made when undertaking the technical design.

* 1. Known issues and Usage Restrictions

|  |  |  |
| --- | --- | --- |
| **Assumption** | **Associated Component(s)** | **Doc. Ref(s)** |
| DB UploadingDB is implemented as DB within MS SQL Server 2008 R2 instance | UploadingDB | Please, execute SQL script UploadingDB\_test.sql within MS SQL Server instance in order to create DB with tables and SPs. |

Table 2 : List of Known Issues and Restrictions

* 1. Compliance and Standards Issues

All software components, either client or server side, are designed to meet Task 2 specifications.

1. Technical Design Summary

The majority of the solution is provided by a number of individual components that are assembled together.

* 1. Inter-component Dependencies

This section provides a summary of each component, and their dependencies on other components. The first three columns cross-reference against the appropriate component specifications; the final column identifies which component fulfils that need.

|  |  |  |  |
| --- | --- | --- | --- |
| Component Name | Dependency Type | Dependant Component | Dependency Description |
| Business Layer | Functional | WEB Application | Business Layer modules provide all Business logic and reusable components for WEB Application. |
| Core Layer | Functional | Business Layer | Core Layer modules provide data access logic and exceptions handling methods and classes. |

Table 3 : Summary of Dependencies

* 1. Component Layering

The components shown in the earlier diagram (Ошибка: источник перёкрестной ссылки не найден) provide functionality which are specific to a given layer in an application, or span multiple layers in an application (a remote business service for example, will require an API in the service layer and an adapter implementation in the integration layer). The following table details all these components, and indicates which layers of the application in which they reside.

|  |  |  |  |
| --- | --- | --- | --- |
| Component Name | Layer Name | Main functions | Data source |
| Core.DAL | Core Layer | Data Access Layer for retrieving/processing/storing data from DBs: - Read/write date to DB tables by calling specific SP | DB tables, SPs, UDF |
| BusinessLayer.UploadingData | BusinessLayer | Implements all methods for intercommunication between Core.DAL and web application | Modules of Business and Client Layers |
| WEB-based Application | Client Layer | Application is implemented on basis of MVC 4 patterns and provides GUI controls, validation of input and client layer logic for interacting with Business Layer modules. | Modules of Business Layer |

Table 4 : Architectural Structure

* + 1. Components general specification

The following table reflects general description of each component of the current software system alongside with system requirements, security and configuration items.

|  |  |
| --- | --- |
| **Component name** | **Specification** |
| Business Layer Components | **Description:**  Business Layer components represent business logic of the application, reusable components for client layer components.  **System requirements:**  OS Windows Server 2008, .NET Framework 4.0  **Security:**  Security is maintained by Core Layer components  **Configuration items:**  Only sections of settings.conf or app.conf files that are integrated into Web.config file |
| Core Layer Components | **Description:**  Core Layer components provide methods and functions for access to DB Servers (SQL Server, Oracle).  **System requirements:**  OS Windows Server 2008, .NET Framework 4.0  **Security:**  **Configuration items:** |
| Client Layer Component | **Description:**  Client Layer component is a web-based MVC 4 application that integrates Business Layer components and its corresponding functions and methods. The main logic of client layer is implemented within application controllers that perform validation of User input, initialize classes and call its methods integrated from Business Layer components.  **System requirements:**  OS Windows Server 2008, .NET Framework 4.0, IIS 7.0  **Security:**  Security is maintained by Core Layer components in conjunction with intrinsic MVC techniques and patterns available for views and controllers.  **Configuration items:**  Web.config |

Table 5 : Component general specification

1. Configuration Summary

Client Layer configuration is entirely implemented in Web.config file that is consisted of the sections specified in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| Configuration Item/Section | Mechanism (File, DB table, etc.) | Value Description | Component References |
| <connectionStrings>  <add name="UploadingDB" connectionString="" providerName="System.Data.SqlClient"/>  <add name="Default" connectionString="" providerName="System.Data.SqlClient"/>  </connectionStrings> | Web.config | Connection strings to default and UploadingDB | Core.DAL.dll |

Table 6 : Configuration Summary

1. Database

The main database UploadingDB includes schema Test. Module BusinessLayer.UploadingData.dll entirely interacts with DB objects of schema Test. The next sections briefly specify DB objects of scheme Test only.

* 1. Tables

Table [Test].[PRIMARY\_TABLE] keeps records of all records derived from file of primary and daily loading procedures.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ORDINAL\_POSITION | COLUMN\_NAME | DATA\_TYPE | CHARACTER\_MAXIMUM\_LENGTH | IS\_NULLABLE | DESCRIPTION |
| 1 | RNN | nvarchar | 12 | NO | RNN |
| 2 | IIN\_BIN | nvarchar | 12 | YES | IIN\BIN |
| 3 | TAXPAYER\_TYPE | tinyint | NULL | YES | Company or Individual |
| 4 | RESIDENCY\_TYPE | tinyint | NULL | YES | Resident or Non-resident |
| 5 | ORGANIZATION\_PERSISTANCY | tinyint | NULL | YES | Persistent or non-persistent organization |
| 6 | IB\_TYPE | tinyint | NULL | YES | Individual or non-individual employer |
| 7 | IB\_NAME | nvarchar | 500 | YES | Employer name |
| 8 | INDIVIDUAL\_NAME | nvarchar | 100 | YES | Surname, First Name, Middle Name |
| 9 | LAST\_RESIGN\_DATE | datetime | NULL | YES | Last date when the Taxpayer was subjected to change tax registration |
| 10 | RESIGN\_REASON | tinyint | NULL | YES | Reason for changing tax registration by Taxpayer |
| 11 | INDIVIDUAL\_TYPE | tinyint | NULL | YES | Is Subject is Lawyer or notary |
| 12 | DORMANT\_TAXPAYER\_TYPE | tinyint | NULL | YES | Type of dormant Taxpayer |
| 13 | TAXPAYER\_COMPANY\_TYPE | int | NULL | YES | Type of Taxpaying company |
| 14 | TAXPAYER\_NAME\_BY\_LOCATION | nvarchar | 100 | YES | Taxpayer name by location |

Table [Test].[PRIMARY\_LOAD] keeps records being uploaded from file of primary loading procedure.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ORDINAL\_POSITION | COLUMN\_NAME | DATA\_TYPE | CHARACTER\_MAXIMUM\_LENGTH | IS\_NULLABLE | DESCRIPTION |
| 1 | RNN | nvarchar | 12 | NO | RNN |
| 2 | IIN\_BIN | nvarchar | 12 | YES | IIN\BIN |
| 3 | TAXPAYER\_TYPE | tinyint | NULL | YES | Company or Individual |
| 4 | RESIDENCY\_TYPE | tinyint | NULL | YES | Resident or Non-resident |
| 5 | ORGANIZATION\_PERSISTANCY | tinyint | NULL | YES | Persistent or non-persistent organization |
| 6 | IB\_TYPE | tinyint | NULL | YES | Individual or non-individual employer |
| 7 | IB\_NAME | nvarchar | 500 | YES | Employer name |
| 8 | INDIVIDUAL\_NAME | nvarchar | 100 | YES | Surname, First Name, Middle Name |
| 9 | LAST\_RESIGN\_DATE | datetime | NULL | YES | Last date when the Taxpayer was subjected to change tax registration |
| 10 | RESIGN\_REASON | tinyint | NULL | YES | Reason for changing tax registration by Taxpayer |
| 11 | INDIVIDUAL\_TYPE | tinyint | NULL | YES | Is Subject is Lawyer or notary |
| 12 | DORMANT\_TAXPAYER\_TYPE | tinyint | NULL | YES | Type of dormant Taxpayer |
| 13 | TAXPAYER\_COMPANY\_TYPE | int | NULL | YES | Type of Taxpaying company |
| 14 | TAXPAYER\_NAME\_BY\_LOCATION | nvarchar | 100 | YES | Taxpayer name by location |

Table [Test].[DAILY\_LOAD] keeps records being uploaded from file of daily loading procedure.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ORDINAL\_POSITION | COLUMN\_NAME | DATA\_TYPE | CHARACTER\_MAXIMUM\_LENGTH | IS\_NULLABLE | DESCRIPTION |
| 1 | RNN | nvarchar | 12 | NO | RNN |
| 2 | IIN\_BIN | nvarchar | 12 | YES | IIN\BIN |
| 3 | TAXPAYER\_TYPE | tinyint | NULL | YES | Company or Individual |
| 4 | RESIDENCY\_TYPE | tinyint | NULL | YES | Resident or Non-resident |
| 5 | ORGANIZATION\_PERSISTANCY | tinyint | NULL | YES | Persistent or non-persistent organization |
| 6 | IB\_TYPE | tinyint | NULL | YES | Individual or non-individual employer |
| 7 | IB\_NAME | nvarchar | 500 | YES | Employer name |
| 8 | INDIVIDUAL\_NAME | nvarchar | 100 | YES | Surname, First Name, Middle Name |
| 9 | LAST\_RESIGN\_DATE | datetime | NULL | YES | Last date when the Taxpayer was subjected to change tax registration |
| 10 | RESIGN\_REASON | tinyint | NULL | YES | Reason for changing tax registration by Taxpayer |
| 11 | INDIVIDUAL\_TYPE | tinyint | NULL | YES | Is Subject is Lawyer or notary |
| 12 | DORMANT\_TAXPAYER\_TYPE | tinyint | NULL | YES | Type of dormant Taxpayer |
| 13 | TAXPAYER\_COMPANY\_TYPE | int | NULL | YES | Type of Taxpaying company |
| 14 | TAXPAYER\_NAME\_BY\_LOCATION | nvarchar | 100 | YES | Taxpayer name by location |
| 15 | RECORD\_PROCESSING\_TYPE | tinyint | NULL | YES | Processing types being indicated for any record in PRIMARY\_TABLE |

* 1. Stored Procedures

All DB writing and reading operations of BusinessLayer.UploadingData module are performed via stored procedures of DB scheme Test.

|  |  |  |
| --- | --- | --- |
| SP Name | Parameters | Description |
| sp\_GetDailyFileName | @ErrCode (nvarchar(3), Output),  @SysCode (nvarchar(10), Output),  @ErrDesc (nvarchar(500), Output) | Returns file name for daily uploading procedure from table SYS\_PARAMETERS |
| sp\_GetDailyLoadRecords | @RNN (nvarchar(12), Input);  @ErrCode (nvarchar(3), Output),  @SysCode (nvarchar(10), Output),  @ErrDesc (nvarchar(500), Output) | Returns records from table DAILY\_LOAD being selected with parameter ‘rnn’ |
| sp\_GetPrimaryFileName | @ErrCode (nvarchar(3), Output),  @SysCode (nvarchar(10), Output),  @ErrDesc (nvarchar(500), Output) | Returns file name for primary uploading procedure from table SYS\_PARAMETERS |
| sp\_GetPrimaryLoadRecords | @RNN (nvarchar(12), Input);  @ErrCode (nvarchar(3), Output),  @SysCode (nvarchar(10), Output),  @ErrDesc (nvarchar(500), Output) | Returns records from table PRIMARY\_LOAD being selected with parameter ‘rnn’ |
| sp\_LoadDailyFile | @ErrCode (nvarchar(3), Output),  @SysCode (nvarchar(10), Output),  @ErrDesc (nvarchar(500), Output) | Uploads records from file daily\_load into table DAILY\_LOAD and refines table PRIMARY\_TABLE based on processing types. |
| sp\_LoadPrimaryFile | @ErrCode (nvarchar(3), Output),  @SysCode (nvarchar(10), Output),  @ErrDesc (nvarchar(500), Output) | Uploads records from file primary\_load into table PRIMARY\_LOAD and refines table PRIMARY\_TABLE based on processing types. |

* 1. Backup and Restore policy

Backup and restore policy is entirely implemented as backup and restore policy of database MS SQL Server which procedures are performed on daily basis.

1. Installation

Installation part is consisted of 3 steps which are as follows:

1. Create DB UploadingDB within MS SQL Server 2008 R2 instance by executing script ‘*UploadingDB\_test.sql*’.
2. Install in IIS 7 attached web application deploying package named ‘*UlpoadingData.Web.zip*’ under any convenient site name and application pool credentials.
3. Modify Web.config of installed web application by changing the following sections:
   1. In section <connectionStrings>, with subsection named ‘Test’ define appropriate connection string for previously created UploadingDB
   2. In section <connectionStrings>, with subsection named ‘Default define appropriate connection string for previously created UploadingDB
4. System Requirements

Common system requirements, being valid for all software modules and components are as follows:

MS Windows Server 2008, .NET Framework 4.0, IIS 7.0 for web-based application of the Client Layer.

MS Windows Server 2008, .NET Framework 4.0 for all components of both Business and Core Layers.

1. Record of Review

Technical design and component specification review can be undertaken at any point during a project/iteration lifecycle; some of those reviews will be informal, but the development process includes specific points where formal review must take place. This section documents all reviews, both formal and informal that have been undertaken on the implementation specification.

|  |  |
| --- | --- |
| **Review Date** | **Reviewer(s)** |
|  |  |
|  |  |
|  |  |

Table 7 : Review Record