TSV2 Architecture & Best practices

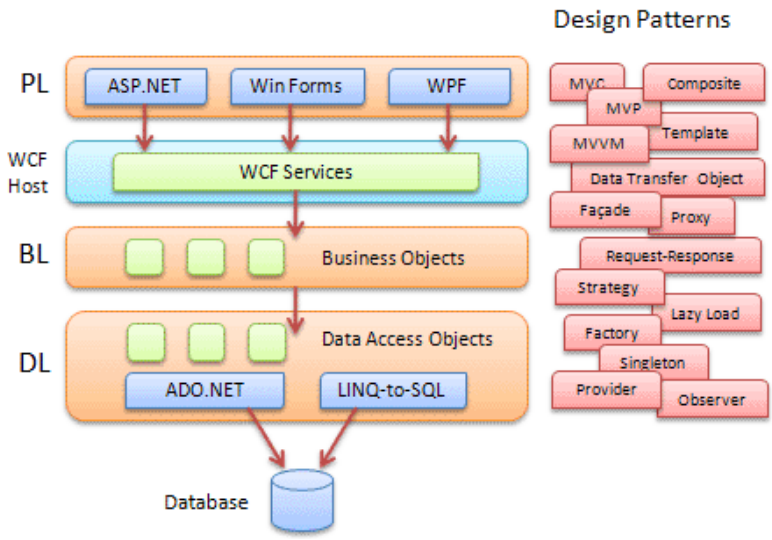
1. OVERVIEW OF TSV2 ARCHITECTURE

Open Destinations is looking at transitioning existing users gradually from TS to TSV2. For this the Travel Studio (windows application) functionality is being migrated to TSV2(Web application), keeping the underlying data model the same.
TSV2 is developed in C# on the.Net framework 3.5.

NEW CHANGE REQUESTS

- Customers who require changes to functionality in TSV2 are taken up as change requests and are codded for in TSV2. This functionality will not be present in TSVB.

.NET 3 Tier Architecture



The PL/GUI: The concern of the Presentation Layer (PL/GUI) is to present information in a consistent and easy-to-understand manner to the end-user. ASP.NET with associatedcoding of client side JAVA and AJAX scripts to the server to enhance the UI experience.

From WCF to BAL: The WCF Service Layer receives messages from the PL/GUI layer. It interprets the message, unpacks the Data Transfer Objects, and coordinates the interaction between Business Objects and Data Access Objects. The Services Layer is also the place where authorization, authentication, data validation, and database transactions are implemented.

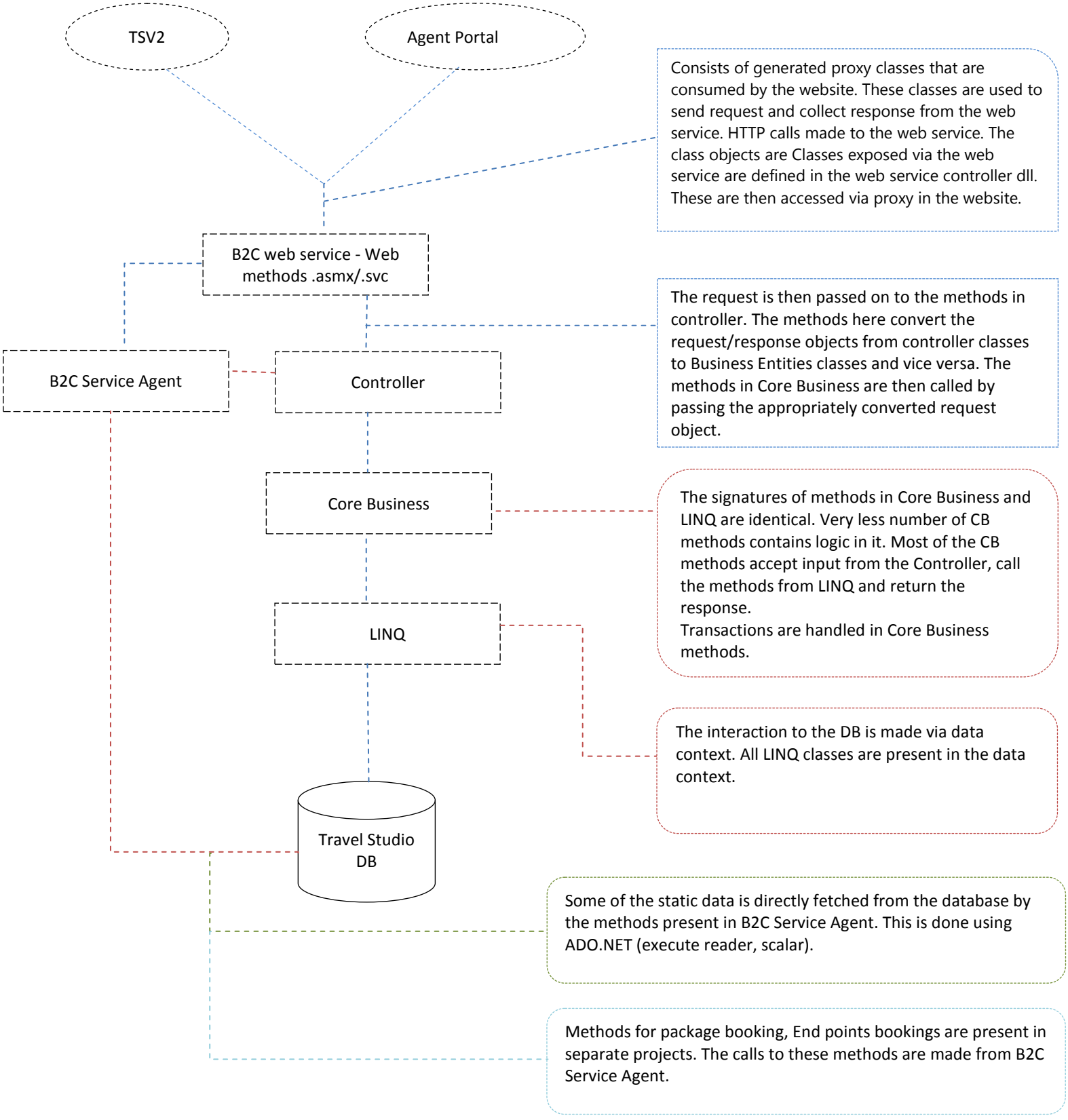
The BAL: The BAL serves as an "intermediary" for data exchanges between the Presentation Layer (GUI) via WCF and the Data Access Layer (DAL).

From BAL to DAL: In fact, Business Objects do not directly interact with the Data Access Layer (DAL). Instead, persistence is handled by dedicated Data Access Objects (another Enterprise Pattern) that extract data from the Business Objects and subsequently store it in the database.

The DAL: The Data Access Layer (DAL) will serves as an "intermediary" for data exchanges between the Business Access Layer (BAL) and the Data Source – Microsoft SQL Server (MS SQL).

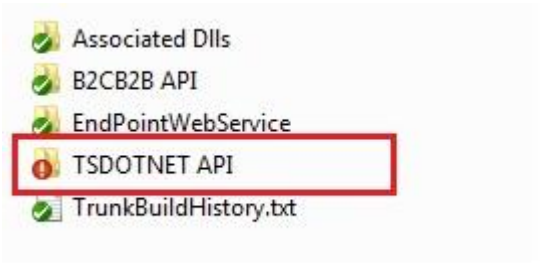
There no direct interactions between the PL and the BAL or DAL. In other words, the UI only interacts with the application via the WCF Service Layer. This is by design. The reason is that security, data validation, and transactions are managed at the Service Layer. The GUI is not allowed to bypass these important functions. Therefore, all communication with the application must go through this single point-of-entry (Cloud Facade) which is hosted under WCF.

TSV2 ARCHITECTURE



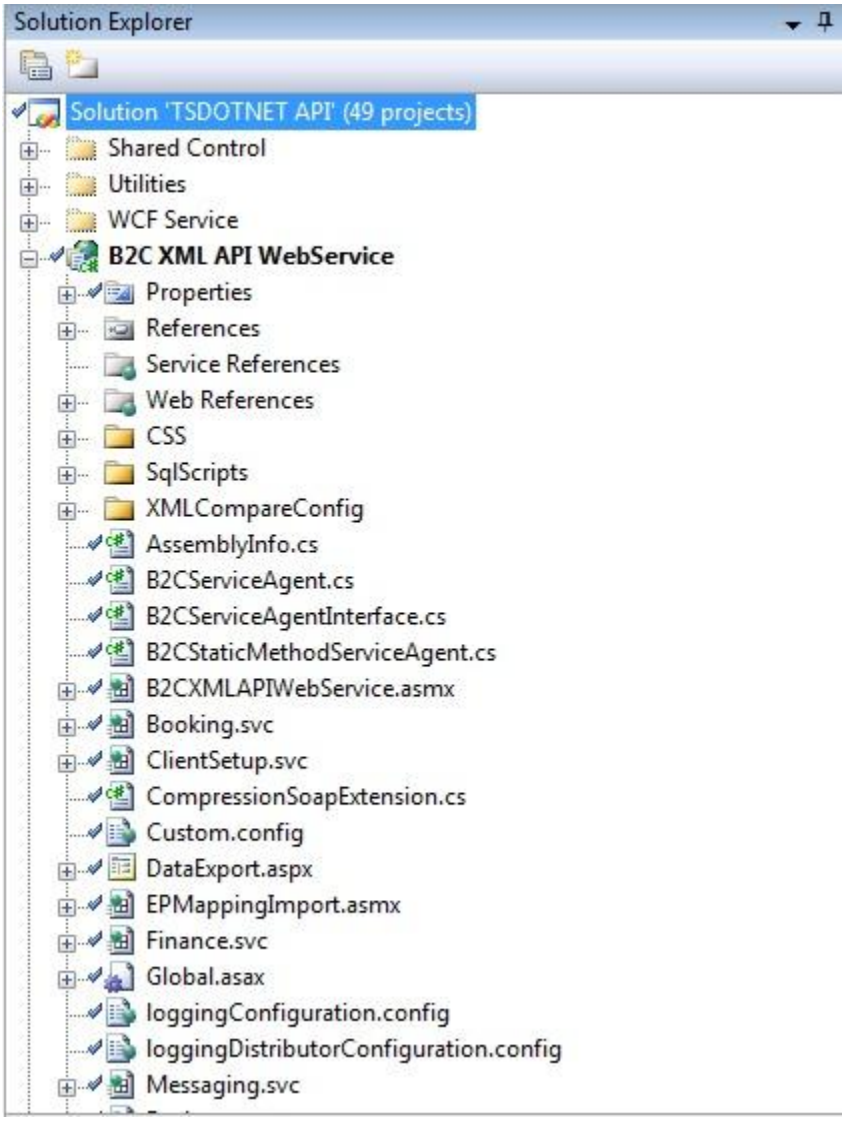
2. TSV2 SOLUTION STRUCTURE

API

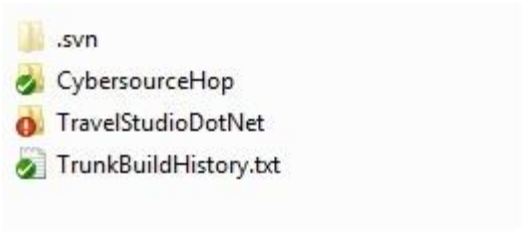


- B2CB2B API contains the solution that is used to expose methods to the clients.
- TSDOTNET API contains the solution that would be consumed byTSV2.

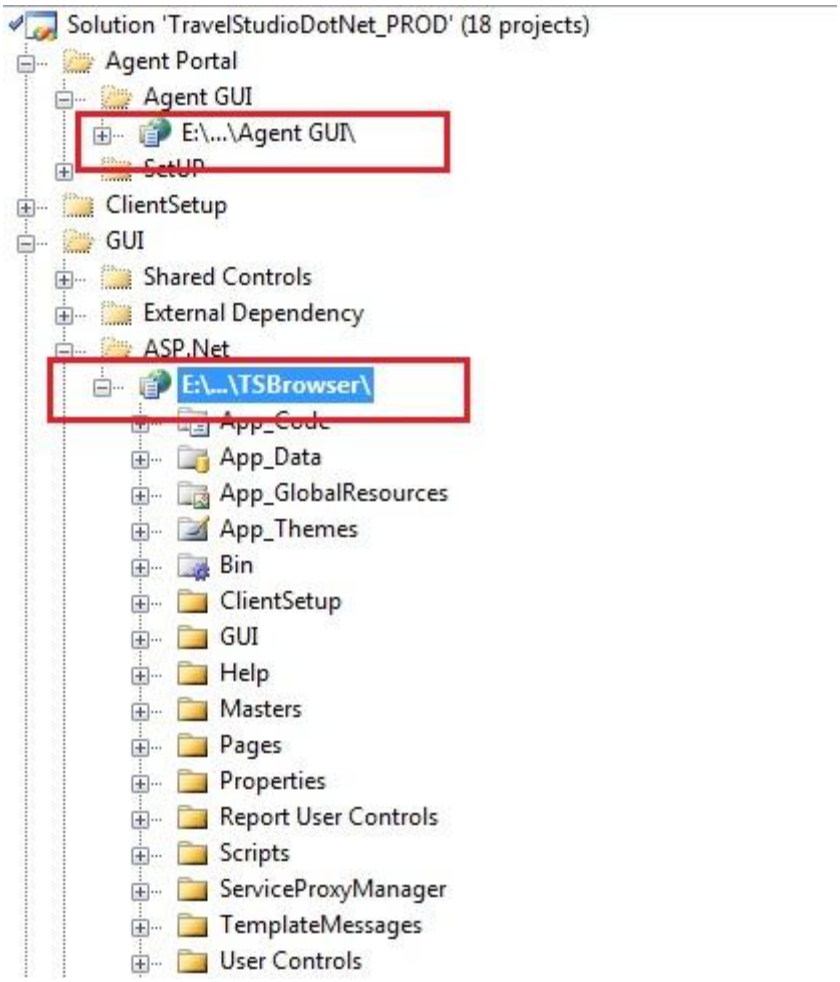
The below image will give the detail structure of TSDOTNET API



TSV2 UI



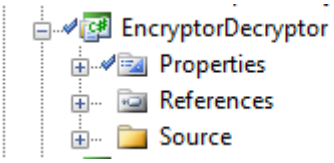
The below image will give the detail structure of TravelStudioDotNet



Common Projects:

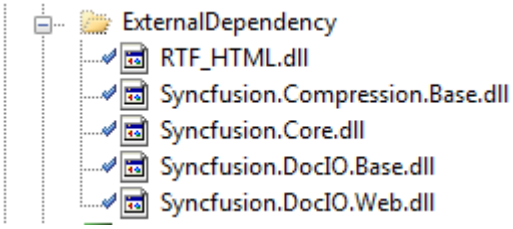
The common projects are as follows:

1. **EncryptorDecryptor**
   This project will be responsible for encryption and decryption of sensitive data like passwords etc. ”Source” folder will have the all classes for Encryption and Decryption.



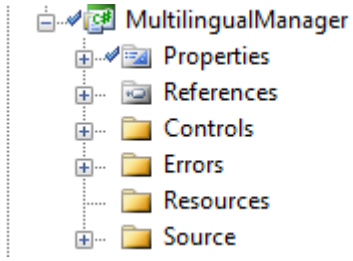
2. External Dependency

This folder will be responsible for to save any Third Party DLLs or some external dll to avoid problem of "Reference not found".



3. MultilingualManager

This project will be responsible for getting the messages in the appropriate language (supported language). 'Controls' folder contains the resource files defining IDs of all Labels with respective texts in different languages. 'Errors' folder will contain the resource files defining the error messages in different languages."Errors " folder is further divided into "Validation" & "Exception" folders for validations and exceptions repectively."Source" folder will have the classes for all multilingual functionality.



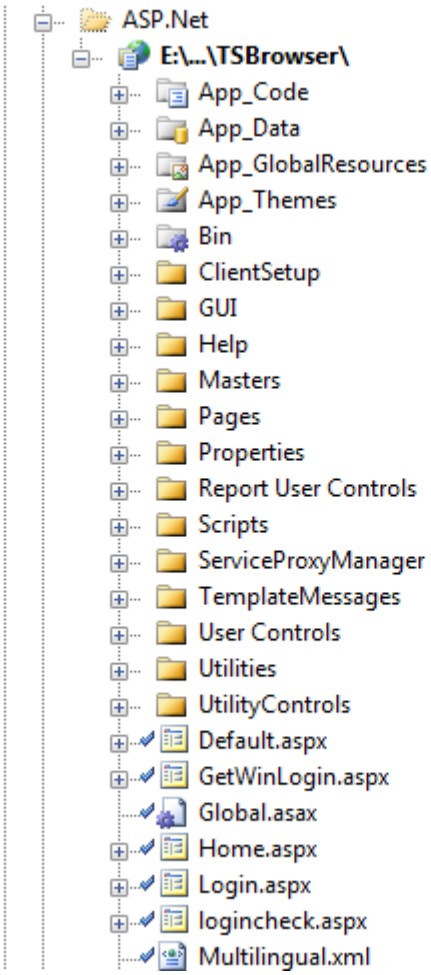
GUI:

This folder will contain the projects that are specific to the GUI Layer i.e. the projects that are required by the UI.

Projects are as follows:

1. TSBrowser Website project

This project will be responsible for interacting with the user. Here is where all the pages and the user interaction form are to be defined.

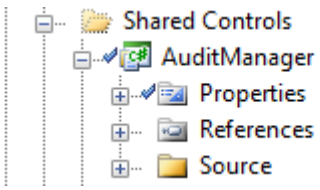


It will in turn contain the following folders:

- App_Code – This will contain all the .cs files needed by the UI.
- App_Data – This will contain all the data needed by the UI.
- App_Themes – This will contain the themes needed by the UI. All the CSS files and images will go under this folder.
- Masters – This will contain all the master pages needed by the application.
- Pages– This will in turn contain subfolders. These sub folders will contain the pages for that functionality, e.g. the Booking folder which will contain all the booking related pages and so on.
- User Controls - This folder will contain all the Web User Controls used by the pages.
- Scripts – This will contain all theJavaScript files used by the UI pages.

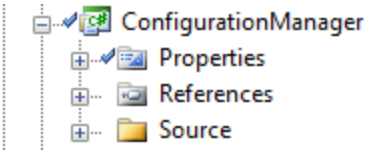
2. AuditManager

This project will be responsible for logging information or errors in files or event logs.”Source” folder will have the classes of audit or logging.



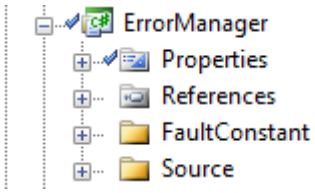
3. ConfigurationManager

This project will be responsible for reading the TS Browser config file for various settings..All ConfigurationManger classes will go to “Source” folder.



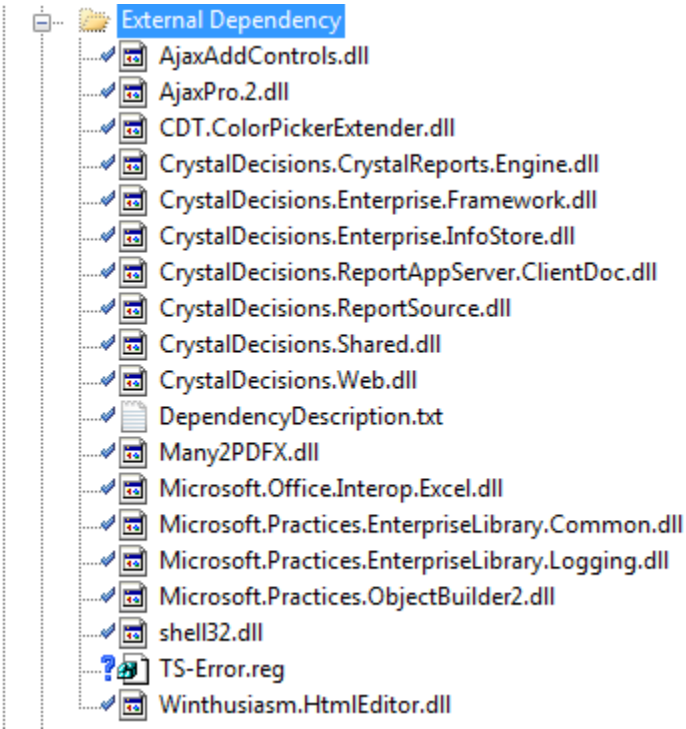
4. ErrorManager

This project will be responsible for logging the errors and exceptions. All ErrorManager classes will go to “Source” folder.



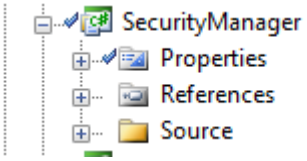
5. External Dependency

This folder will be responsible for to save any Third Party DLLs or some external dll to avoid problem of “Reference not found”.



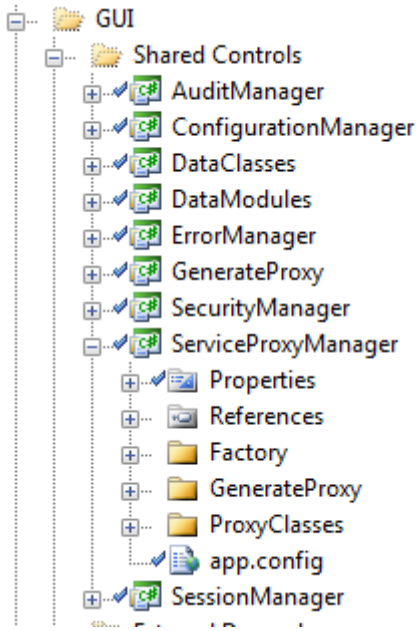
6. **SecurityManager**

This project will be responsible for security of WCF services.All SecurityManager classes will go to “Source” folder.



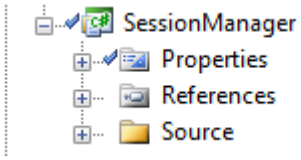
7. ServiceProxyManager

This project will be responsible for interacting with the TSDOTNETAPI.



8. SessionManager

This project will be responsible for maintaining session variables that may be needed to set the session state to exchange data between multiple APSX pages. All SessionManager classes will go to “Source” folder.

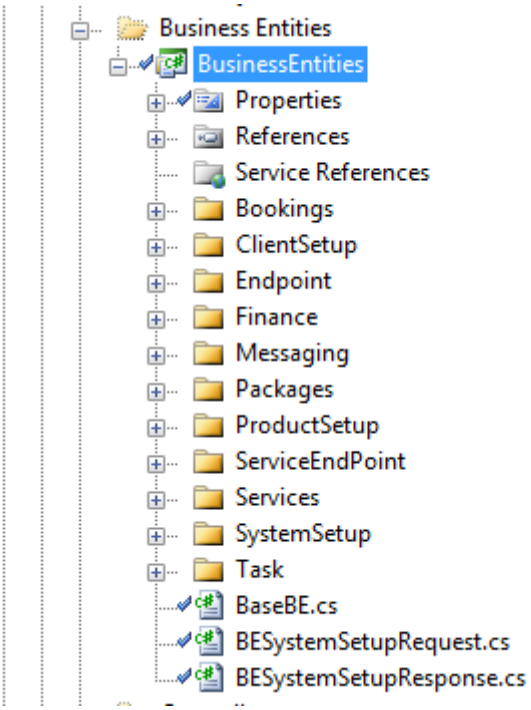


BAL:

This folder contains the projects that perform the business logic. The projects are as follows:

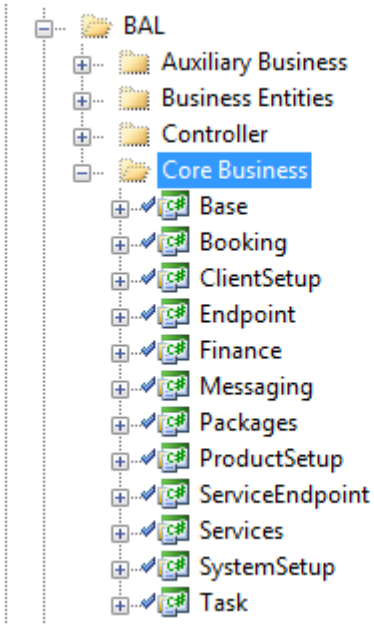
1. Business Entities

This folder will contain all the projects that are needed to decribe all request and response classes.



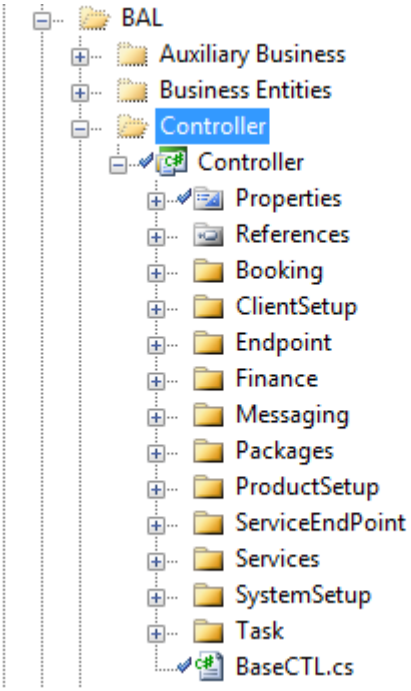
2. Core Business

This folder will contain all the projects that are needed to perform the core business logic. It will in turn contain projects that are defined based on various functionalities exposed to the user through svc files. These folders are shown below in the figure.



3. Controller

This project will be responsible for interaction between WCF and Core Business.

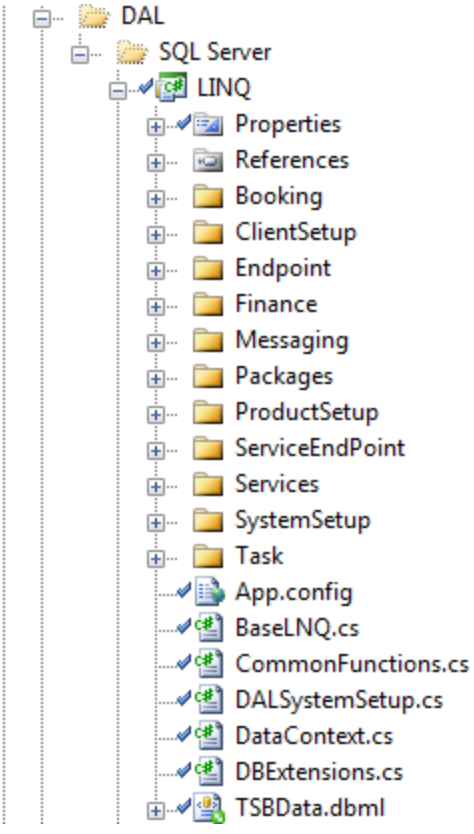


DAL:

This folder contains the projects that perform the data acess layer logic. The projects are as follows

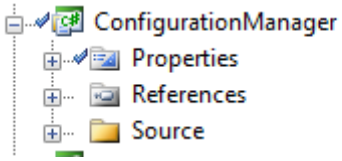
1. LINQ

This project will contain the dbml file needed for accessing the database using LINQ to SQL technology. All LINQ classes for their repective modules are present in the respective folders.



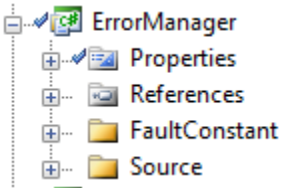
2. ConfigurationManager

This project will be responsible for reading the config file for various settings. All ConfigurationManager classes will go to “Source” folder.



3. ErrorManager

This project will be responsible for logging the errors and exceptions. All ErrorManager classes will go to “Source” folder.

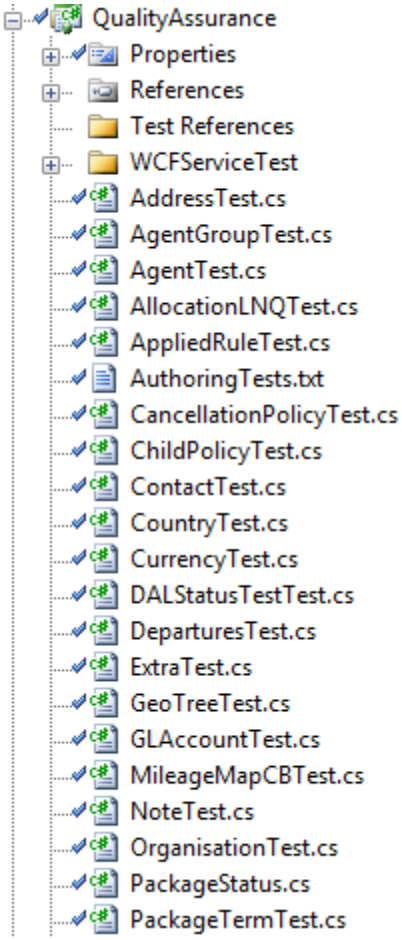


4. External Dependency

This folder will be responsible for to save any Third Party DLLs or some external dll to avoid problem of “Reference not found”.

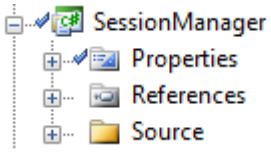
5. QualityAssurance

This project will be responsible for test methods for various functions used in communication layer.



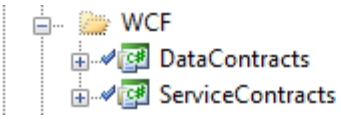
6. SessionManager

This project will be responsible for maintaining session variables that may be needed to set the session state of BAL. All SessionManger classes will go to “Source” folder.



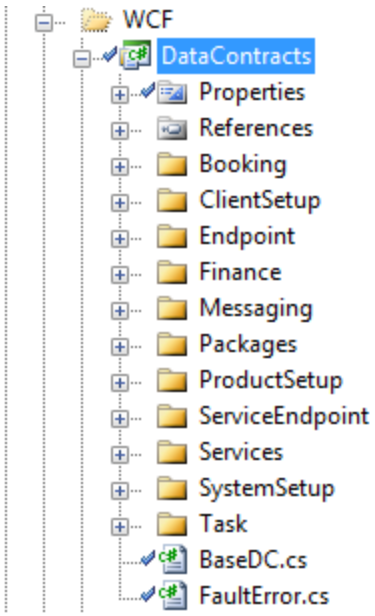
WCF:

This folder will contain classes defining the custom request and response object, also the WCF service contracts classes.



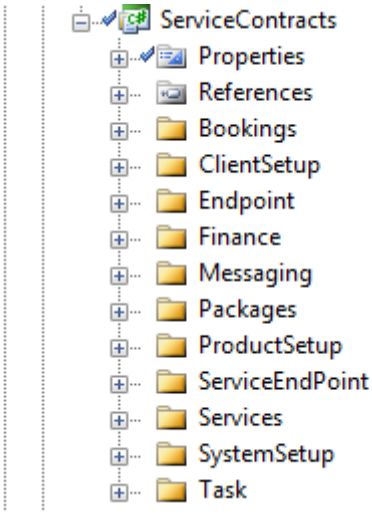
The projects in this folder are as follows:

DataContracts: This proect will contain classes defining the custom request and response objects.

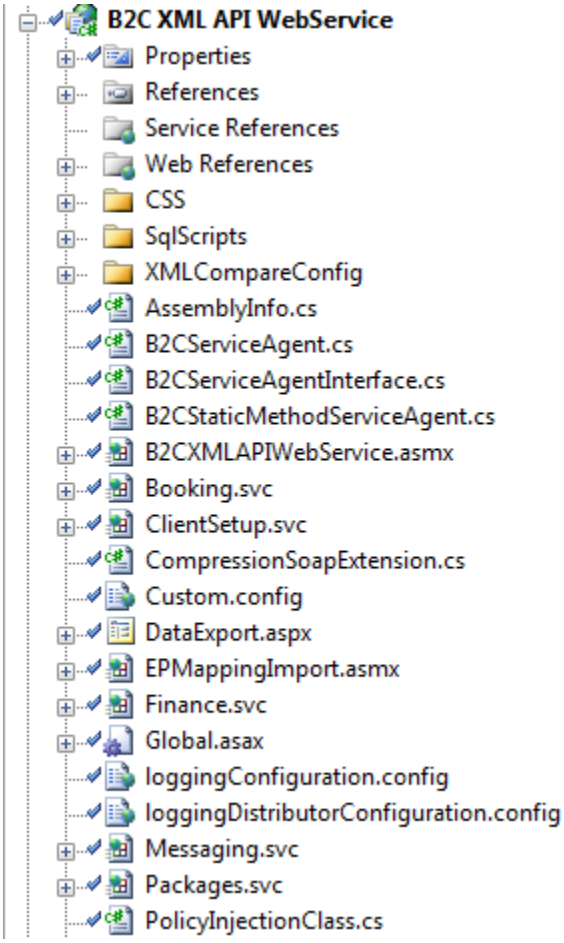


ServiceContracts: This project will contain the interfaces defining the service contracts exposed to the UI.

The following figure shows the various interfaces grouped according to their functionality.



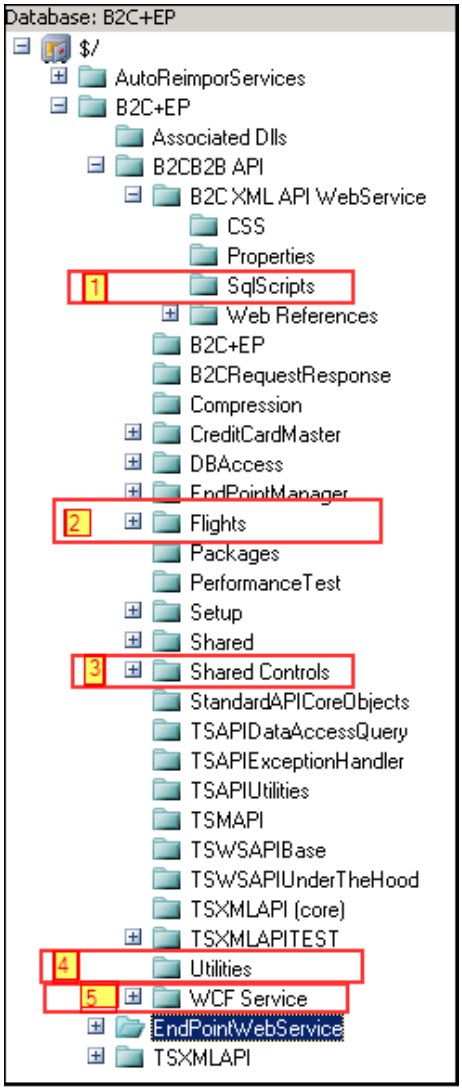
B2C XML API WebService: This will contain the .asmx and all the WCF services (.svc files) exposed to the UI. The various services are grouped into separate .svc files based on functionalities.



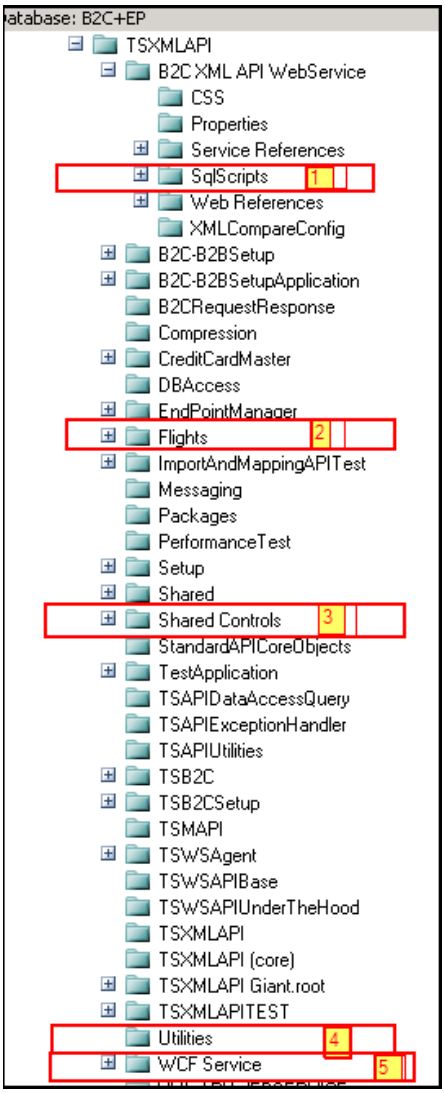
3. SHARING IN B2C/B2B

The following folders and all files in them are shared between B2CWS and TSDOTNETAPI

1) SQL Scripts
2) Flights
3) Shared Folders
4) Utilities
5) WCF



B2C-B2B

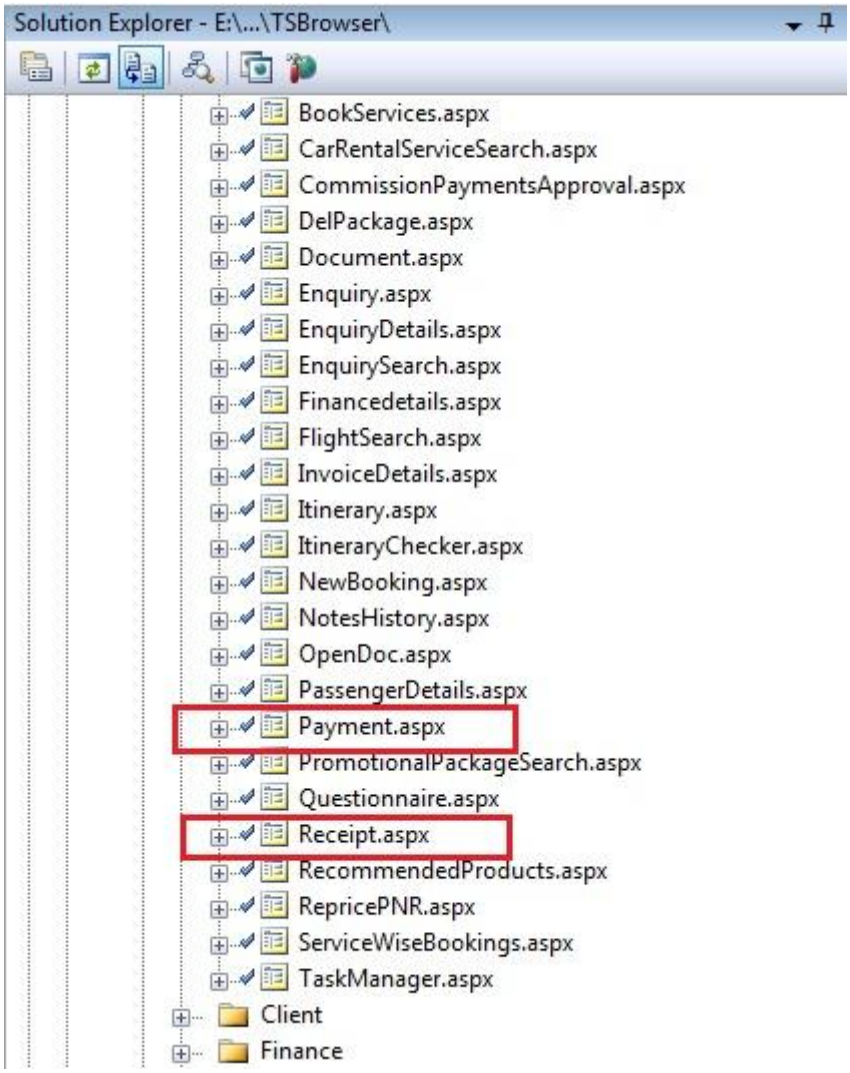


TSDOTNETAPI

If any changes are made to the datacontracts in the WCF folder we need to ensure to handle the same in the B2CB2B solution if they have methods that are using these datacontracts.

If a script has been added the corresponding reference needs to be added manually in the other project as well. Here the default is include. We need to change it to content.

In the TSV2 UI Payment.aspx and Receipt.aspx files are shared between TSV2 and CybersourceHOP Project.



4. IMPORTANT INSTALLATION SETTINGS

Installation Guide for TravelStudio V2.docx

5. DEBUGING A FUNCTIONALITY

6. GUIDELINES - UI, PERFORMANCE, USERCONTROL USAGE

Web guidelines V1.0.docx

7. UI KEEPING PERFORMANCE IN MIND

Travel Studio V2 Optimisation V1.2.doc

8. SESSION, VIEWSTATE MANAGEMENT AND SERVER SIDE VIEWSTATE

TSv2 Performance and LoadBalancer Set

9. CROSS BROWSER COMPATIBILITY

Browser Compatibility.doc