# Economic Data Reporting Application Technical Documentation

## Introduction

The Economic Data Reporting (EDR) application was developed to provide a method for Economic Data Survey participants to fill out their surveys online. It is built using the ASP.NET MVC framework and uses an Oracle database for storage. The application hosts three Crab surveys, three Goa Trawl surveys, a Saltwater Charter survey, and Amendment 91 surveys. The application also has management functions for administrative users to manage surveys and users.

## Components

This application uses the ASP.NET MVC framework that is based on the Model-View-Controller (MVC) architectural pattern.


### Controllers

Controllers are the components that mediate the display of data to the user through views, and saving of data in the models, along with containing business logic. The EDR Application controller hierarchy is shown below.



### Models

Each survey has a model created for it which contains steps. Each step model contains the fields which are filled in by users. The survey model class hierarchy is shown below.



### Views

Each survey step has a corresponding controller action and view. Shared views are used for components which appear repeatedly.

### Databases

The EDR system uses three separate schemas in the Oracle database provided by PSMFC. Schema ‘EDR’ is considered the production database, where all the survey data is eventually migrated, and once there it is accessed by analysts. The ‘EDR\_Stage’ schema is the staging schema. Once a user fills out a survey their data is moved to ‘EDR\_Stage’ and it along with other records is examined by PSFMC staff for any issues. ‘EDR\_App’ is the schema that is directly accessible to the EDR application.

The EDR application utilizes a lightweight ORM called Dapper to access the Oracle database. **Common/Database.cs** contains the methods of connecting and interacting with the database. This file also contains classes that map to the EDR tables in EDR\_APP schema. **Common/Utilities.cs** file is composed of the survey related queries, and **Common/UtilitesA91.cs** file contains A91 Entity related queries. Code lookup methods that are specific to Crab surveys are in **Common/Codes/CrabCodes.cs**. Similarily, code lookup methods for Goa Trawl surveys are located in **Common/Codes/GoaTrawlCodes.cs**.

The EDR\_APP data model is shown below.



## Survey Data Flow

The process below describes how data flows through the system given the most common use case: a user with a single survey.

1. User logs in and begins survey.
2. Data entered by the user is converted into an XML format and stored in the EDR\_APP.User\_Surveys table as the user progresses through each step.
3. Once all the steps are completed, the user reviews the data on the review page and completes the survey by signing the certification page.
4. When a survey is marked as completed the user can no longer make any changes. At the same time the ETL process to move data from EDR\_APP schema to EDR\_Stage schema is triggered. A database script runs and populates the tables in EDR\_STAGE schema with the extracted survey data.
5. Once the survey period is over, data migration scripts are run to move the audited survey data from EDR\_STAGE schema to EDR schema.

### Data Process

User entered survey data is stored in EDR\_APP schema in XML format. This XML data is extracted into the appropriate tables in EDR\_STAGE schema. This happens automatically as a part of the survey completion process. In **Common/Utilities.cs** file, the ToStage method directs the data processing according to the survey type.

The various scripts which move data to EDR\_Stage schema are shown below and they are located in the EDR\_STAGE schema. The permissions between the three schemas are found in Database/EDR\_STAGE/EDR\_Stage\_Permissons.sql file. EDR\_APP has access to the various processes on EDR\_STAGE, and EDR\_STAGE has access to both EDR\_APP and EDR. However, EDR does not have any permission to access either EDR\_APP or EDR\_STAGE.



### Surveys and Processes

The following scripts are utilized when a survey is completed. The processes extract xml data and populate the tables in EDR\_STAGE schema. When a survey is re-certified, previous survey data is wiped and repopulated with updated data. Various supporting functions are located in Forms\_Process package.

|  |  |  |
| --- | --- | --- |
| Survey | To\_Stage Process | To\_Stage Batch Process |
| Amendment 91 | A91\_Process.Amend\_90\_Process | A91\_Process.Amend\_90\_Process\_All |
| Crab Catcher/Processor | Crab\_Process.Crab\_Process | Crab\_process.Crab\_Process\_All |
| Crab Catcher/Vessel | Crab\_Process.Crab\_Process | Crab\_process.Crab\_Process\_All |
| Crab Processor | Crab\_Process.Crab\_Process | Crab\_process.Crab\_Process\_All |
| Goa Trawl Catcher/Processor | Goa\_CP\_Process.Goa\_CP\_Process | Goa\_CP\_Process.Goa\_CP\_Process\_All |
| Goa Trawl Shoreside Processor | Goa\_SP\_Process.Goa\_SP\_Process | Goa\_SP\_Process.Goa\_SP\_Process\_All |
| Goa Trawl Catcher/Vessel | Goa\_CV\_Process.Goa\_CV\_Process | Goa\_CV\_Process.Goa\_CV\_Process\_All |
| Saltwater Charter | Charter\_Process.Charter\_Process | Charter\_Process.Charter\_Process\_All |

When the survey period is over and the data is sufficiently audited, the survey data is migrated to EDR schema. The following list shows the data migration scripts for moving data from EDR\_STAGE to EDR.

|  |  |  |
| --- | --- | --- |
| Survey | To\_Prod Process | To\_Prod Batch Process |
| Amendment 91 | A91\_Process.Move\_To\_Prod | A91\_Process.Move\_All\_To\_Prod |
| Crab Catcher/Processor | Crab\_Process.Move\_Crab\_To\_Prod | Crab\_process.Move\_All\_Crab\_To\_Prod |
| Crab Catcher/Vessel | Crab\_Process.Move\_Crab\_To\_Prod | Crab\_process.Move\_All\_Crab\_To\_Prod |
| Crab Processor | Crab\_Process.Move\_Crab\_To\_Prod | Crab\_process.Move\_All\_Crab\_To\_Prod |
| Goa Trawl Catcher/Processor | Goa\_CP\_Process.Move\_Goa\_To\_Prod | Goa\_CP\_Process.Move\_All\_Goa\_To\_Prod |
| Goa Trawl Shoreside Processor | Goa\_CP\_Process.Move\_Goa\_To\_Prod | Goa\_CP\_Process.Move\_All\_Goa\_To\_Prod |
| Goa Trawl Catcher/Vessel | Goa\_CP\_Process.Move\_Goa\_To\_Prod | Goa\_CP\_Process.Move\_All\_Goa\_To\_Prod |

## Behavior Diagrams

Below are activity and use case diagrams. Activity diagrams are used to describe dynamic aspects of the system utilizing flow charts to represent the flow from one activity to another activity.

In addition, use cases are shown for Amendment 91 to show that some surveys are required while others are not depending on the entity profile information. The use cases also capture the aspect that the entity owner invites skippers and the Skipper survey can be filled out by both the owner and the skipper.









## Development Environment

### Technologies Used

.NET 4.5

ASP.NET MVC 4

Dapper ORM

JQuery

JQuery Validate

Bootstrap.js

Visual Studio 2015

### Deployment

Development is done on developer machines that access the shared development database on an RDI server. When development has ended and we want to deploy to the test server we use Remote Desktop to access the server and manually copy the files over. The same process is used for deploying to production. Deployments are fairly rare so we have not automated this process.
Below are the server addresses and URLs to access these resources. Access is restricted by PSMFC. Contact their IT department to be issued an account.

|  |  |  |
| --- | --- | --- |
|  | TEST | PRODUCTION |
| Server | Canarydev.psmfc.org | canary.psmfc.org |
| URL | http://canarydev.psmfc.org/EDR/ | <https://survey.psmfc.org/> |

### Databases

##### Shared Development Database on RDI server

|  |  |  |  |
| --- | --- | --- | --- |
| Schema | EDR\_APP | EDR\_STAGE | EDR |
| Hostname | por-spdev1.resdat.com | por-spdev1.resdat.com | por-spdev1.resdat.com |
| Port | 1521 | 1521 | 1521 |
| SID | xe | xe | xe |
| Username | EDR\_APP | EDR\_STAGE | EDR |
| Password | EDR\_APP | EDR\_STAGE | EDR |

##### Test Database on Copper (client server)

|  |  |  |  |
| --- | --- | --- | --- |
| Schema | EDR\_APP | EDR\_STAGE | EDR |
| Hostname | copper.psmfc.org | copper.psmfc.org | copper.psmfc.org |
| Port | 1521 | 1521 | 1521 |
| Service name | akfindw2 | akfindw2 | akfindw2 |
| Username | EDR\_APP | EDR\_STAGE | EDR |
| Password | EDR\_APP | p$cific2 | $kfinFish |

##### Production Database on China (client server)

|  |  |  |  |
| --- | --- | --- | --- |
| Schema | EDR\_APP | EDR\_STAGE | EDR |
| Hostname | china.psmfc.org | china.psmfc.org | china.psmfc.org |
| Port | 2045 | 2045 | 2045 |
| Service name | akfindw1 | akfindw1 | akfindw1 |
| Username (schema) | EDR\_APP | EDR\_STAGE | EDR |
| Password |  |  |  |

## Miscellaneous Tasks

### How to load users

User are loaded into the system or created by administrators in the admin section of the application. Every year a list of users with usernames and passwords is generated by PFMFC (Pacifc Stages Marine Fisheries Commission) staff as an excel file. We built a console application called UserImportStatementGenerator which converts the list into a SQL file with insert statements. This file is then run manually on the EDR\_APP schema to load the users.

#### UserImportStatementGenerator

The UserImportStatementGenerator console app is located in /ConsoleApp folder. It reads the user list as an excel file, generates hashed passwords using SHA1, and generates insert statements. Hard coded input and output file locations need to be specified in the program.

### How to add a survey

New surveys may be added in the future. To create them do the following:

1. Create a record in EDR\_APP.SURVEYS table which describes the survey.
2. Create a record in EDR\_APP.ROLES table which describes access to the survey.
3. Users can access multiple surveys (this is the exception). For each user that needs access to a particular survey, create a record in EDR\_APP.SURVEY\_ROLES table. This is handled automatically if done in the administration part of the EDR application. For bulk user loading we do this manually in SQL.
4. Create a model extending **EDR.Common.Survey** which has a name that matches the one in EDR\_APP.SURVEYS for column SURVEY\_CLASS\_NAME. This is used by the application to correctly serialize and deserialize the survey model to XML.
5. Create a controller extending **SurveyController** and a folder to store all the views you will need. The URL path to reach your new controller and its actions must match the SURVEY\_URL\_PATH defined in your EDR\_APP.SURVEYS record.
6. Build out the Model, Controllers, and Views as you would with any normal MVC project. By convention each survey is broken down into steps and each step is its own view.

### Validations

Input validation is done mostly on the client side. EDR app uses jQuery validate for common validations such as telephone number, zip code, and email. Also, there are EDR specific client side validations which can be found in **Scripts/EDR\_Validations.js**. Many of the EDR specific validations deals with allowing 'N/A' into the fields as an alternative to null.

There are also server side validations which check business logic such as whether a skipper has already been invited for a particular vessel survey or if the invitation token is valid for that skipper.