# **Implementing CDR Analysis**

## Prerequisites:

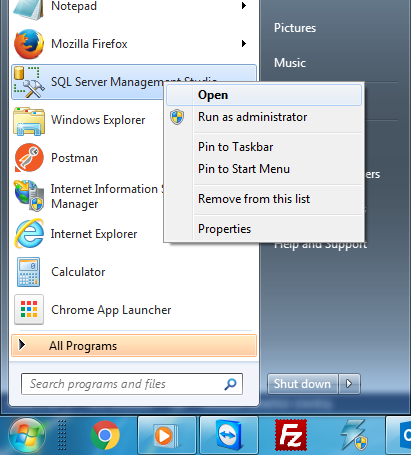
1. **SQL server with analysis services feature installed**
2. **Net framework 4.5**
3. **IIS**

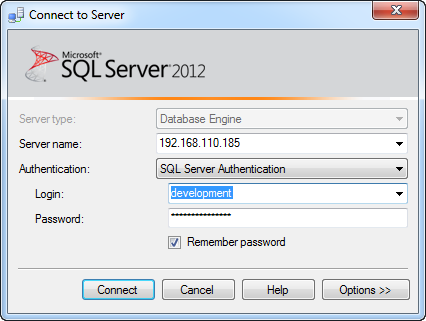
## Steps:

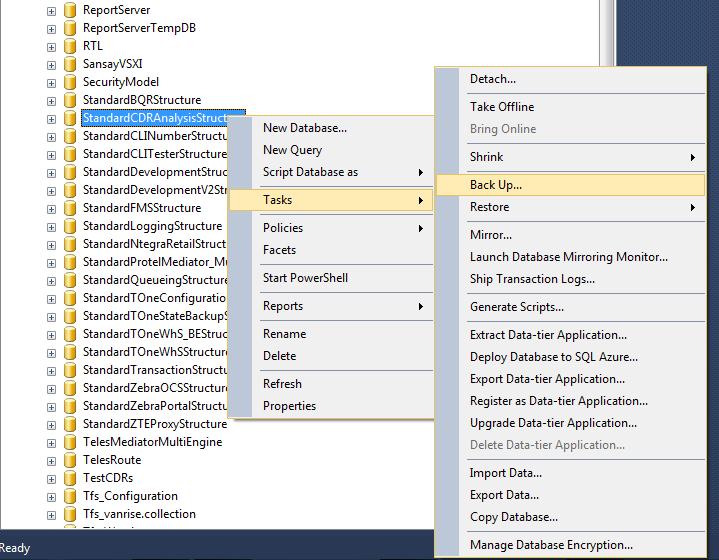
### Taking backup of databases (5)

* Databases to be backed up: StandardCDRAnalysisStructure, StandardLoggingStructure, StandardQueueingStructure, StandardTransactionStructure, StandardConfigurationStructure.
* Steps for backup:

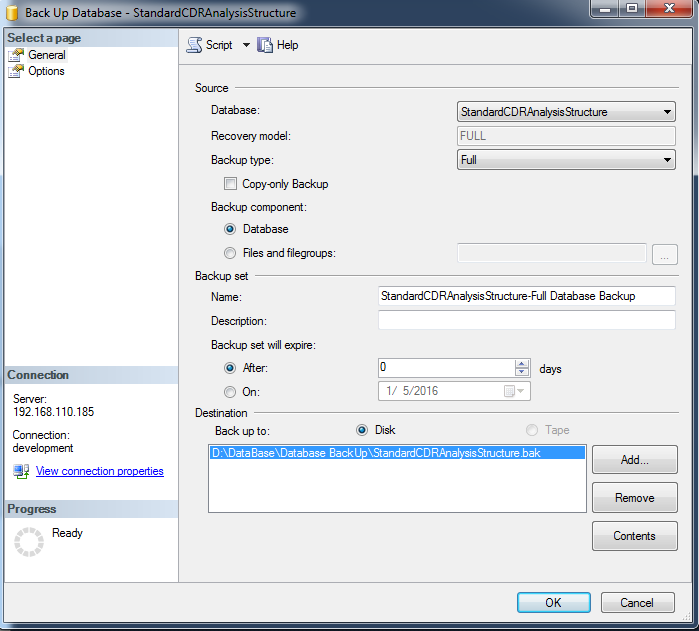
1. Open Microsoft SQL Server Engine.



1. Connect to the instance you want to take backup from.
2. Right click the database you want to back and choose : Tasks>>Back Up …



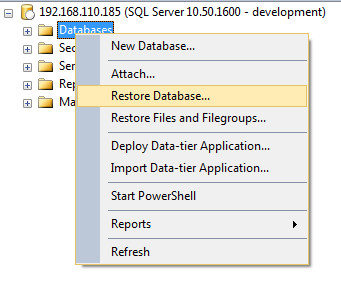
1. Choose destination you want to back your file to.



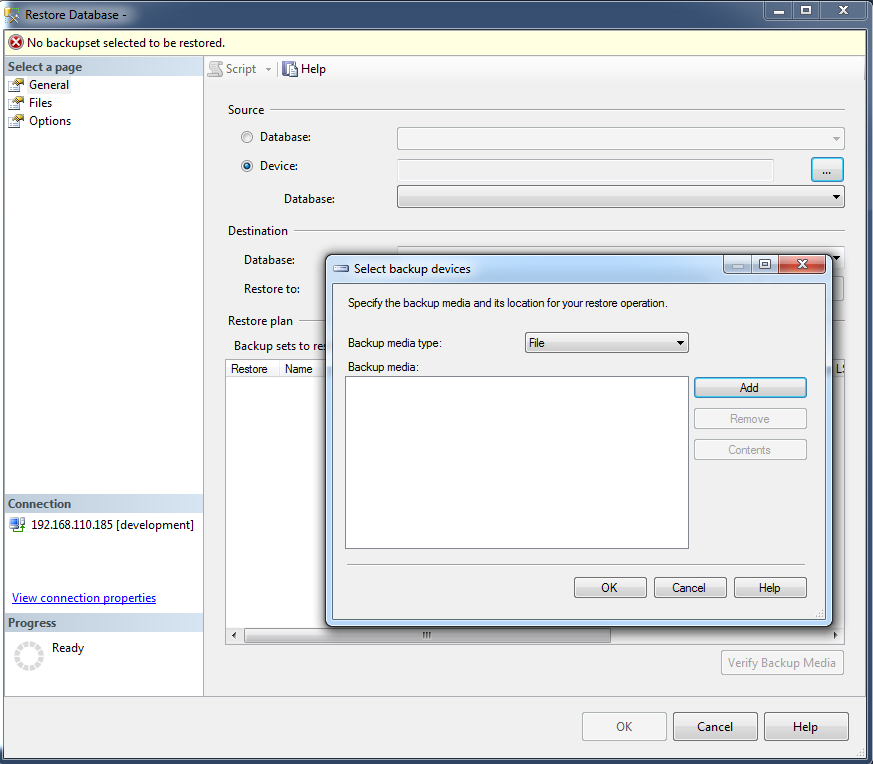
### Restoring backup files of databases (5)

* Steps for Restore:

1. Right click the “Databases” folder and choose : Restore Backup



1. Choose the source to be device and then locate the file you have already backed up.



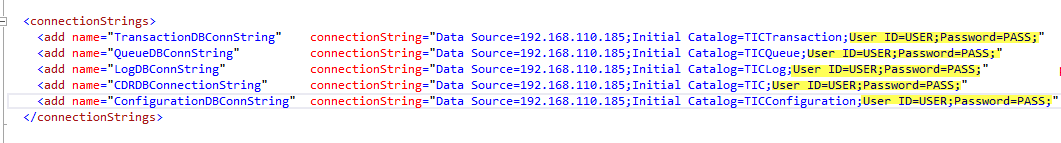
1. Choose the last backup set to be restored, and click ok finally.
2. Apply post scripts for databases restored, each script should be executed in its database.

### Get Latest Published Versions from

* CDR Analysis website new versions will be released under [\\192.168.110.185\Fixes\WebSite\CDRAnalysis\](file:///\\192.168.110.185\Fixes\WebSite\CDRAnalysis\)
* CDR Analysis service new versions will be released under [\\192.168.110.185\Fixes\WebSite\CDRAnalysis\Runtime\](file:///\\192.168.110.185\Fixes\WebSite\CDRAnalysis\Runtime\)

### Update configuration files for both website and runtime

* Config Files for website and runtime are called: Web.config and anrise.Fzero.DevRuntime.exe.config consecutively



* + For Both website and runtime config files you should update the credentials of connection strings to match the credentials you need
  + update below key



* For runtime config
  + check the app setting key BCPDonotDeleteFiles is set to false



* + create a directory for bcp temp files and update below key in app setting section



* + create a directory for BCP Files and update below key in app setting section

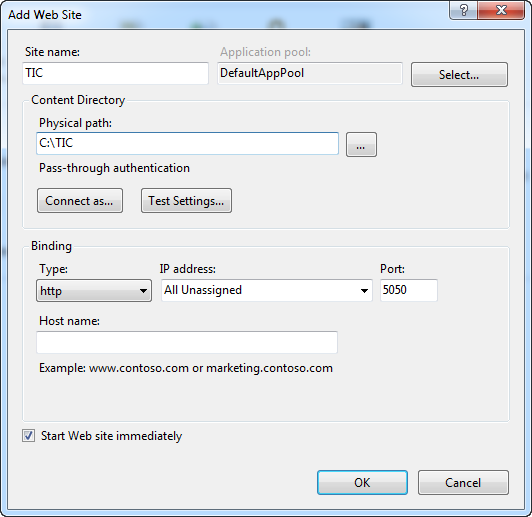


* + check that app setting IsRuntimeService is set to true

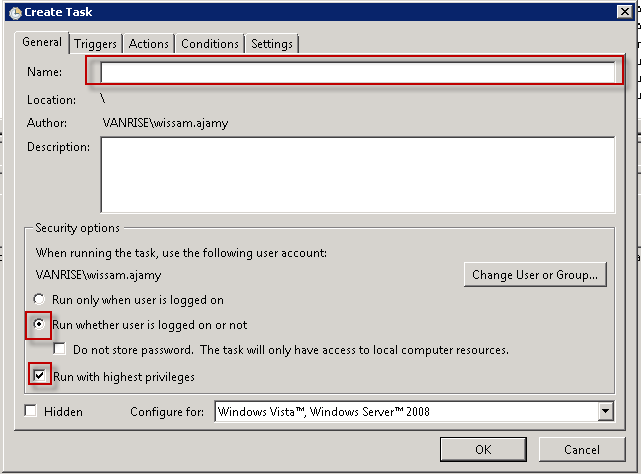


### Publish website and runtime

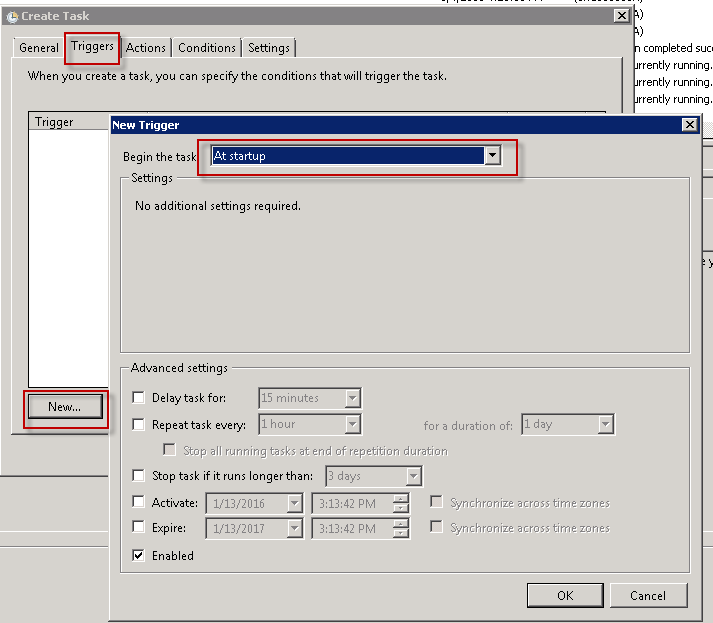
* Publishing Website:



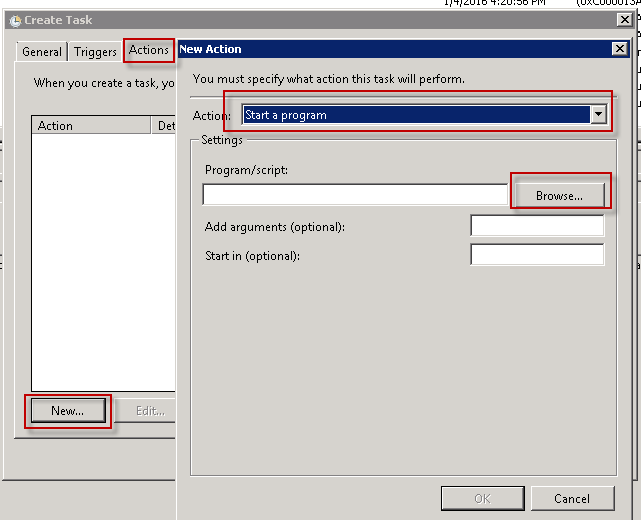
* 1. Click Start, point to Settings, and then click Control Panel.
  2. Double-click Administrative Tools, and then double-click Internet Services Manager.
  3. Click Action, point to New, and then click Web Site.
  4. After the Web Site Creation Wizard starts, click next.
     1. Type Name for the Web site TIC.
     2. Application Pool: choose DefaultAppPool.
        1. .**Net CLR Version: 4.0**
        2. Managed Pipeline Mode: Integrated.
        3. Identity: ApplicationPoolIdentity
     3. Under physical path, either type the path to the folder that is holding the Web site documents or click Browse to select the published version folder
     4. Set the port number 5050, or choose different port if this port is not available
     5. Click Finish.
* Publishing Runtime:
  1. Click Start, type Task Scheduler and press enter.
  2. Choose from actions **Create Task**.
  3. After the Task creation wizard starts.
     1. Type name for the task to be created TIC Runtime, then click ok.



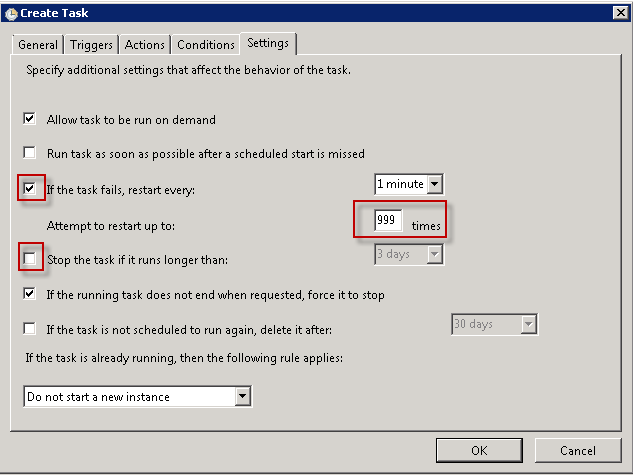
* + 1. Triggers: at start up, then click ok.



* + 1. Actions: start a program, Point to exe of runtime called Vanrise.Fzero.DevRuntime.exe, then click ok.



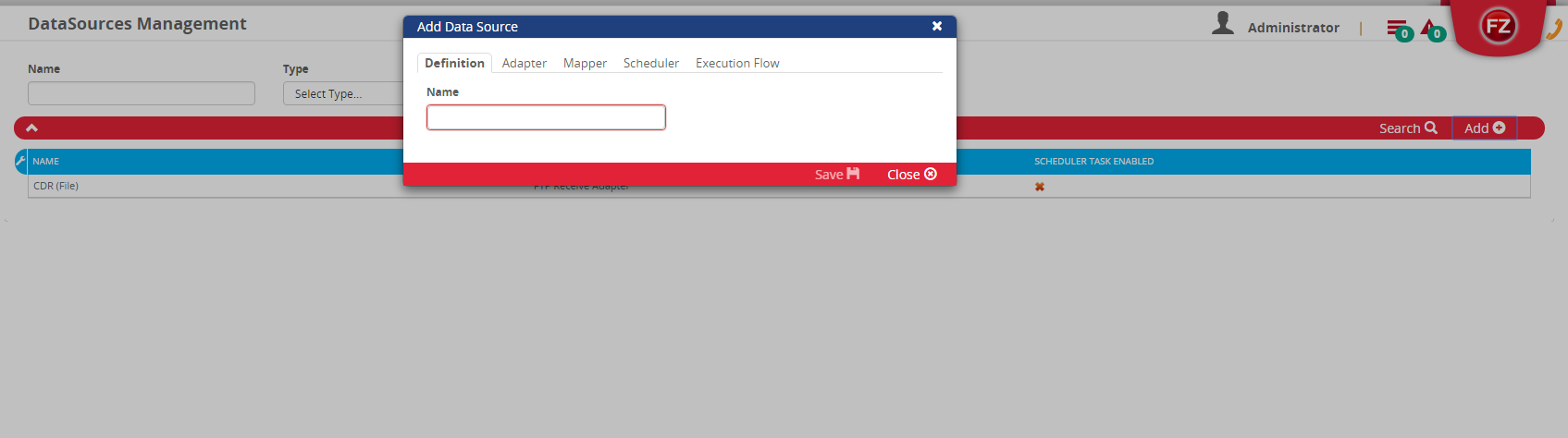
* + 1. Settings: uncheck stop the task, check if the task fails and update number of times, then Finish.



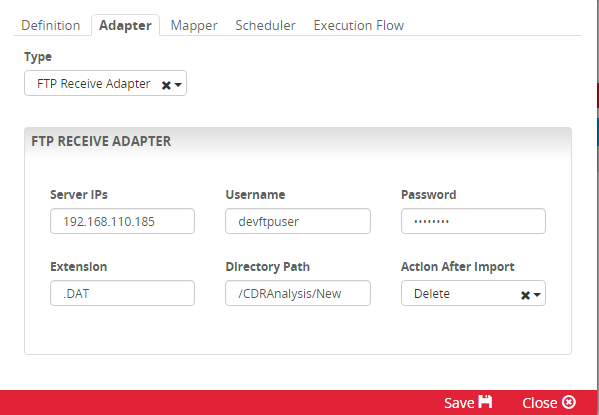
### Create Data sources through application interface

* There is three types of data sources: File, FTP, and SQL Server
* Steps

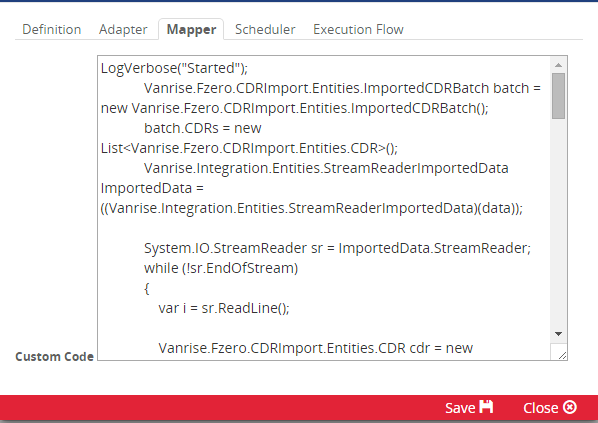
1. Go to Data source management screen and click add



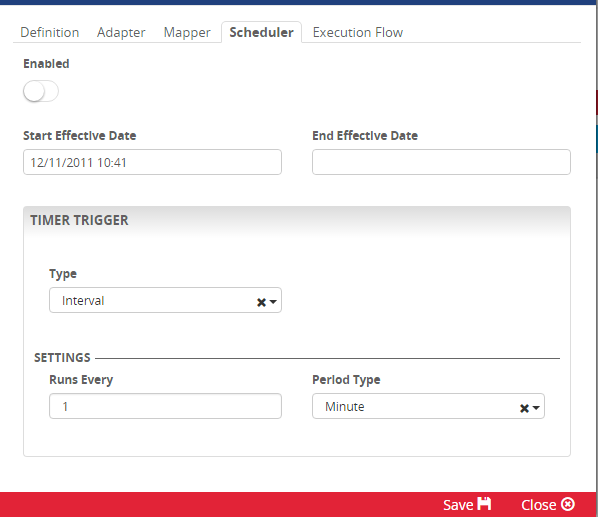
1. Fill all the required Fields for the adapter of data source being created



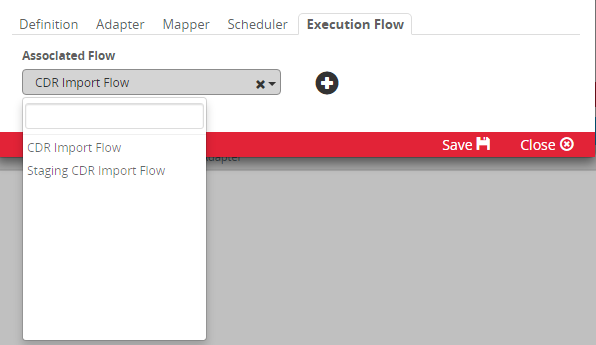
1. Past the Mapper code provided for corresponding data source, find bellow mapper samples:



1. Create Schedule for importing through this data source.



1. Choose the execution flow that you want for the imported CDRs



* Mapper Code

1. File or FTP:

/\*This is a sample code that maps CDRs from a file adapter to the system

There is one section in this file that should be subject to change, other parts should remain the same for the current project. \*/

Vanrise.Fzero.CDRImport.Entities.ImportedCDRBatch batch = new Vanrise.Fzero.CDRImport.Entities.ImportedCDRBatch();

batch.CDRs = new List<Vanrise.Fzero.CDRImport.Entities.CDR>();

Vanrise.Integration.Entities.StreamReaderImportedData ImportedData = ((Vanrise.Integration.Entities.StreamReaderImportedData)(data));

System.IO.StreamReader sr = ImportedData.StreamReader;

while (!sr.EndOfStream)

{

var i = sr.ReadLine();

Vanrise.Fzero.CDRImport.Entities.CDR cdr = new Vanrise.Fzero.CDRImport.Entities.CDR();

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//Start of Dynamic Section

cdr.MSISDN = i.Substring(125, 20).Trim();

cdr.IMSI = i.Substring(125, 20).Trim();

cdr.Destination = i.Substring(198, 20).Trim();

cdr.CallClass = i.Substring(434, 10).Trim();

cdr.SubType = i.Substring(165, 10).Trim();

cdr.IMEI = i.Substring(105, 20).Trim();

cdr.CellId = i.Substring(252, 22).Trim();

cdr.InTrunkSymbol = i.Substring(414, 20).Trim();

cdr.OutTrunkSymbol = i.Substring(394, 20).Trim();

cdr.ReleaseCode = i.Substring(274, 50).Trim();

DateTime ConnectDateTime;

if (DateTime.TryParseExact(i.Substring(221, 14).Trim(), "yyyyMddHHmmss", System.Globalization.CultureInfo.InvariantCulture,

System.Globalization.DateTimeStyles.None, out ConnectDateTime))

cdr.ConnectDateTime = ConnectDateTime;

int callType = 0;

if (int.TryParse(i.Substring(102, 3).Trim(), out callType))

cdr.CallType = (Vanrise.Fzero.CDRImport.Entities.CallType)callType;

decimal cellLatitude;

if (decimal.TryParse(i.Substring(609, 9).Trim(), out cellLatitude))

cdr.CellLatitude = cellLatitude;

decimal durationInSeconds;

if (decimal.TryParse(i.Substring(235, 5).Trim(), out durationInSeconds))

cdr.DurationInSeconds = durationInSeconds;

decimal upVolume;

if (decimal.TryParse(i.Substring(588, 10).Trim(), out upVolume))

cdr.UpVolume = upVolume;

decimal cellLongitude;

if (decimal.TryParse(i.Substring(618, 9).Trim(), out cellLongitude))

cdr.CellLongitude = cellLongitude;

decimal downVolume;

if (decimal.TryParse(i.Substring(598, 10).Trim(), out downVolume))

cdr.DownVolume = downVolume;

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//End of Dynamic Section

batch.CDRs.Add(cdr);

}

mappedBatches.Add("Normalize CDRs", batch); // Normalize then Save

Vanrise.Integration.Entities.MappingOutput result = new Vanrise.Integration.Entities.MappingOutput();

result.Result = Vanrise.Integration.Entities.MappingResult.Valid;

return result;

1. SQL Server:

/\*This is a sample code that maps CDRs from an sql adapter to the system

There is one section in this file that should be subject to change, other parts should remain the same for the current project. \*/

Vanrise.Fzero.CDRImport.Entities.ImportedCDRBatch batch = new Vanrise.Fzero.CDRImport.Entities.ImportedCDRBatch();

batch.CDRs = new List<Vanrise.Fzero.CDRImport.Entities.CDR>();

Vanrise.Integration.Entities.DBReaderImportedData ImportedData = ((Vanrise.Integration.Entities.DBReaderImportedData)(data));

IDataReader reader = ImportedData.Reader;

string index = ImportedData.LastImportedId;

while (reader.Read())

{

Vanrise.Fzero.CDRImport.Entities.CDR cdr = new Vanrise.Fzero.CDRImport.Entities.CDR();

cdr.Id = Convert.ToInt32(reader["Id"]);

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//Start of Dynamic Section

cdr.MSISDN = reader["MSISDN"] as string;

cdr.IMSI = reader["IMSI"] as string;

cdr.Destination = reader["Destination"] as string;

cdr.CallClass = reader["Call\_Class"] as string;

cdr.SubType = reader["Sub\_Type"] as string;

cdr.IMEI = reader["IMEI"] as string;

cdr.CellId = reader["Cell\_Id"] as string;

cdr.InTrunkSymbol = reader["In\_Trunk"] as string;

cdr.OutTrunkSymbol = reader["Out\_Trunk"] as string;

cdr.ReleaseCode = reader["ReleaseCode"] as string;

//When you have a property that allows null and is not a string, make sure to cast it as follow (Datetime?), (decimal?), (int?) etc...

cdr.ConnectDateTime = (DateTime?) reader["ConnectDateTime"];

cdr.CallType = (Vanrise.Fzero.CDRImport.Entities.CallType?) reader["Call\_Type"];

cdr.CellLatitude = (decimal?) reader["Cell\_Latitude"];

cdr.DurationInSeconds = (decimal?) reader["DurationInSeconds"];

cdr.UpVolume = (decimal?) reader["Up\_Volume"];

cdr.CellLongitude = (decimal?) reader["Cell\_Longitude"];

cdr.DownVolume = (decimal?) reader["Down\_Volume"];

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//End of Dynamic Section

index = cdr.Id.ToString();

batch.CDRs.Add(cdr);

}

ImportedData.LastImportedId = index;

mappedBatches.Add("Normalize CDRs", batch);

Vanrise.Integration.Entities.MappingOutput result = new Vanrise.Integration.Entities.MappingOutput();

result.Result = Vanrise.Integration.Entities.MappingResult.Valid;

return result;