****

**Area PI Tag Searcher Module Developer Manual for the**

**I-Field Tags Synchronization System**

**at**

**EXPEC Computer Center (ECC)**

Saudi Aramco, Saudi Arabia

|  |
| --- |
|  |

**Document Type: Developer Manual**

**Document Name: ECC Area PI Tag Searcher Developer Manual**

**Version: 1**

**Version Date: 11-May-2018**

**Crucial Solutions & Services (CSS)**

**Address**: P.O. Box 10381 | Jubail Industrial City 31961 | Kingdom of Saudi Arabia   
**Tel**: +966 (013) 349 3391 | **Fax**: +966 (013) 349 3392

**Email**: sales@css.net.sa | **Website**: [www.css.net.sa](http://www.css.net.sa)

Copyright © 2018 Crucial Solutions and Services (CSS)

The information contained herein is the sole property of Crucial Solutions & Services (CSS)**.** This document is reserved for viewing by EXPEC Computer Center (ECC), Saudi Aramco (Aramco) and consented recipients. No part of this document, may be reproduced, transmitted or modified (in any form or by any means) outside of the intended recipients without the express prior written consent of Crucial Solutions & Services (CSS).

Windows, SQL Server, SharePoint, Office, and Excel are registered trademarks of Microsoft Corporation.

ProcessBook, DataLink, WebParts, Coresight, AF, HA, and PI are registered trademarks of OSIsoft Inc.

All other trademarks are the property of their respective owners.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Date | Updated by | Revision Comments |
| 1 | 11-May-2018 | Houssam al Moughrabi | First Issue |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[SECTION 1: Summary 4](#_Toc514065725)

[SECTION 2: Architecture 5](#_Toc514065726)

[2.1 Diagram 5](#_Toc514065727)

[2.2 ECCPIAreaSearcher\_Service - Windows Service 6](#_Toc514065728)

[2.3 ECC\_DataLayer - Data Layer 7](#_Toc514065729)

[2.3.1 Repositories 7](#_Toc514065730)

[2.3.2 Stores 7](#_Toc514065731)

[2.3.3 Helpers 8](#_Toc514065732)

[2.3.4 DataModels 8](#_Toc514065733)

[2.4 ECC\_ServicesLayer – Service Layer 9](#_Toc514065734)

[2.4.1 Services 9](#_Toc514065735)

[2.4.2 Helpers 10](#_Toc514065736)

[SECTION 3: Workflow 11](#_Toc514065737)

[3.1 Diagram 11](#_Toc514065738)

[3.2 Description 12](#_Toc514065739)

[3.2.1 Get the I-Field servers information 12](#_Toc514065740)

[3.2.2 Find newly created PI points in each I-Field 12](#_Toc514065741)

[3.2.3 Create the found tags in Oracle database 12](#_Toc514065742)

[3.2.4 Update the last pull date for each I-Field server in Oracle database 12](#_Toc514065743)

[SECTION 4: Database Dependencies 13](#_Toc514065744)

[4.1 Flags 13](#_Toc514065745)

[4.2 Queries 13](#_Toc514065746)

[SECTION 5: Service App.Config 14](#_Toc514065747)

[SECTION 6: Installer 15](#_Toc514065748)

# Summary

This document describes the development manual of the Area PI Tag Searcher part of the I-Field Tags Synchronization System Project at ECC, Saudi Aramco

This Manual considers a Visual Studio 2017 installed as a pre-requisite.

# Architecture

This section will visualize and describe how the code is structured for this service.

Each module represented is a standalone .NET project which are connected together via reference.

The main objective of this architecture is to provide a scalable and flexible solution implemented in multiple projects each with a specific role and concern.

## Diagram

**Windows Service**

**ECCPIAreaSearcher\_Service**

**ECC\_DataLayer**

**.NET Class Library**

Oracle DB

**ECC\_Services\_Layer**

**.NET Class Library**

PI AF SDK

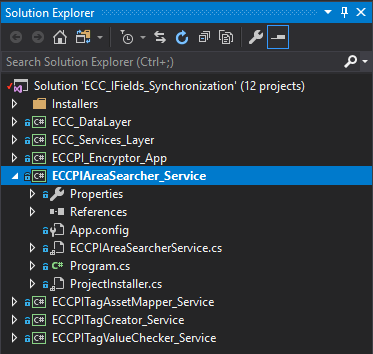
PI AF/

PI Data Archive

## ECCPIAreaSearcher\_Service - Windows Service

This is a windows service which is the main output project that will be installed on the server.

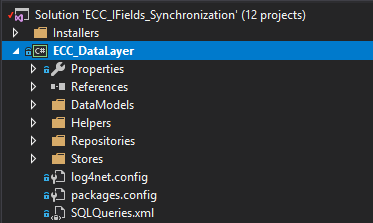
The main role of this module is to wrap all the logic implemented inside a windows service and handle the execution of the process.



## ECC\_DataLayer - Data Layer

This is a .NET class library responsible for communicating with database (mainly Oracle).

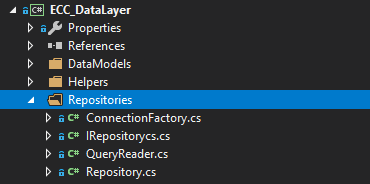
The data layer contains all the methods that are used by the service to Get/Update/Write data in database, also it contains the data models that represents the tables in the database.



Here is how the data layer is composed:

### Repositories

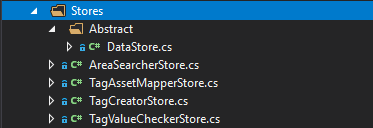
This is the repository which contain the classes and methods that Open/Close connection with the database, Execute a query (sync or async) using dapper ORM and Read a query from the ***SQLQueries.xml*** file.



### Stores

This namespace contains the implementation of the queries used, each class is for a specific windows service and contains all the methods that being used for executing the queries.

All stores inherit the ***DataStore.cs*** which implements some generic methods related to all inherited stores.



### Helpers

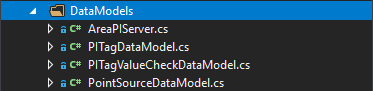
This namespace implements the general functionalities that could be used by any module or class in the project such like Encryption/Decryption of text or Logging.



### DataModels

It’s a representation of the tables used in the queries (specifically the SELECT queries) to manipulate from within the code.

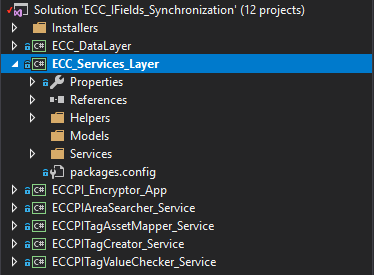
The queries results are mapped into these models using Dapper ORM (.NET SDK).



## ECC\_ServicesLayer – Service Layer

It contains all the business logic behind the service and communicate with PI Data Archive / AF via SDK and also communicates with the ECC\_DataLayer when needed.

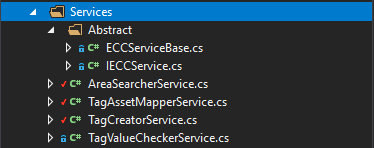
All the services required for the I-Field Tags Synchronization project are implemented inside this .NET class library.



Here is how the services layer is composed:

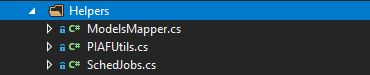
### Services

This namespace contains the implementation of the business logic of all services, each service is represented with a class where it contains all the workflow.



### Helpers

Contains the generic functionalities that are being used by all the services such like Objects Mapping, Communication with PI or AF via PI AF SDK and Service Job scheduling.



# Workflow

ECCPIAreaSearcher is responsible for searching for the tags newly created in the I-Fields PI servers and put these tags into the ECC Oracle database.

## Diagram

Below a diagram representing the workflow.

Schedule next execution date/time

Find newly created PI points in each I-Field

Create the found tags in Oracle database

Update the last pull date for each I-Field server in Oracle database

Get the I-Field servers information

## Description

### Get the I-Field servers information

Query the existing I-Field servers recorded in Oracle database in table “ECCPI\_SERVERS\_LIST” and get the last pull date and server code to be used during the process.

### Find newly created PI points in each I-Field

Using the last pull date and server code collected previously, the service is querying each server finding the created or modified tags after the last pull date.

### Create the found tags in Oracle database

The above found tags are inserted into table “ECCPI\_AF\_WELL\_FOUND\_TAGS” along with the tags attributes such like ENGUNITS, DIGITALSET, POINTTYPE, LOCATION2, LOCATION3, LOCATION5, USERINT1, USERINT2, USERREAL1, USERREAL2, COMPRESSING, COMPDEV, COMPMAX, COMPMIN, COMPDEVPERCENT, EXCDEV, EXCMAX, EXCMIN, EXCDEVPERCENT, SPAN, STEP, TYPICALVALUE and ZERO.

### Update the last pull date for each I-Field server in Oracle database

After inserting the tags found in I-Field server into database, the service will set the current date as the last pull date from the servers to be used in the next scheduled job so it won’t collect the same tags again and keeping the behavior monitored and optimized.

# Database Dependencies

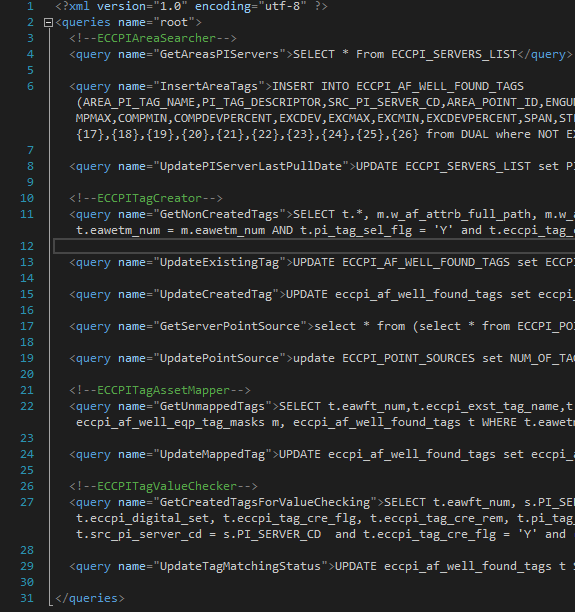
## Flags

The table contains the main columns/flags that are being updated after the execution of the service.

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Column Name | Type | Usage |
| ECCPI\_SERVERS\_LIST | PI\_LAST\_TAG\_PULL\_DT | Date | Used when querying PI servers for newly created or modified tags and it’s updated after inserting the tags in database |

## Queries

The below list contains the queries that has been used in this service located in **SQLQueries.xml** file.



The queries names listed below are identified in the **“name”** attribute of the <query> node, example: (<query name="QueryName"> QUERY GOES HERE </query>.)

All the query are being executed from the classes inherited from **ECC\_DataLayer.Stores.Abstract.DataStore** class which is responsible for executing the database queries.

**Queries Used:**

* GetAreasPIServers
* InsertAreaTags
* UpdatePIServerLastPullDate

# Service App.Config

This file is located under the **ECCPIAreaSearcher\_Service** windows service and it contains the below keys.

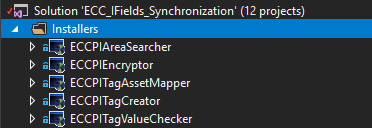
|  |  |
| --- | --- |
| Key Name | Description |
| ECC\_PI\_ServerName | ECC PI Data Archive server name or collective |
| ECC\_AF\_ServerName | ECC AF server name |
| ECC\_DB\_DataSource | Oracle data source name |
| ECC\_DB\_UserId | Oracle user Id |
| ECC\_DB\_Password | Encrypted oracle user password generated from Encryption tool |
| RUN\_FREQUENCY | The running frequency of the service e.g. Daily, Weekly, Monthly or Yearly |
| RUN\_HOUR | Time of scheduled job in Hours |
| RUN\_MINUTE | Time of scheduled job in Minutes |

# Installer

This section describes where the installers are located inside the source code and how to generate an installation file for windows service.

The installer is a .NET Setup Project configured to build the windows service output and pack it into an installer file (.msi and/or .exe file).

Setup project is located under **Installers** directory, it contains the setup project of all services each in a separate project as shown in the below image.



Once the developer has changed in the service code regardless in which project the changes has been made, a new installer should be provided to apply those changes.

In order to do that follow the below steps:

1. Locate the **ECCPIAreaSearcher** Setup Project under **Installers** directory
2. **Right click** and then click on “**Rebuild**”
3. Wait until the build is done then **Right click** on the project and click on “**Open folder in file explorer**”
4. Locate the installer under **Debug** directory and double click on one of the installers created (.msi or .exe) to start the installation process.