

openXDA

Installation

SOFTWARE OVERVIEW

The openXDA is an open source software (OSS) application enhanced for the Electric Power Research Institute (EPRI) by the Grid Protection Alliance (GAP) that enables the analysis of power system disturbance records and power quality (PQ) data files. It is assumed that a user will have a basic understanding of power system disturbances and PQ data, including the concepts of ‘faults’, ‘events’ and ‘trends’, and the measurement quantities typically included in disturbance records or PQ data files.

openXDA is an enterprise class back-office Windows service and performs the functions of a data integration and analysis platform. openXDA can be used as an automated stand-alone analysis system to populate a relational database and historian archive with data and analysis results, and can be used to generate automated email notifications. It can also be used to position data for use in other applications such as the Open PQ Dashboard. An overview diagram is included below to describe the context in which openXDA is deployed. The Open PQ Dashboard and other related applications are covered in their respective manuals.

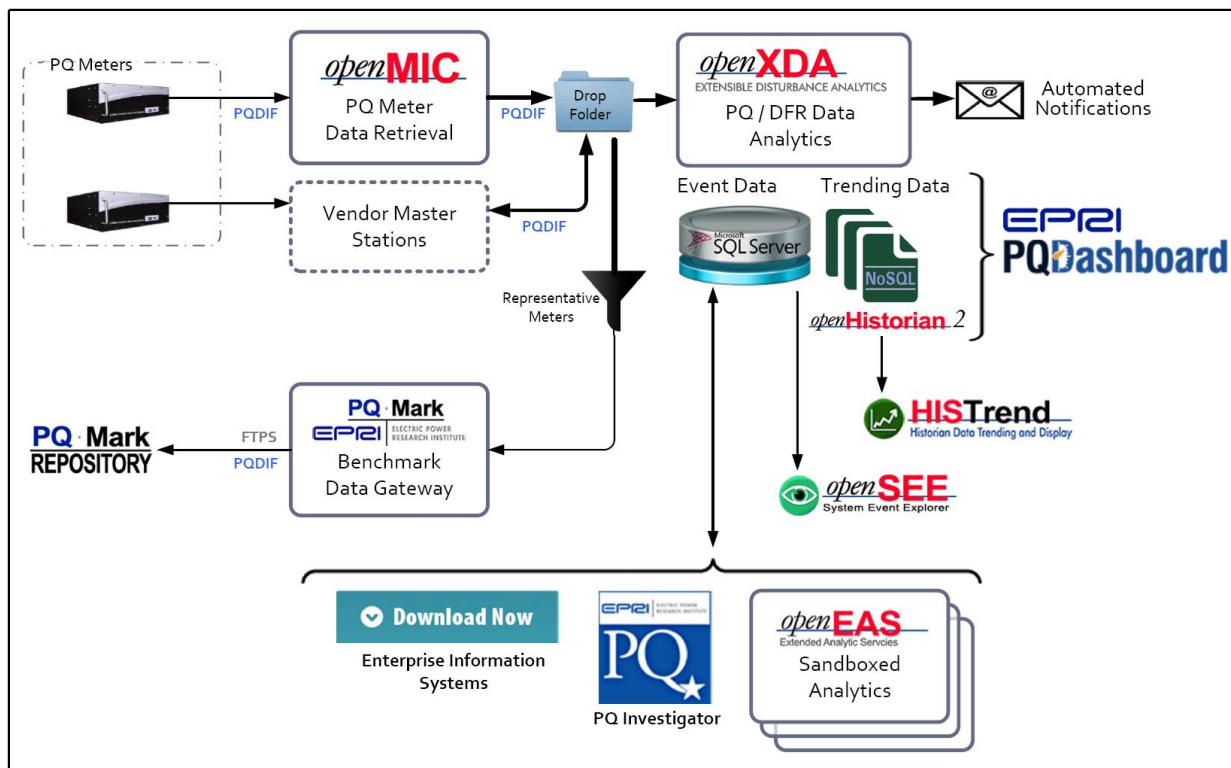


Figure 1. Architecture: OSS Disturbance Analytic Tools

Description

openXDA is an extensible platform for processing event and trending records from disturbance monitoring equipment such as digital fault recorders (DFRs), relays, power quality meters, and other power system IEDs. It includes a parser for COMTRADE and PQDIF formatted records, and demonstrations have been conducted using Schweitzer Engineering Laboratories (SEL) .eve

files. openXDA can be used as a data integration layer and can facilitate the development of automated analytic systems. openXDA has been deployed in a number of major US utilities to perform automated fault distance calculations based on disturbance waveform data combined with line parameters. openXDA determines the fault presence and fault type, and uses 6 different single ended fault distance calculation methods to determine the line-distance to the fault. If data is available from both ends of the line for a specific event a double-ended calculation is also performed. For more information see: The BIG Picture - Open Source Software (OSS) for Disturbance Analytic Systems and openFLE overview.

openXDA is a ‘back-end’ Windows service, and is typically added to existing IT infrastructure. It can be deployed in a test environment, but it is designed to be an enterprise class analysis engine and typically requires the assistance of IT professionals such as database, network, and application administrators. As noted above, the output is one or more of the following; new values in a relational database, new entries in a time series historian, and/or automated email notifications. By design, it does not include a user interface (UI) to visualize or interact with results. Downstream applications can access the information produced by openXDA, or the Open PQ Dashboard can be installed for visualization. This link provides information on the latest development version of the Open PQ Dashboard:

<https://github.com/GridProtectionAlliance/PQDashboard>

Information on EPRI published versions of the Open PQ Dashboard can be obtained directly from EPRI by emailing askepri@epri.com

Proper operation of the openXDA service, can be monitored and managed through the openXDA Admin Console which is included in the installation package.

Benefits and Value

openXDA is a platform comprised of a back office service designed to consume all disturbance and PQ records that conform to either the IEEE PQDIF or IEEE COMTRADE formats, and a configuration database that controls the operation of the service and the automated analytics.

The benefits of this approach includes:

- A single analysis platform for multiple data types
- The ability to accept standard data formats from any vendor’s device
- The ability to automatically analyze input data for fault information
- The ability to extend the analytics with any appropriate algorithms
- The ability to evaluate trended data values with respect to alarms and normal operating ranges
- The ability to automatically send email notifications based on alarms and configuration

The value of these new insights include:

- More quickly recognizes faults and PQ system failures
- More timely reaction to faults and PQ issues
- Better allocation of resources for corrective measures
- Better management of data acquisition devices and processes

Platform Requirements

The following items are minimum requirements for successful installation and deployment of the openXDA.

Operating System

64-bit Windows 7 or Windows Server 2008 R2 (or newer).

Minimum Hardware

- 2.0 GHz CPU.
- 2.0 GB RAM.
- 50 GB of available disk space for installation and testing. Operational disk space requirements will be proportional to the volume of input data.

Software

- .NET 3.5 SP1 (required by SQL Server 2012).
- .NET 4.6.
- SQL Server 2012 with management tools.
 - Free Express version is fine, but has a 10GB limit.
 - Mixed mode authentication must be enable on the SQL Server
- openHistorian 2.0

CONTENTS

1 INSTALLATION INSTRUCTIONS	8
2 PREREQUISITES	8
3 INSTALLING .NET 4.6	9
4 INSTALLING OPENHISTORIAN 2.0	9
5 INSTALLING OPENXDA	10
Run openXDASetup	10
End-User License Agreement	12
Custom Setup	13
Database Connection	14
Ready to install openXDA	15
User Account Control	16
Installing openXDA Progress	17
Setup Finish	18
6 LOAD TEST SYSTEM CONFIGURATION	19
7 LOAD TEST DATA	24
Included Test Data	29
8 CONFIGURING OPENXDA TO USE YOUR OWN DATA	29
Method 1: Modify DeviceDefinitions.xml	30
Method 2: Excel Spreadsheet for Creating DeviceDefinitions.xml	30
Load DeviceDefinitions.xml to Configure openXDA for Your Devices	30
9	33
OPENXDA SERVICE CONFIGURATION	33
10	39
OPENEAS TEMPLATE	39
Description	39
Benefits and Value	40
Prerequisites	40
Implementation	40
Step 1: Download Zip File	40
Step 2: Extract Files from Zip	41
Step 3: Rename project files	42
Step 4: Open Your Project	43
Step 5: Modify Port Assignment	45
Step 6 Update Sandbox.sql file	48
Step 7 Your Code: Functions or External dlls	51
Step 8 PQ Dashboard Display	53
11	58

PQI INTEGRATION	58
Description	58
Step 1: Set up database permissions	58
Step 2: Execute the PQI Integration script	59
Step 3: Populate the MeterFacility table	59
PQI – openXDA – Open PQ Dashboard Integration Complete	60

1

INSTALLATION INSTRUCTIONS

This document provides a list of all required hardware and software components, but is focused only on the installation, configuration, and testing of the openXDA. References to prerequisite commercial and OSS software packages are provided as a convenience. Following the steps in the order presented in sections 2 through 7 below should result in a successful installation including the openXDA database configuration and data processing for a test dataset.

Instructions for configuring the openXDA database for your own data are included in Section 8.

2

PREREQUISITES

The following hardware and software items are required before the openXDA can be successfully installed. The operating system and database server are assumed to be standard IT infrastructure and are not addressed in this document. The .NET framework is a commercially available third party software package. If a prerequisite software element is already installed, the respective section of this document can be skipped.

Note: The openXDA service requires mixed mode authentication to be enabled on the SQL Server instance where the openXDA database is installed. This setting can be selected during SQL Server installation and is turned off by default. If your instance is configured to allow only Windows authenticated users or if you are unsure whether mixed mode authentication is enabled, refer to the following link for instructions on how to modify the setting:

<https://msdn.microsoft.com/en-us/library/ms188670.aspx>.

Operating System

64-bit Windows 7 or Windows Server 2008 R2 (or later versions)

Minimum Hardware

- 2.0 GHz processor
- 2.0 GB of memory
- 50 GB of available disk space for installation and testing
- Operational disk space requirements will be proportional to the volume of input data

Software

- .NET 3.5 SP1 (required by SQL Server 2012)
- .NET 4.6
- SQL Server 2012 with management tools
 - Free Express version is fine, but has a 10GB limit.
 - Mixed mode authentication must be enable on the SQL Server

- openHistorian 2.0

3

INSTALLING .NET 4.6

Note: If .NET 4.6 is installed, please go to [INSTALLING OPENHISTORIAN 2.0](#).

Download the .NET 4.6 installer from the following location:

<https://www.microsoft.com/en-us/download/confirmation.aspx?id=48130>

Install this version of .NET before continuing to other installation steps.

4

INSTALLING openHistorian 2.0

Note: If openHistorian 2.0 is installed, please go to [INSTALLING OPENXDA](#).

Download the openHistorian 2.0 from the following location:

<http://www.gridprotectionalliance.org/NightlyBuilds/openHistorian/Beta/openHistorian.Installs.zip>

Extract the downloaded archive and run Setup.exe to begin the installation. Follow the installation steps until you reach the following step. Enter the information as shown in the screenshot.

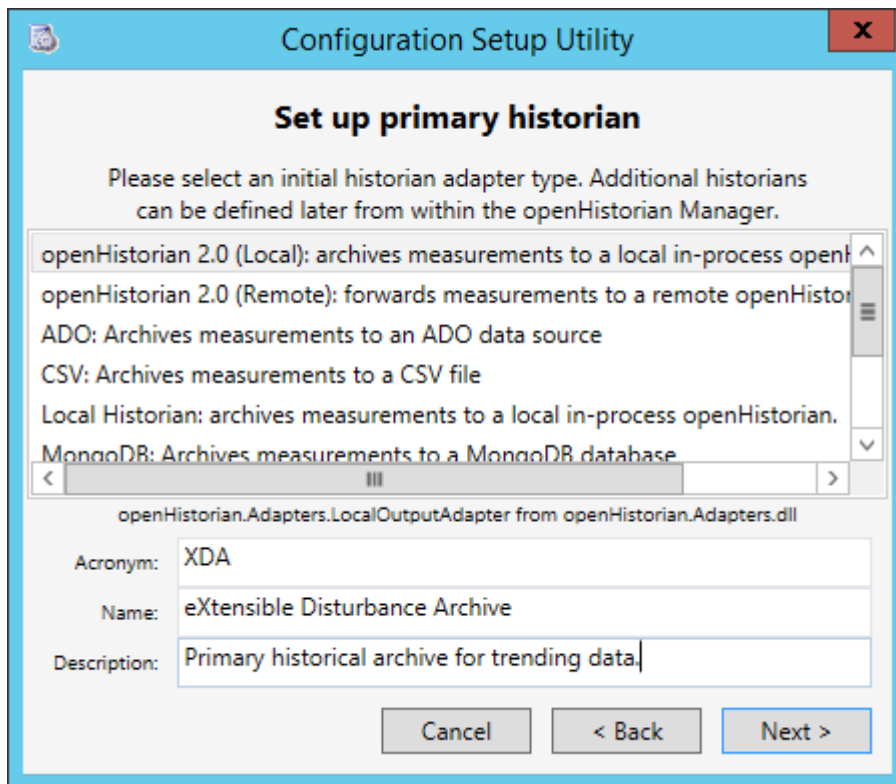


Figure 2. openHistorian Setup: Set up primary historian

After entering the information as shown above, follow the rest of the installation steps to completion. openHistorian 2.0 installation must be completed before continuing to other sections of this manual.

5 INSTALLING openXDA

Download the latest release of openXDA here:

<https://github.com/GridProtectionAlliance/openXDA/releases/latest>

Run openXDASetup

The file openXDASetup.msi is located in the openXDA.Installs.zip archive. Upon launching the MSI, the following screen will appear, click Next to install.



Figure 3. openXDA Setup: initial screen

End-User License Agreement

Click the check box to accept the MIT License terms then click Next to continue, or Cancel to exit the installation.

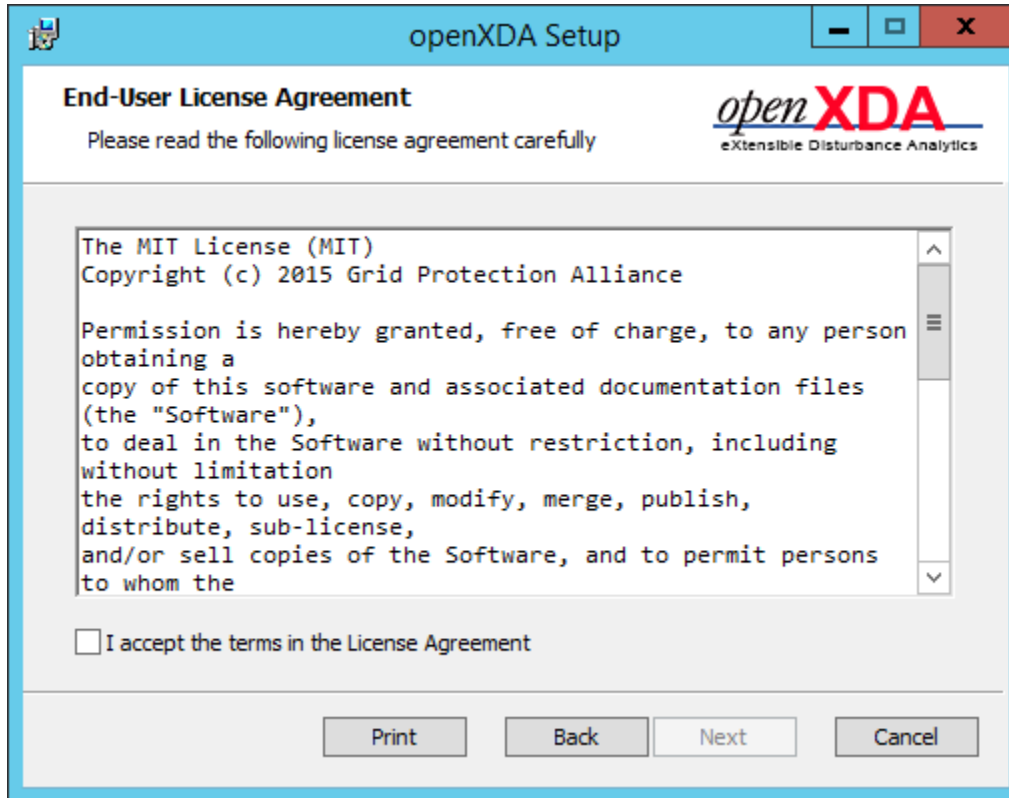


Figure 4. openXDA Setup: end-user license agreement

Custom Setup

For a new installation all components should be installed as shown in the screen below. If a different installation location is desired click the Browse button and select the location. When any changes to the setup screen are complete click Next.

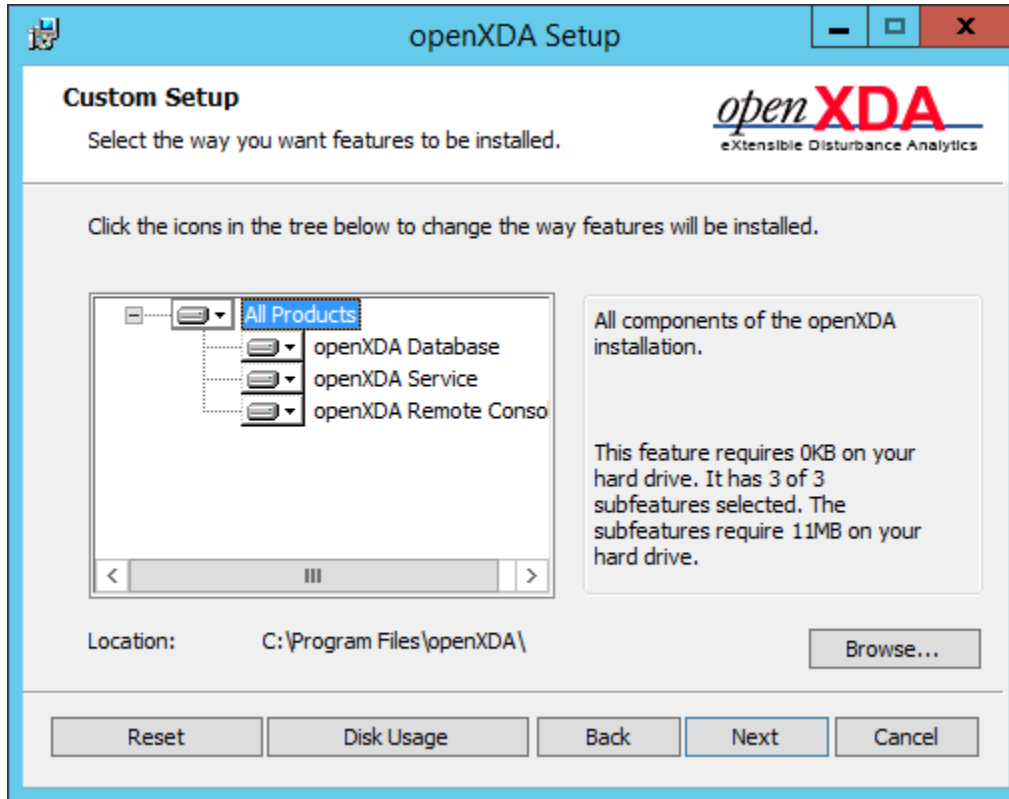


Figure 5. openXDA Setup: custom setup screen

Database Connection

For a new installation the default values are recommended but may be changed as specified by your database administrator. When the database connection is specified as desired click Next.

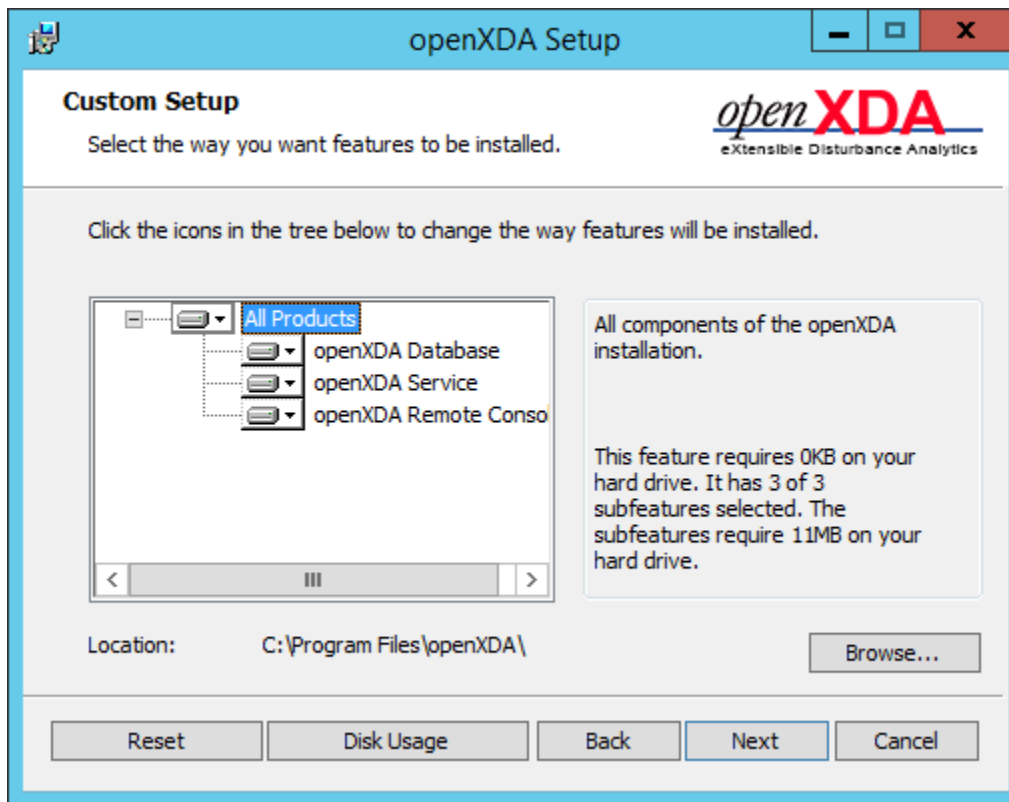


Figure 6. openXDA Setup: database connection

Ready to install openXDA

When you are ready to install openXDA click the Install button.

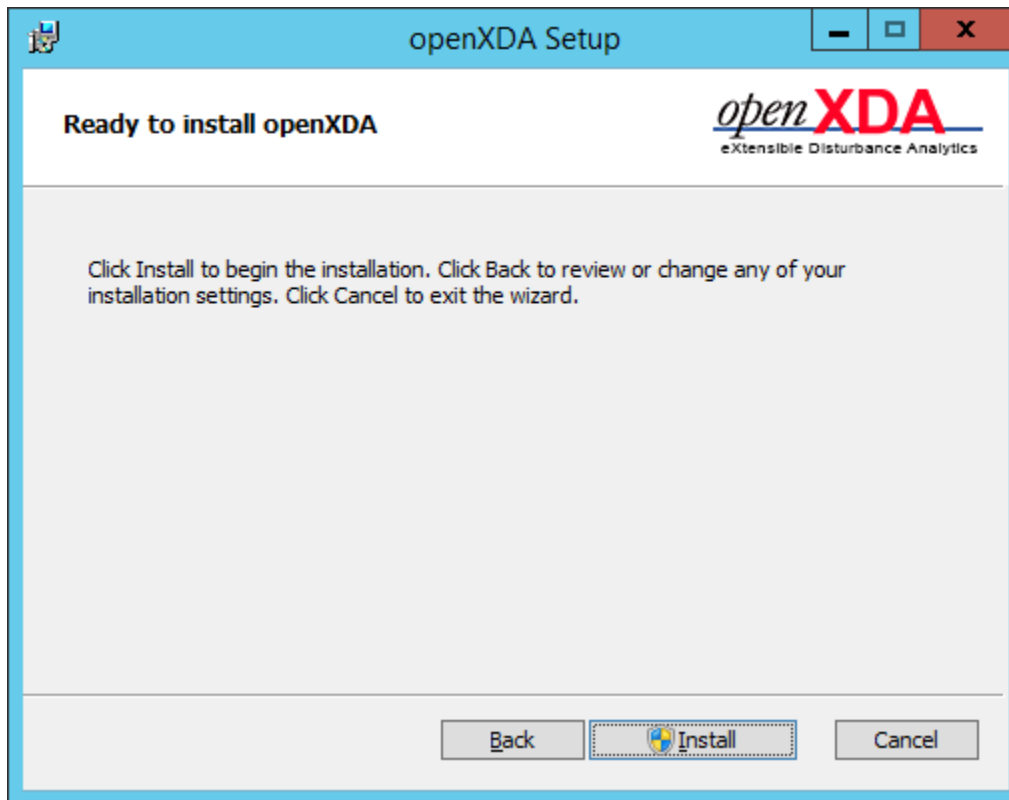
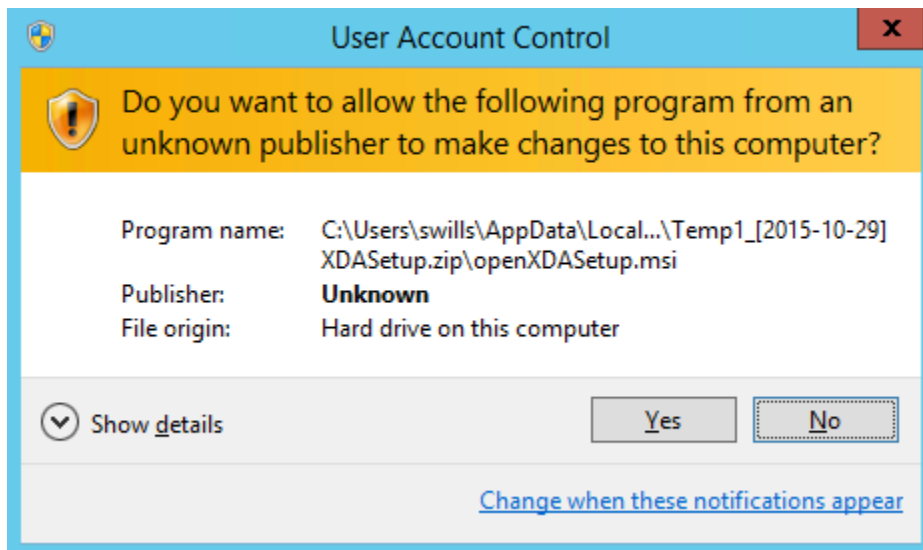


Figure 7. openXDA Setup: ready to install screen

User Account Control

If you want the openXDA setup to install openXDA on your computer click the Yes button, if not click No to cancel the installation.



Installing openXDA Progress

Installation progress will be indicated in the screen below. Click next when the install is complete.

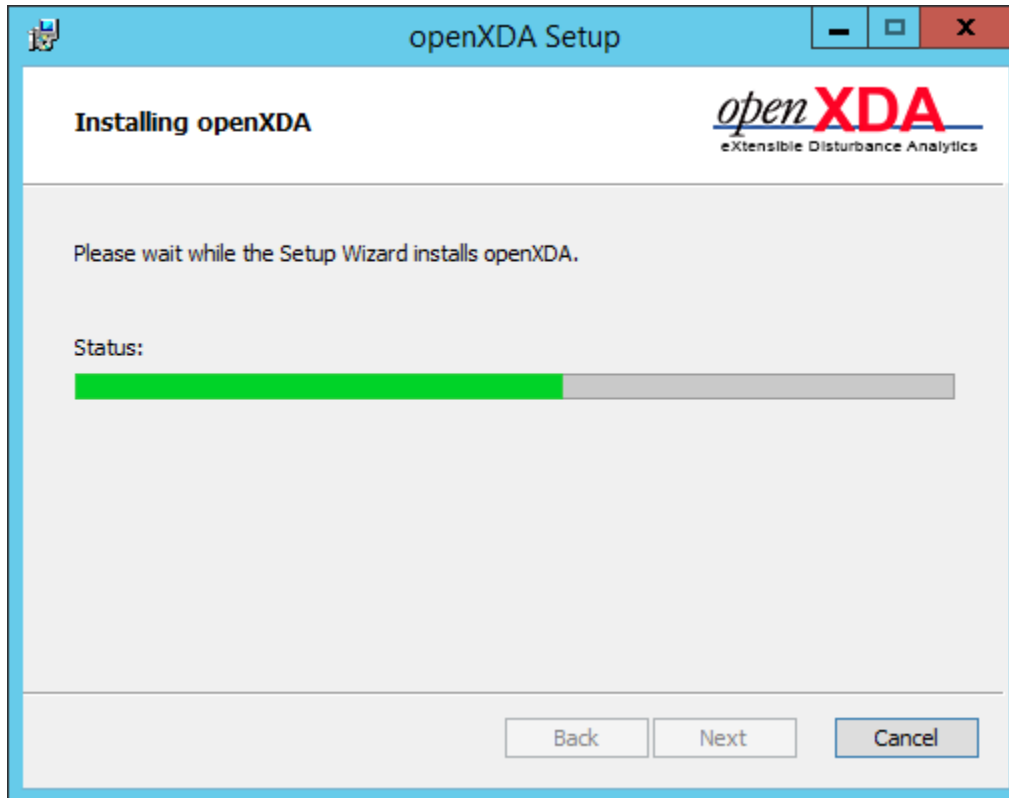


Figure 8. openXDA Setup: installation progress

Setup Finish

When the screen below is displayed to indicate that openXDA Setup has completed click the Finish button to dismiss the screen.

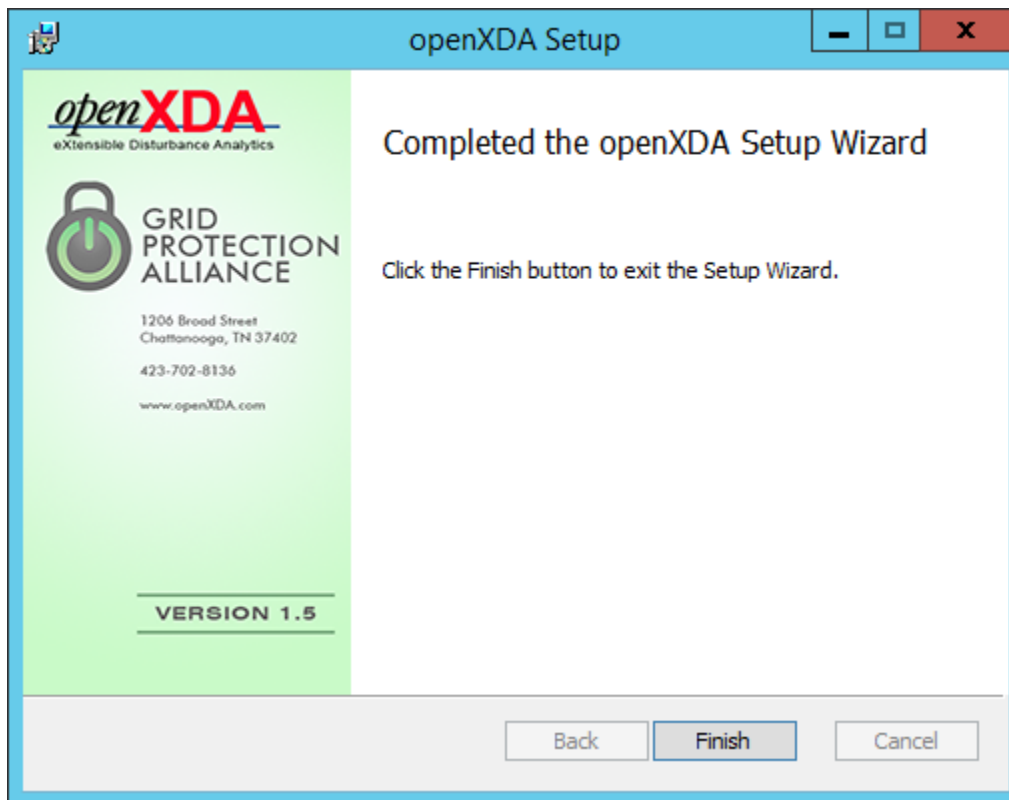


Figure 9. openXDA Setup: installation completed screen

6

LOAD TEST SYSTEM CONFIGURATION

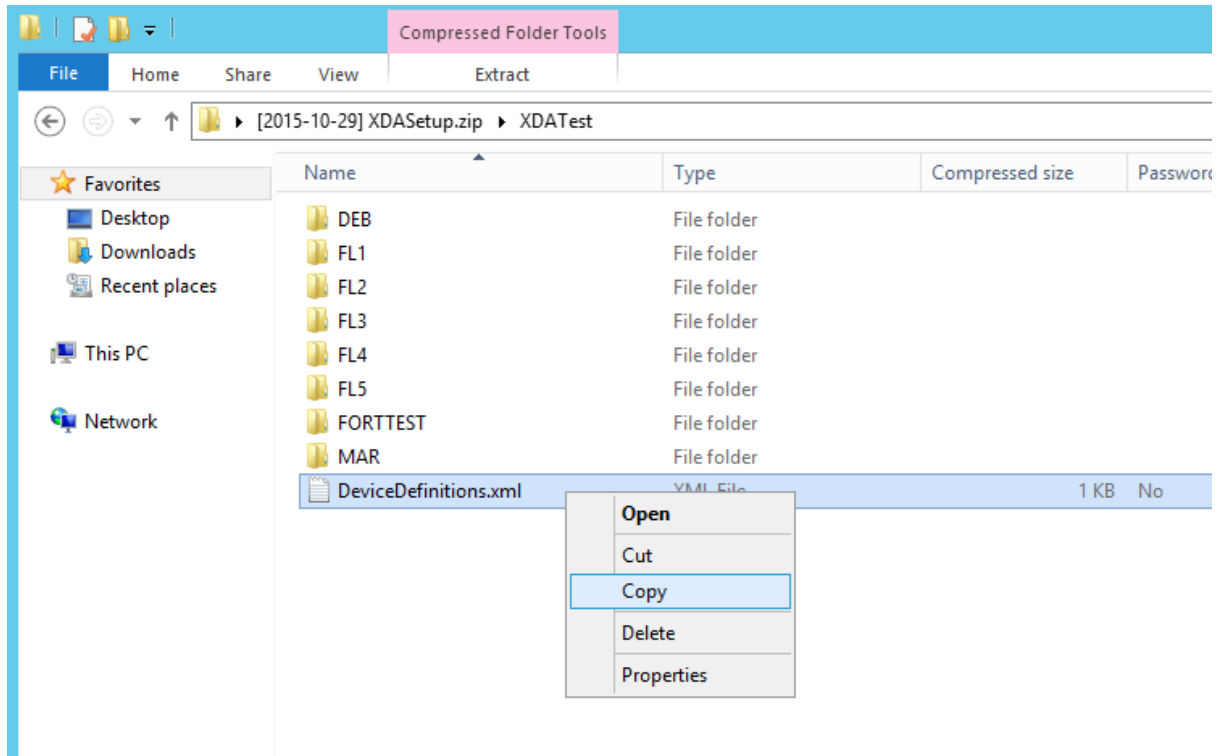


Figure 10. Copying Device Definitions File

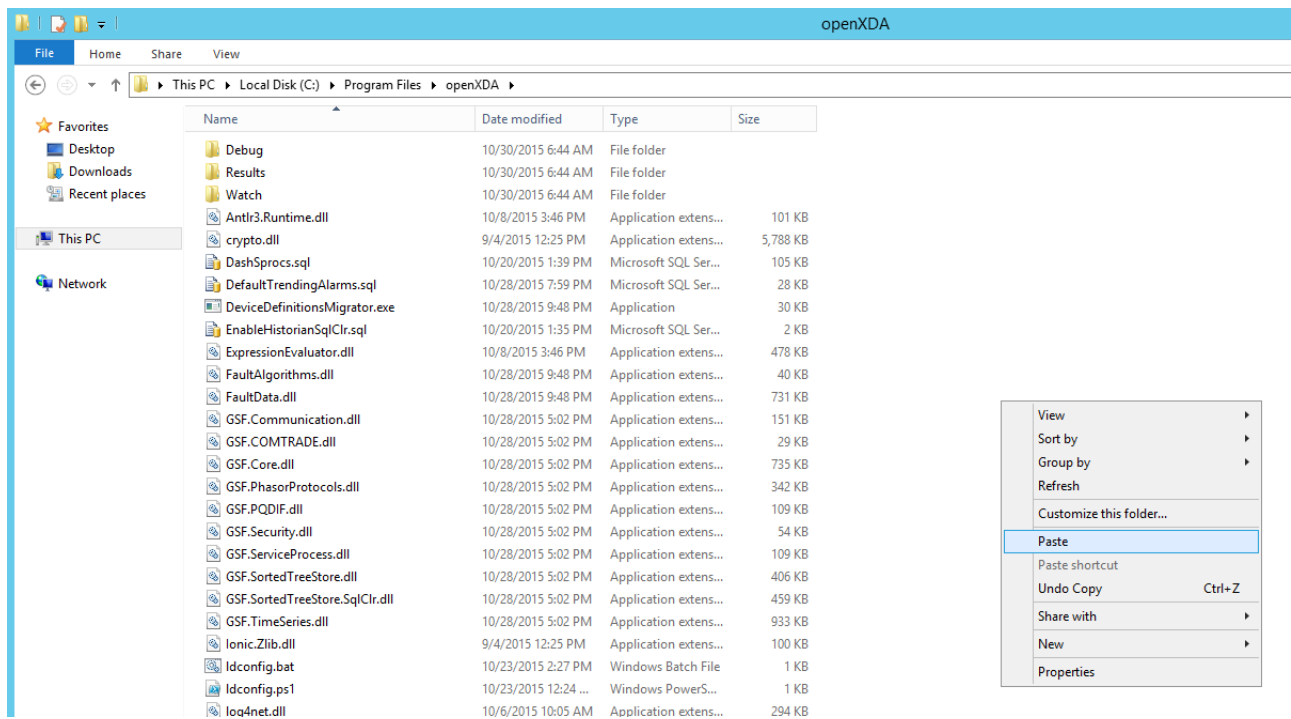


Figure 11. Paste Device Definitions File in openXDA installation folder

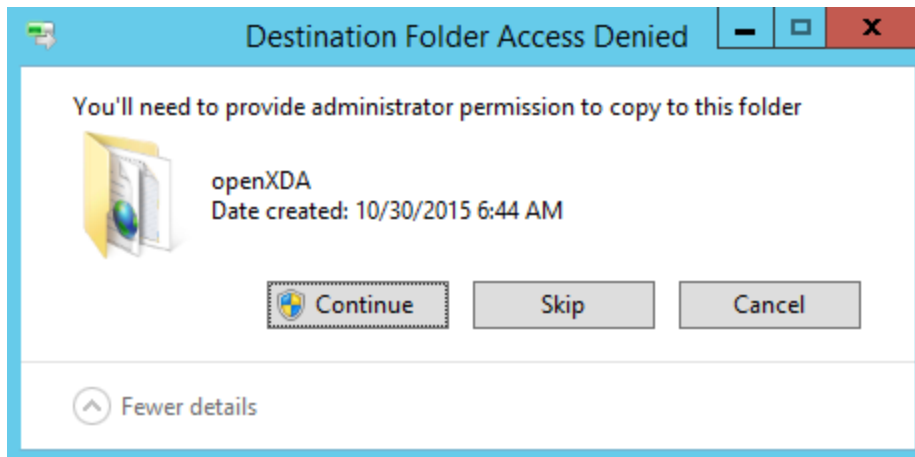


Figure 12. If prompted for administrator permissions press Continue

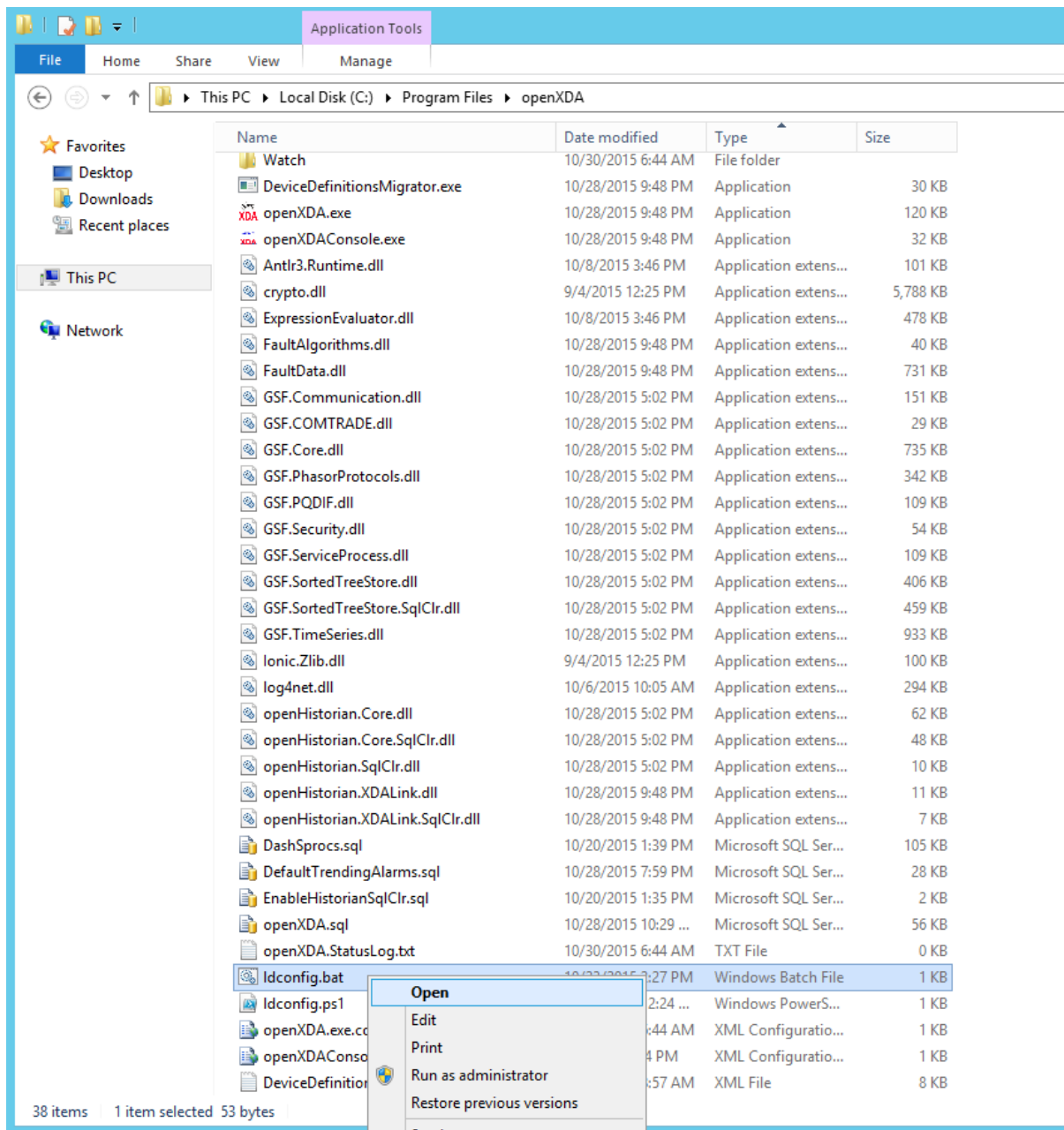
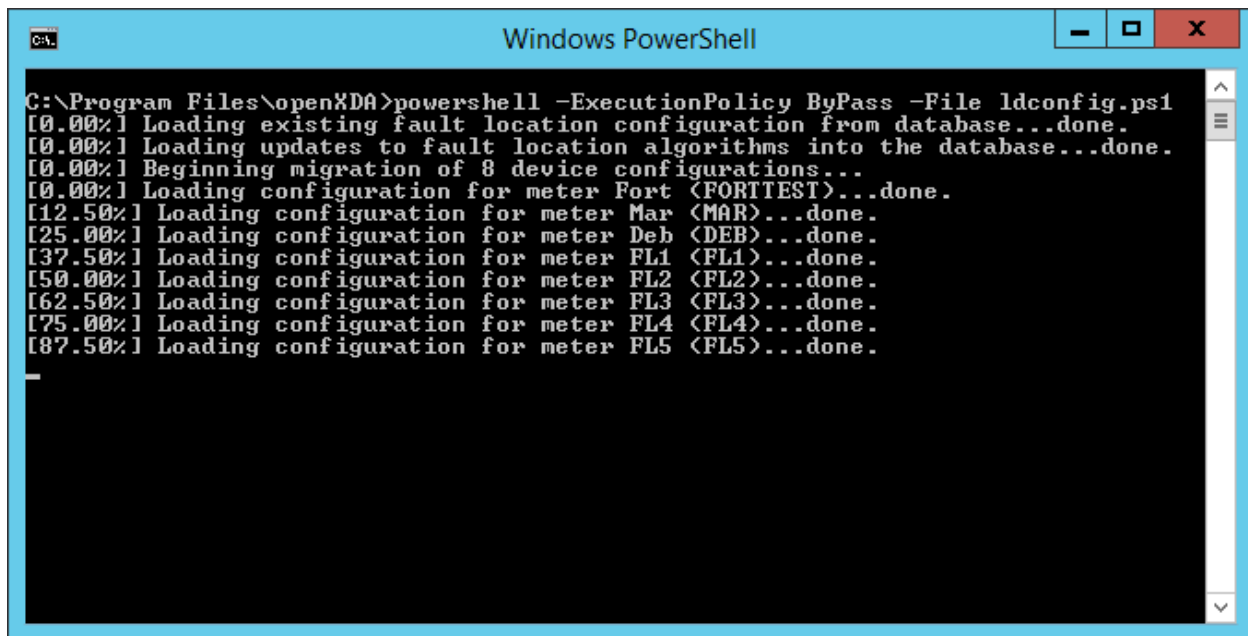


Figure 13. Open Idconfig.bat file



```
C:\Program Files\openXDA>powershell -ExecutionPolicy Bypass -File ldconfig.ps1
[0.00%] Loading existing fault location configuration from database...done.
[0.00%] Loading updates to fault location algorithms into the database...done.
[0.00%] Beginning migration of 8 device configurations...
[0.00%] Loading configuration for meter Fort (FORTTEST)...done.
[12.50%] Loading configuration for meter Mar (MAR)...done.
[25.00%] Loading configuration for meter Deb (DEB)...done.
[37.50%] Loading configuration for meter FL1 (FL1)...done.
[50.00%] Loading configuration for meter FL2 (FL2)...done.
[62.50%] Loading configuration for meter FL3 (FL3)...done.
[75.00%] Loading configuration for meter FL4 (FL4)...done.
[87.50%] Loading configuration for meter FL5 (FL5)...done.
```

Figure 14. Idconfig.bat loading system configuration file

7

LOAD TEST DATA

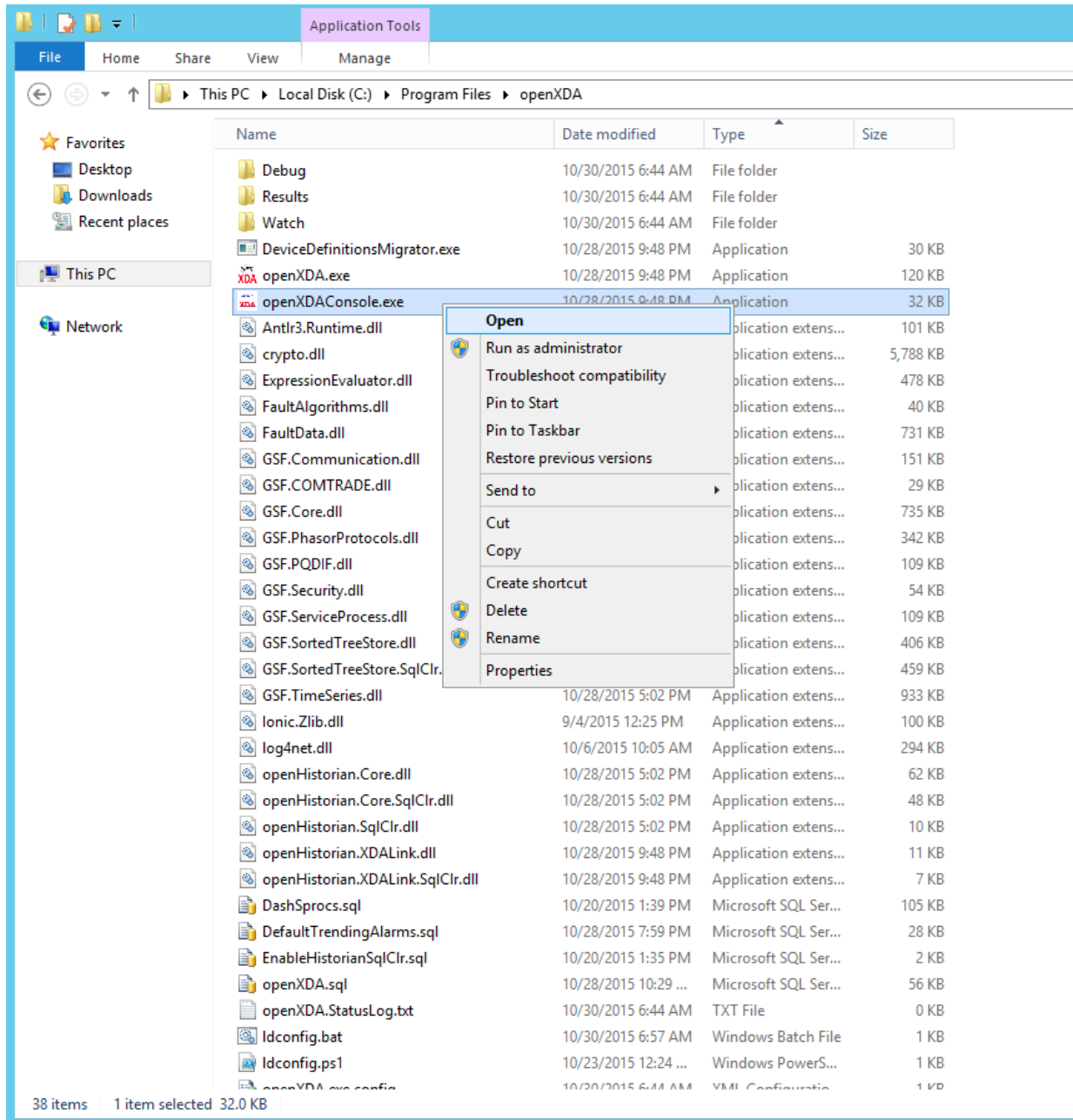
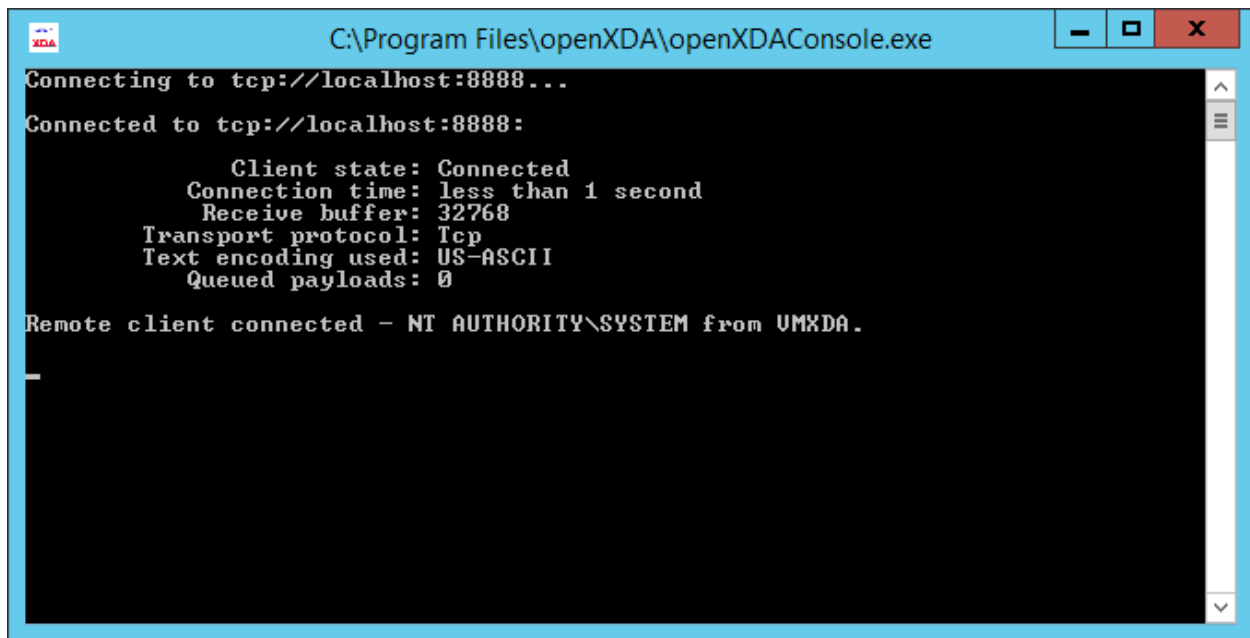


Figure 15. Open openXDA console to monitor service operation

A screenshot of a Windows console window titled "C:\Program Files\openXDA\openXDAConsole.exe". The window has a blue title bar and standard Windows window controls (minimize, maximize, close). The console output is as follows:

```
Connecting to tcp://localhost:8888...
Connected to tcp://localhost:8888:
    Client state: Connected
    Connection time: less than 1 second
    Receive buffer: 32768
    Transport protocol: Tcp
    Text encoding used: US-ASCII
    Queued payloads: 0
Remote client connected - NT AUTHORITY\SYSTEM from UMXDA.
```

Figure 16. openXDA console display

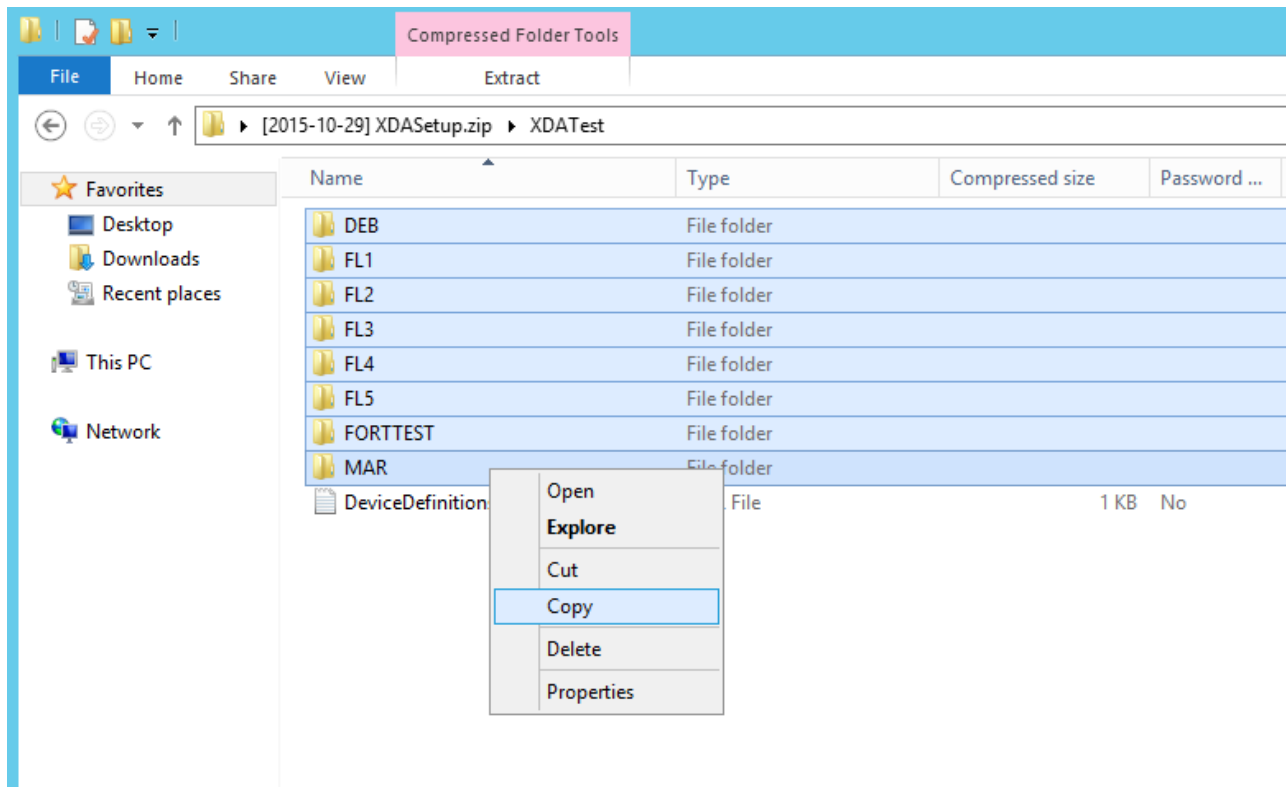


Figure 17. Copy Test Data

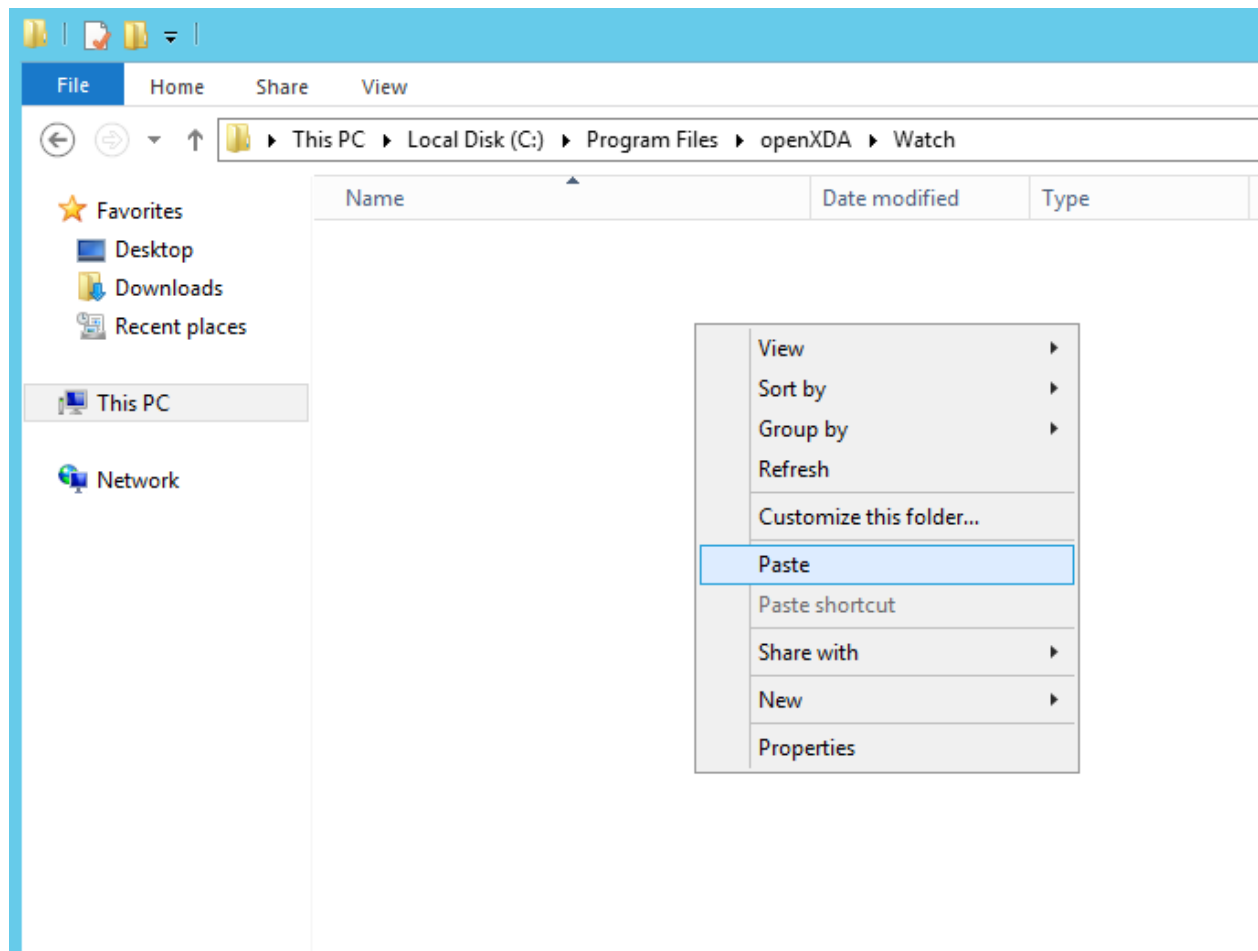


Figure 18. Past Test Data to Watch Folder

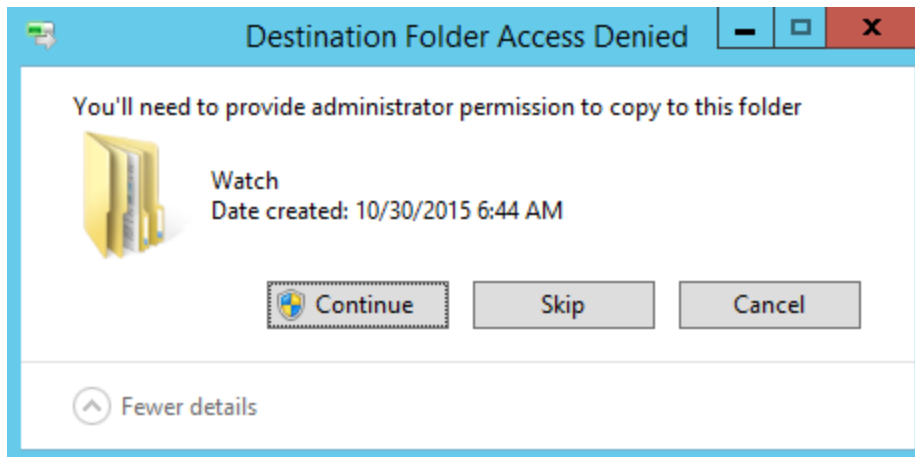
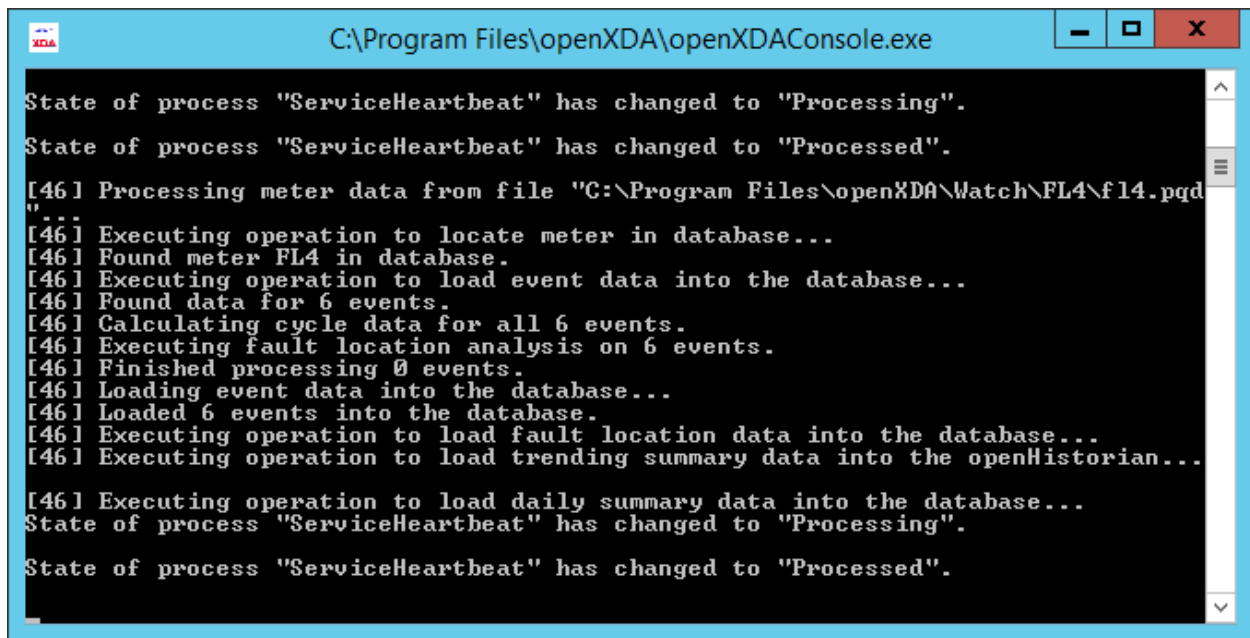


Figure 19. If prompted for administrator permissions press continue

The image shows a screenshot of a Windows console window titled "C:\Program Files\openXDA\openXDAConsole.exe". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The console output is as follows:

```
State of process "ServiceHeartbeat" has changed to "Processing".
State of process "ServiceHeartbeat" has changed to "Processed".
[46] Processing meter data from file "C:\Program Files\openXDA\Watch\FL4\fl14.pqd"
[46] Executing operation to locate meter in database...
[46] Found meter FL4 in database.
[46] Executing operation to load event data into the database...
[46] Found data for 6 events.
[46] Calculating cycle data for all 6 events.
[46] Executing fault location analysis on 6 events.
[46] Finished processing 0 events.
[46] Loading event data into the database...
[46] Loaded 6 events into the database.
[46] Executing operation to load fault location data into the database...
[46] Executing operation to load trending summary data into the openHistorian...
[46] Executing operation to load daily summary data into the database...
State of process "ServiceHeartbeat" has changed to "Processing".
State of process "ServiceHeartbeat" has changed to "Processed".
```

Figure 20. openXDA console display of service messages

Included Test Data

The test dataset contains data for 8 sites with data available for exercising all tabs. Dates from 12/29/2013 through 01/08/2014 should be used for all tabs except faults. Data to exercise the fault tab is on 09/03/2014.

8 CONFIGURING OPENXDA TO USE YOUR OWN DATA

As described in section 6, Load System Configuration, one method for configuring the openXDA database is to run the `ldconfig.bat` file which uses the `DeviceDefinitions.xml` file to obtain values for configuration. This is the best method for anyone except a database administrator (dba). A dba might prefer to modify values in the configuration tables directly, but that process is out of the scope of this document.

There are two methods for creating a `DeviceDefinitions.xml` file for your devices:

1. Method 1: copy and modify the appropriate `DeviceDefinitions.xml` file included in the install package using an appropriate xml aware text editor such as Notepad++
2. Method 2: use the Excel spreadsheet example file available from GPA.

These methods are described in detail below. Choose the method that you are most comfortable in using. Examples of `DeviceDefinitions.xml` files are located on Github. There are two example files. One is for DFR and/or Relay data files in COMTRADE format and the other is for PQ data files in the PQDIF format. These examples include a description of the configuration parameters and how they will be used in openXDA.

Method 1: Modify DeviceDefinitions.xml

Create a new folder under the XDASetup.zip folder, or the location of your choice, called My Definitions. Download the appropriate example xml file from the Github site.

DeviceDefinitionsExample-DFR-Relay.xml or DeviceDefinitionsExample-PQ.xml and place it in the new folder. Edit the example xml file using an appropriate text editor. Comments in the respective xml file provide guidance on specifying your device and line definitions.

Method 2: Excel Spreadsheet for Creating DeviceDefinitions.xml

Create a new folder under the XDASetup.zip folder, or the location of your choice, called My Definitions. Download the example Excel spreadsheet from the Github site. Read through the spreadsheet to be sure you understand the required fields. A recommended approach is to use one spreadsheet for each device such as a DFR or PQ monitor. Within each spreadsheet enter your information regarding the device, stations, and lines as appropriate. Duplicate the line specifications for each line associated with the device. Note that PQ monitors may not be associated with a line. In that case, the only required fields are 'id', 'name', and 'voltage'.

Refer to the Github site for details to migrate from spreadsheet to the devicedefinitions.xml file.

Load DeviceDefinitions.xml to Configure openXDA for Your Devices

When you have a completed a device definitions xml file that represents your devices and the input data sources, copy your file to the name DeviceDefinitions.xml in the openXDA folder as shown below.

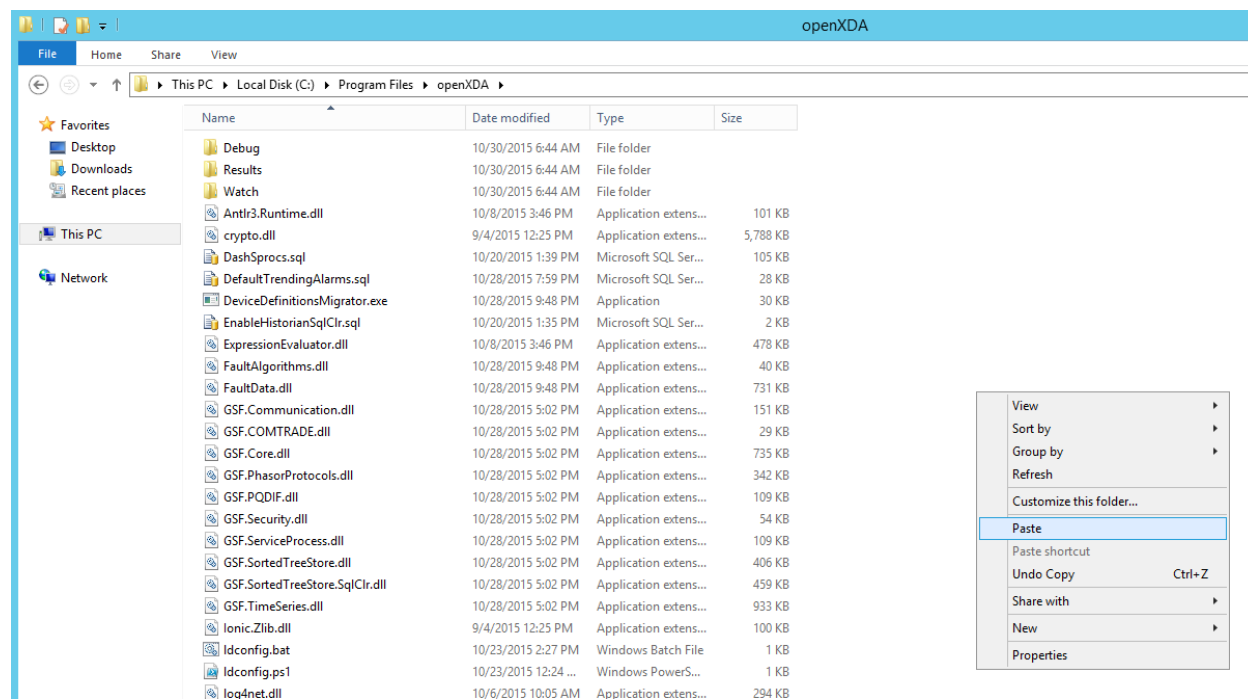


Figure 21 Default location for DeviceDefinitions.xml file

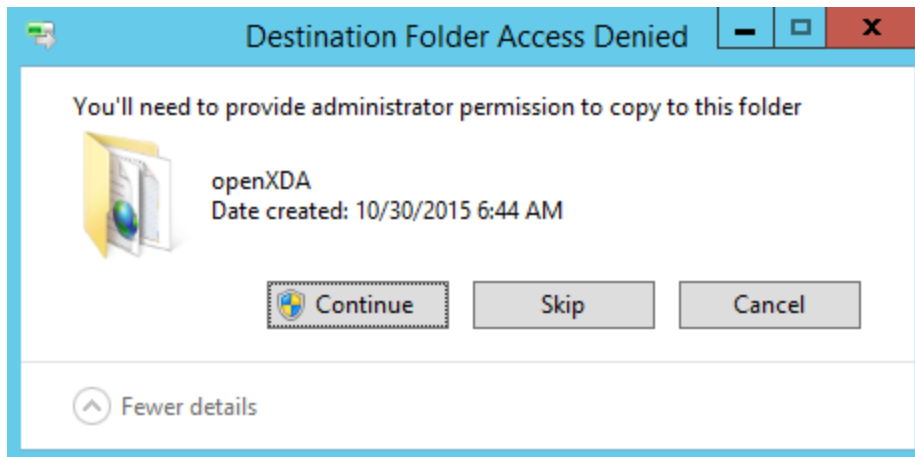


Figure 22 If prompted for administrator permissions press Continue

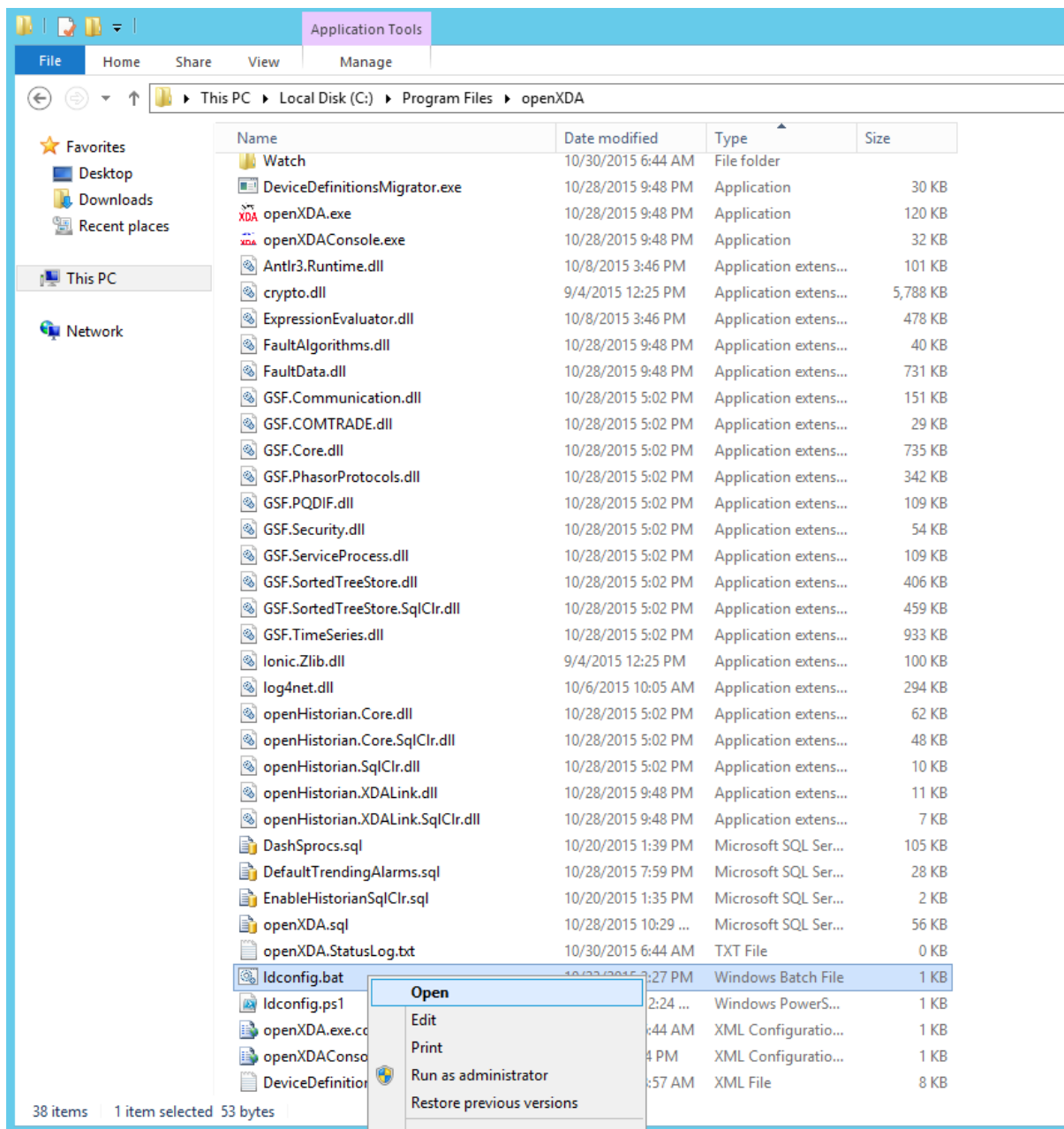
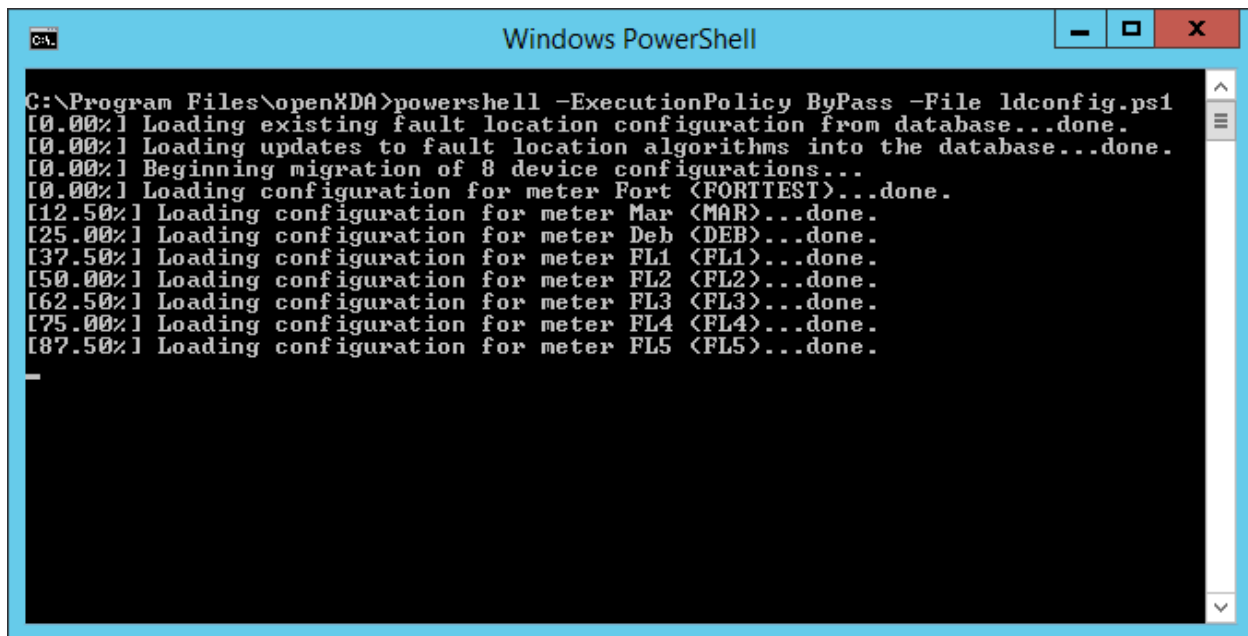


Figure 23 Open Idconfig.bat file



```
C:\Program Files\openXDA>powershell -ExecutionPolicy Bypass -File ldconfig.ps1
[0.00%] Loading existing fault location configuration from database...done.
[0.00%] Loading updates to fault location algorithms into the database...done.
[0.00%] Beginning migration of 8 device configurations...
[0.00%] Loading configuration for meter Fort <FORTTEST>...done.
[12.50%] Loading configuration for meter Mar <MAR>...done.
[25.00%] Loading configuration for meter Deb <DEB>...done.
[37.50%] Loading configuration for meter FL1 <FL1>...done.
[50.00%] Loading configuration for meter FL2 <FL2>...done.
[62.50%] Loading configuration for meter FL3 <FL3>...done.
[75.00%] Loading configuration for meter FL4 <FL4>...done.
[87.50%] Loading configuration for meter FL5 <FL5>...done.
```

Figure 24 ldconfig.bat loading system configuration file

9

OPENXDA SERVICE CONFIGURATION

This section contains a description of the setting table and available configuration variables used to specify the behavior of openXDA. The descriptions below include defaults and acceptable values. Changes to these settings would be determined by an application administrator and modifications to the setting table would be made by a dba.

Apart from the connection string in the **openXDA.exe.config** file, configuration options for the service are located in the database in a table called **Setting**. These are the configuration options that can be defined in the Setting table.

- **DbTimeout**
 - Default: 120
 - Amount of time each database query is given to complete, in seconds.
- **WatchDirectories**
 - Default: Watch
 - Semi-colon separated list of directories where fault records can be discovered by the service.

- **ResultsPath**
 - Default: Results
 - Directory to which the results of fault analysis will be written.
- **FilePattern**
 - Default: (?<AssetKey>[^\]\+)\][^\]\+$
 - Regular expression pattern that defines how files are associated with their meters.
 - A capture group for **AssetKey** must be specified.
 - See [http://msdn.microsoft.com/en-us/library/az24scfc\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx) for more information about regular expression in .NET.
- **DefaultMeterTimeZone**
 - Default: UTC
 - The time zone identifier for the time zone used by meters in the system unless explicitly configured otherwise.
 - See [https://msdn.microsoft.com/en-us/library/ms912391\(v=winembedded.11\).aspx](https://msdn.microsoft.com/en-us/library/ms912391(v=winembedded.11).aspx) for the list of Windows time zone identifiers.
- **XDATimeZone**
 - Default: Local time zone
 - The time zone identifier for the time zone used by openXDA when storing time data in the openXDA database.
 - Supplying an empty string (which is the default value for this setting) will default to the local time zone of the system on which openXDA is running.
- **WaitPeriod**
 - Default: 10.0
 - The amount of time, in seconds, between the time a file is processed by the system and the time an email should be produced by the system.
 - Increasing the wait period will allow openXDA more time to gather details about an event from multiple meters and provide a single email result.
 - Shortening the wait period will decrease the amount of time between the even and the email notification, but may result in redundancy between emails when new information arrives from another meter.
- **TimeTolerance**
 - Default: 0.5
 - The maximum distance, in seconds, between a meter's clock and real time.
 - Adjust this parameter to increase or decrease the tolerance used by openXDA when aligning events that occurred at approximately the same time. The system will attempt to determine whether two events could have occurred at the same time based on their

relative positions in time, taking into account this time tolerance.

- **MaxTimeOffset**
 - Default: 0.0
 - The maximum number of hours beyond the current system time before the time of the event record indicates that the data is unreasonable.
 - Events with data that is considered unreasonable will be excluded from fault analysis.
 - The default value of 0.0 disables this setting so that openXDA processes files regardless of how far in the future the timestamps are.
- **MinTimeOffset**
 - Default: 0.0
 - The maximum number of hours prior to the current system time before the time of the record indicates that the data is unreasonable.
 - Events with data that is considered unreasonable will be excluded from fault analysis.
 - The default value of 0.0 disables this setting so that openXDA processes files regardless of how old the timestamps are.
- **MaxFileDuration**
 - Default: 0.0
 - The maximum duration, in seconds, of the files processed by openXDA. Files with a larger duration will be skipped by openXDA's file processing engine.
 - The default value, 0.0, disables this setting so that openXDA processes all files regardless of the duration.
- **MaxFileCreationTimeOffset**
 - Default: 0.0
 - The maximum number of hours prior to the current system time before the file creation time indicates that the data should not be processed.
 - Use this setting when the file creation time closely coincides with the time of the data in the file to automatically skip old files and not process them.
 - The default value, 0.0, disables this setting so that openXDA processes all files regardless of the creation time.
- **SystemFrequency**
 - Default: 60.0
 - The frequency, in Hz, of the electrical system being analyzed by openXDA.
- **MaxVoltage**
 - Default: 2.0
 - The per-unit threshold at which the voltage exceeds engineering reasonableness.

- Events with data that exceeds engineering reasonableness will be excluded from fault analysis.
- **MaxCurrent**
 - Default: 1000000.0
 - The threshold, in amps, at which the current exceeds engineering reasonableness.
 - Events with data that exceeds engineering reasonableness will be excluded from fault analysis.
 -
- **FaultLocation.PrefaultTrigger**
 - Default: 5.0
 - The threshold at which the ratio between RMS current and prefault RMS current indicates faulted conditions.
 - If the ratio exceeds this threshold, the cycle is considered to have been recorded during a fault, but only if the fault suppression algorithm indicates that there is no reason to believe otherwise.
- **FaultLocation.MaxFaultDistanceMultiplier**
 - Default: 1.05
 - The multiplier applied to the line length to determine the maximum value allowed for fault distance before the results are considered invalid.
- **FaultLocation.MinFaultDistanceMultiplier**
 - Default: -0.05
 - The multiplier applied to the line length to determine the minimum value allowed for fault distance before the results are considered invalid.
- **Breakers.OpenBreakerThreshold**
 - Default: 20.0
 - The maximum current, in amps, at which the breaker can be considered open.
 - This threshold is compared against the amplitude (peak) of the sine wave fitted to each cycle of the current waveform using linear regression. A flat, noisy signal should produce a very low amplitude sine wave.
- **Breakers.LateBreakerThreshold**
 - Default: 1.0
 - The maximum number of cycles that a breaker operation's timing can exceed the configured breaker speed before being considered late.
- **LengthUnits**
 - Default: miles

- The units of measure to use for lengths.
- This value is only applied to human-readable exports and does not affect the fault calculations.
-
- **COMTRADEMinWaitTime**
 - Default: 15.0
 - The minimum amount of time, in seconds, to wait for additional data files after the system detects the existence of a .d00 COMTRADE file.
 - The best way to ensure that all data files are present before openXDA attempts to process them is to copy the data files first, then copy the .cfg file last.
- **ProcessingThreadCount**
 - Default: 0
 - The number of threads used for processing meter data concurrently.
 - Values less than zero indicate that the system should use as many threads as there are logical processors in the system.
- **MaxQueuedFileCount**
 - Default: 10
 - The number of files that can be queued on meter data processing threads before the system starts blocking the file processing thread.
 - Values less than or equal to zero will be set to one.
- **FileWatcherEnumerationStrategy**
 - Default: ParallelSubdirectories
 - Strategy used for enumeration of files in the file watcher.
 - **Sequential** processes all watch directories sequentially on a single thread using a depth-first recursive search.
 - **ParallelWatchDirectories** processes all watch directories in parallel on their own thread using a depth-first recursive search.
 - **ParallelSubdirectories** processes each directory on its own thread, including subdirectories.
 - **None** disables file enumeration.
- **FileWatcherMaxFragmentation**
 - Default: 10
 - The maximum amount of fragmentation allowed before compacting the list of processed files in the file watcher.
 - The amount of fragmentation is measured as the number of files that have been removed from the watch directory since the start of the program or the last compact operation.

- **FileWatcherInternalThreadCount**
 - Default: 0
 - The number of threads used internally to the file processor.
 - Values less than zero indicate that the file processor should use as many threads as there are logical processors in the system.
- **FileWatcherBufferSize**
 - Default: 8192
 - The size, in bytes, of the internal buffer used by the watchers of each of the configured watch directories.
 - This buffer is used to store information about the files involved in file system events. Small buffers may overflow causing file events to be missed by the system. Large buffers will use up large amounts of non-paged memory space which could cause system performance degradation or system errors.
 - This value can be between 4 KB and 64 KB. On Windows systems, use a multiple of 4 KB for better performance.
- **FileShares**
 - A double semicolon-separated list of connection strings that define the credentials required for the service to connect to a file share.
 - Each connection string should contain 3 settings: Name, Username, and Password.
 - **Name:** The name of the file share ([\\server\share](#)).
 - **UserName:** The name of the user to log in as (DOMAIN\USERNAME).
 - **Password:** The password of the user to log in as.
 - Alternatively, each file share can be defined as three separate settings in the Setting table.

Name	Value
FileShares.1.Name	\\server1\share
FileShares.1.Username	DOMAIN1\USER1
FileShares.1.Password	user1pwd
FileShares.2.Name	\\server2\share
FileShares.2.Username	DOMAIN2\USER2
FileShares.2.Password	user2pwd
Etc.	
- **Email.SMTPServer**
 - The hostname or IP address of the SMTP server used for sending emails when a fault is detected.
- **Email.FromAddress**

- Default: openXDA@gridprotectionalliance.org
- The email address placed on the From line of the emails sent when a fault is detected.
- **Email.Username**
 - The username used to authenticate to the SMTP server.
 - Remove this field from system settings or leave it blank if no authentication is required.
- **Email.Password**
 - The password used to authenticate to the SMTP server.
 - Remove this field from system settings if no authentication is required.
- **Email.EnableSSL**
 - Default: False
 - Flag that determines whether to enable SSL when establishing communications with the SMTP server.
- **Historian.Server**
 - Default: 127.0.0.1
 - The hostname and port of the openHistorian 2.0 server to be used for archiving trending data. (e.g. 127.0.0.1:38402).
- **Historian.InstanceName**
 - Default: XDA
 - The instance name of the historian instance to be used for archiving trending data.

10

OPENEAS TEMPLATE

Description

The openEAS template is a framework for the development of a Windows service to establish links between openXDA and external analytic services. It is designed to facilitate and simplify the process of leveraging external analytics while isolating openXDA from potential issues that could be introduced by external code. Implementations of openEAS can take advantage of analytics developed in Matlab or other platforms without modifying openXDA code. Two implementations of this template have been developed around advanced disturbance analytics for incipient cable fault detection and capacitor switching transients. Both implementations incorporate proprietary algorithms developed by EPRI in Matlab and interfaced through openEAS to openXDA and the Open PQ Dashboard using dlls

The openEAS template is developed using the concept of a ‘sandbox’, which is an isolated computing resource to protect other processes. The template code is located on Github in a file called ‘openEAS-master.zip’.

Benefits and Value

Although openXDA is designed to be extensible, there are scenarios where it is desired and beneficial to use external analytics rather than include those functions within openXDA.

Benefits of this approach include the ability to:

- protect intellectual property through DLLs
- expose results from the external analysis in the Open PQ Dashboard
- augment analytics in openXDA without modifying internal code
- quickly bring algorithms developed through research into a production environment

Value derived from this approach includes:

- obtaining production results from research algorithms
- enhanced use of external analytics
- reducing barriers for integration with openXDA
- providing new information for display in the Open PQ Dashboard
- providing production grade datastores for research algorithm results

Prerequisites

A properly installed and configured instance of openXDA

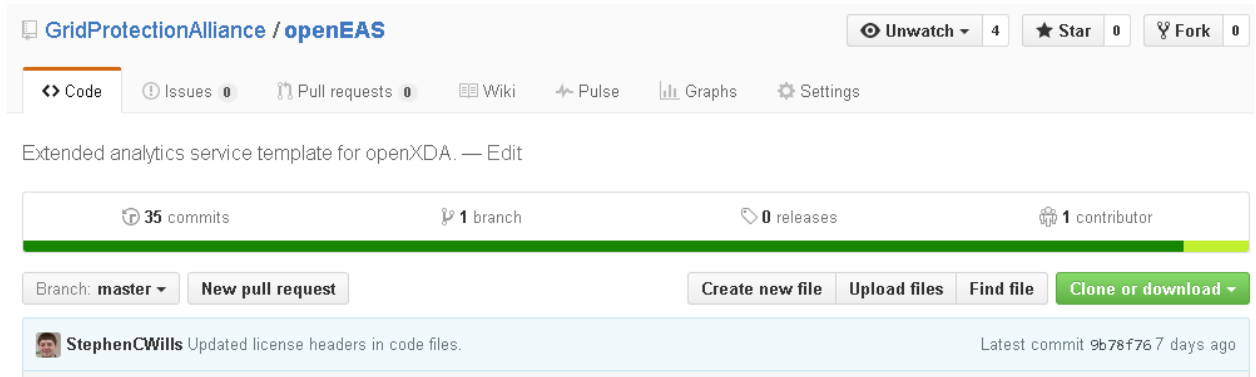
Microsoft Visual Studio for .NET 4.6 or higher

Optionally: Open PQ Dashboard for visualization

Implementation

Step 1: Download Zip File

Download openEAS-master.zip from <https://github.com/GridProtectionAlliance/openEAS> by clicking the ‘Clone or download’ button.



a.

Figure 25 Location of 'Clone or download' button

b. Click 'Download ZIP' button

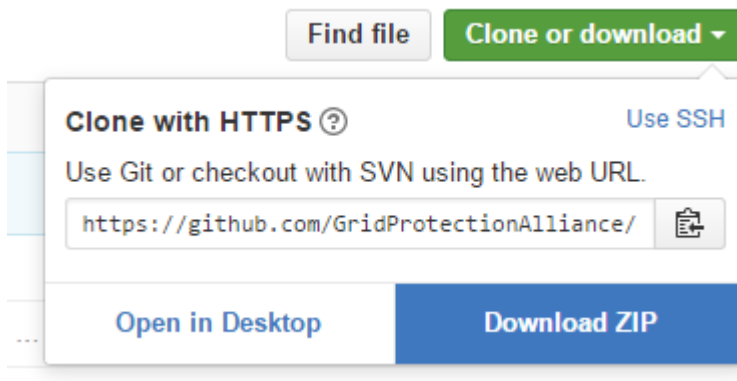


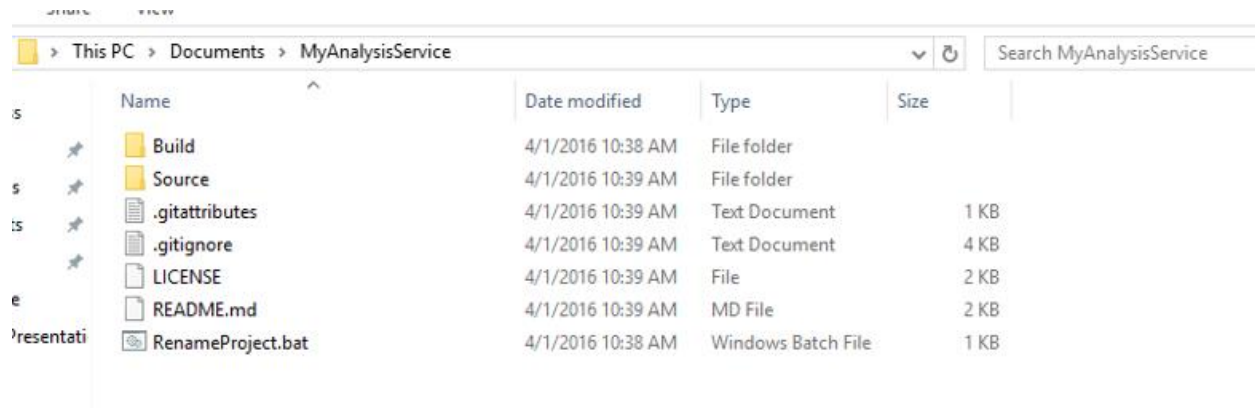
Figure 26 Location of 'Download Zip' button

c. Move the file openEAS-master.zip to a folder of your choice. This document will use the folder name documents\MyAnalysisService

Step 2. Extract Files from Zip

Extract files from the openEAS-master.zip document.

- Move to documents\MyAnalysisService and double click on openEAS-master.zip
- Double click on the new folder openEAS-master
- The list of files in your folder should look like this:



Name	Date modified	Type	Size
Build	4/1/2016 10:38 AM	File folder	
Source	4/1/2016 10:39 AM	File folder	
.gitattributes	4/1/2016 10:39 AM	Text Document	1 KB
.gitignore	4/1/2016 10:39 AM	Text Document	4 KB
LICENSE	4/1/2016 10:39 AM	File	2 KB
README.md	4/1/2016 10:39 AM	MD File	2 KB
RenameProject.bat	4/1/2016 10:38 AM	Windows Batch File	1 KB

Figure 27 List of files expanded from openEAS-master.zip

Step 3. Rename project files

Since you are modifying this template to perform your own functions, rename it to a name of your choice to avoid confusion with the original files. This document uses the name MAService. To follow these instructions, substitute the name you have chosen in every reference to MAService in the remainder of this document.

- a. Double click on the file named RenameProject.bat
- b. If a security dialog box pops up, click run.
- c. Type the name you have chosen for your service and press enter.

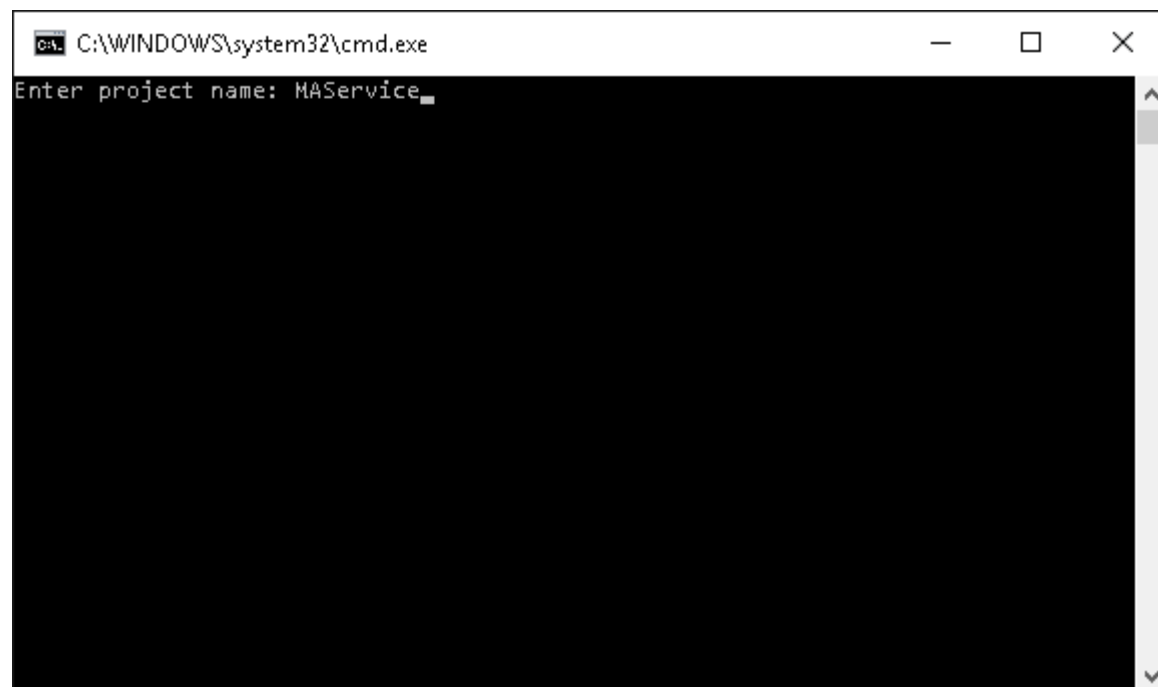
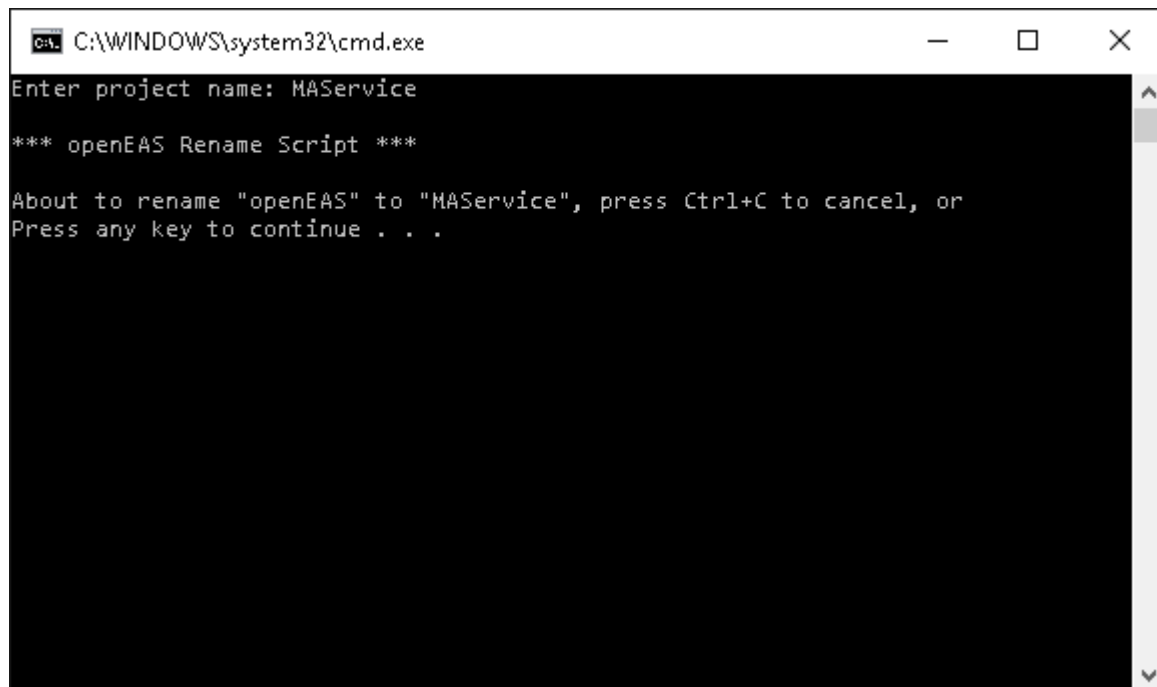


Figure 28 RenameProject.bat initial display

- d. If you mistype the name, hold the Ctrl key and press c. This will close the window and allow you to start over with a. above. If the name is correct, press any key to continue.



```
C:\WINDOWS\system32\cmd.exe
Enter project name: MAService

*** openEAS Rename Script ***

About to rename "openEAS" to "MAService", press Ctrl+C to cancel, or
Press any key to continue . . .
```

Figure 29 RenameProject.bat continue... display

- e. When the script is complete, the window will close automatically.

Figure 30 ProjectRename.bat Rename Complete display

Step 4. Open Your Project

Double click on the Source folder, then open your project by double clicking on the .sln file, e.g. MAService.sln

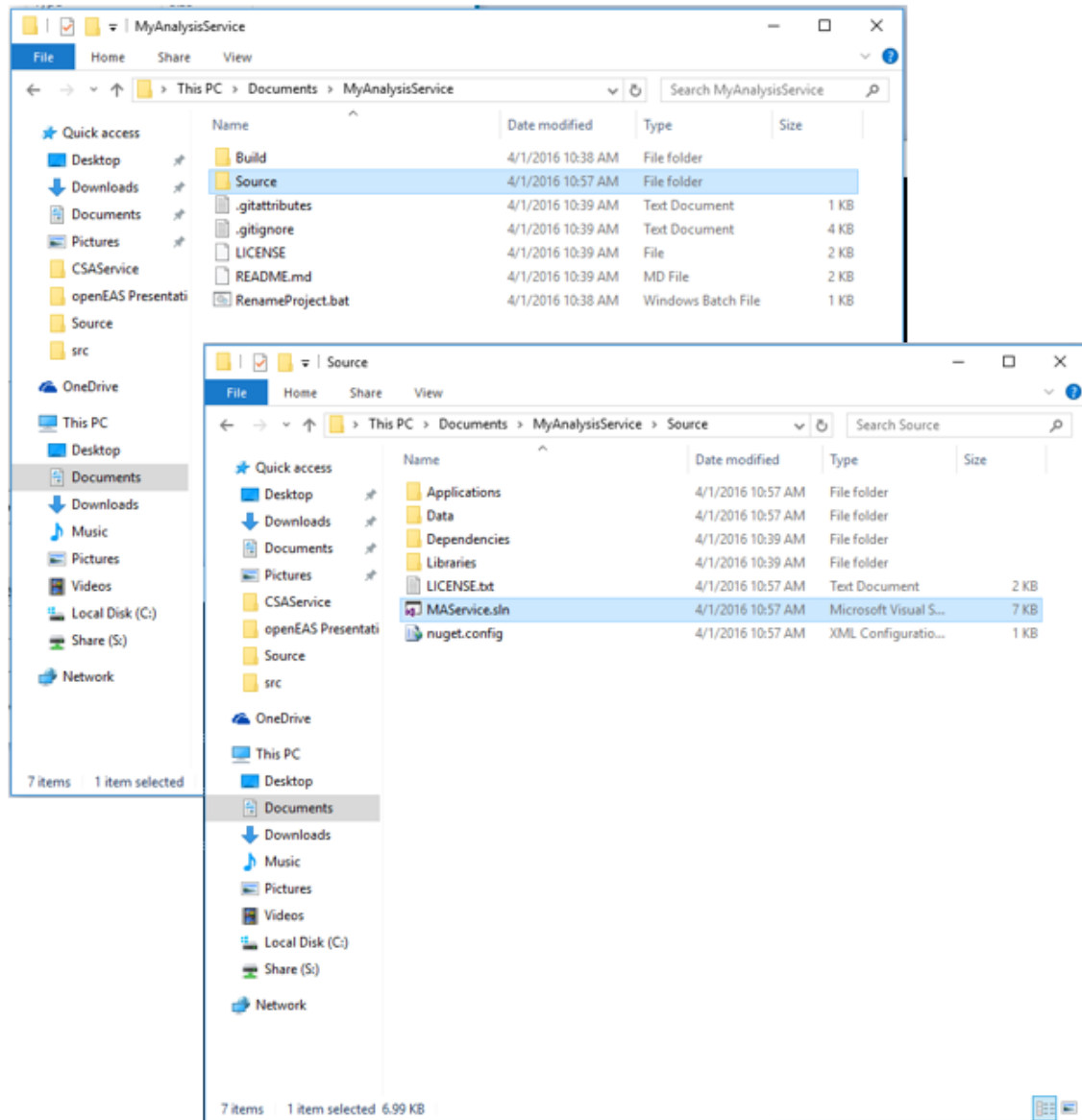


Figure 31 Open Your Project

If your project opened correctly your solution is now ready to be customized using Visual Studio and a window similar to the one below should be displayed. The Solution Explorer will be referenced a number of times in this document and is highlighted with a red outline. If you don't see the Solution Explorer, click View in the menu, then Solution Explorer.

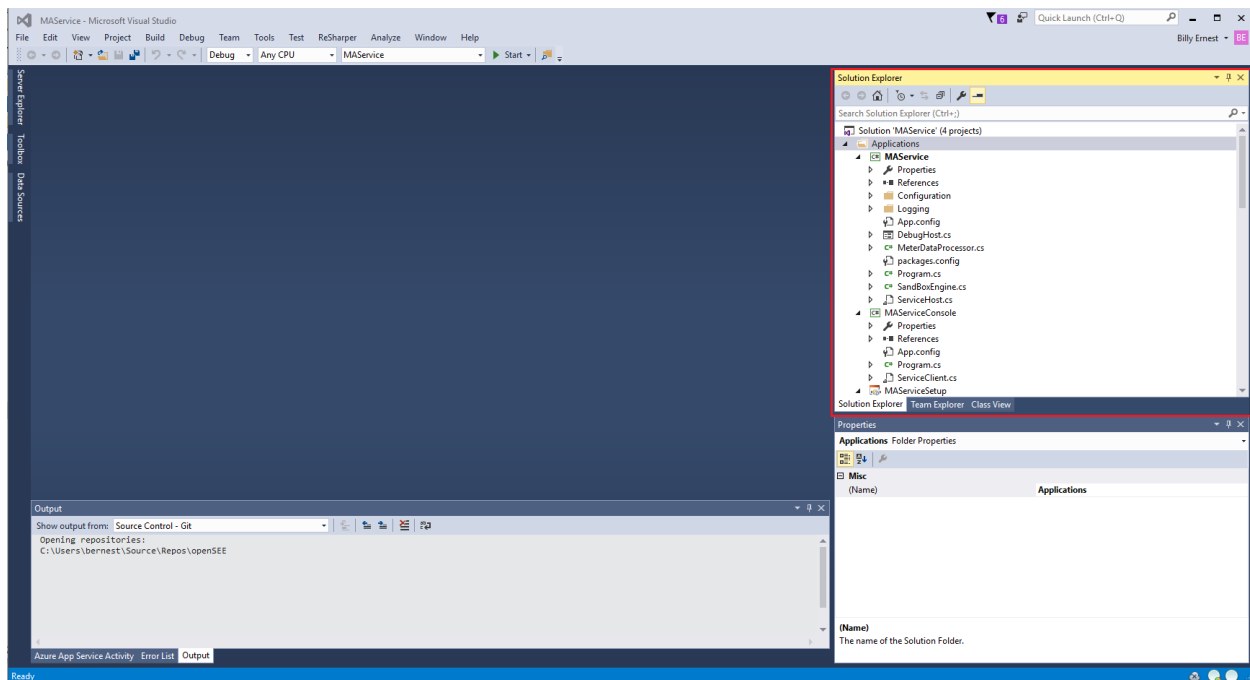


Figure 32 Visual Studio with Solution Explorer highlighted

Step 5. Modify Port Assignment

MAService

The openEAS template is designed to facilitate the creation of any number of services to interact with openXDA. Each service must communicate with openXDA using a unique port address. The port is assigned for your service in the App.config file.

To change the port for your service, open the App.config file under your service e.g. MAService, by double clicking on the name in the Solution Explorer window as shown below. This will open the file for editing in the Visual Studio main window.

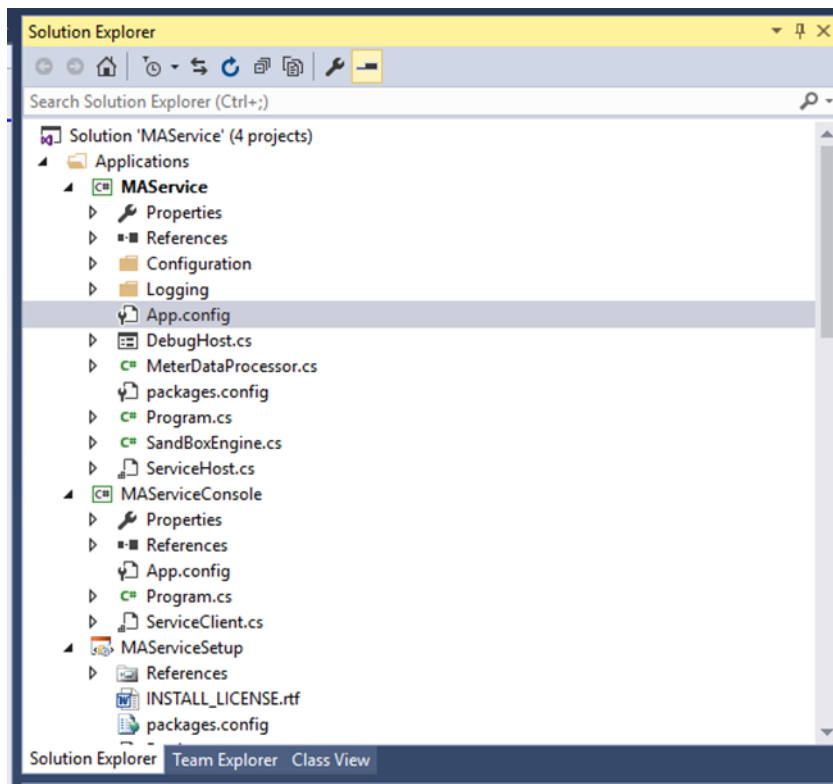


Figure 33 Solution Explorer >> MAService>> App.config

Edit line 11 and change “Port=12623” to the port number of your choice, e.g. “Port=12345”.



Figure 34 Edit Port Number in Line 11

MAServiceConsole

A console is provided to observe the service operation. The console must be configured to use the same port number as specified above for the service. To modify this setting, go to the Solution Explorer window and open the App.config file located under MAserviceConsole by double clicking on the file name as shown below.

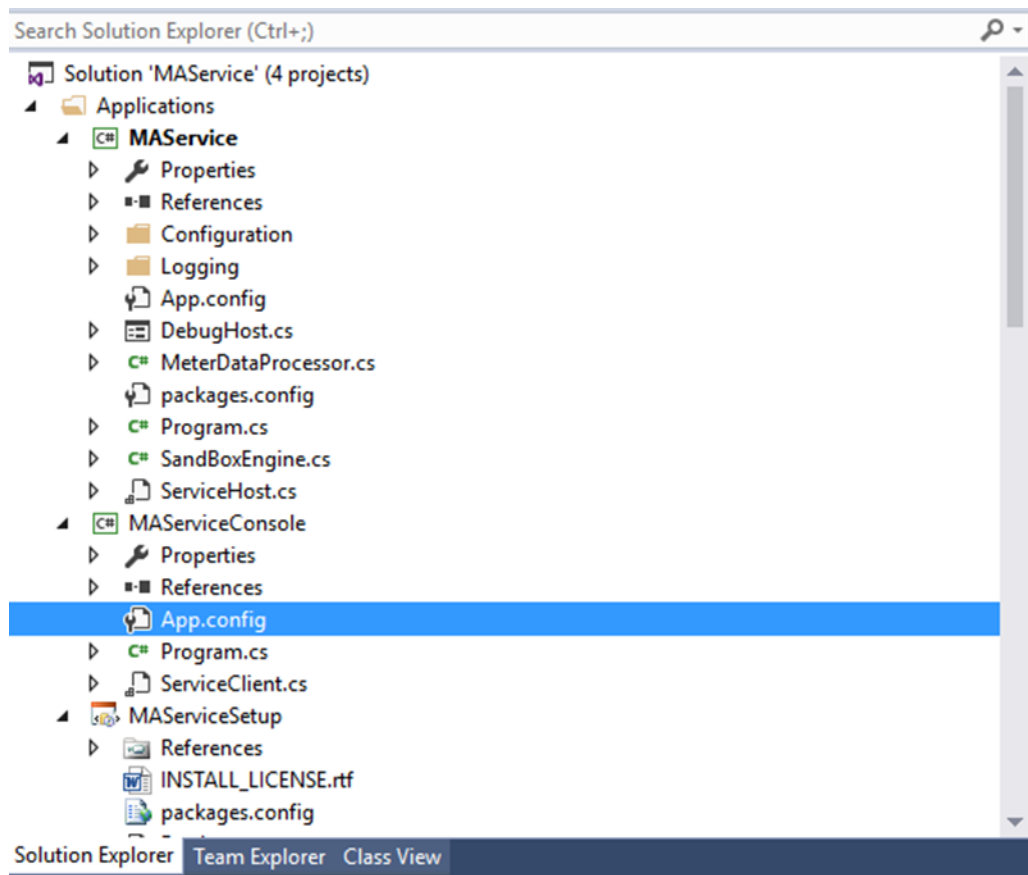


Figure 35 Solution Explorer >> MAserviceConsole >> App.config

Edit line 8 and change “Port=12623” to the port number of your choice, e.g. “Port=12345”. Be certain that this number matches the number used in the MAservice port assignment.

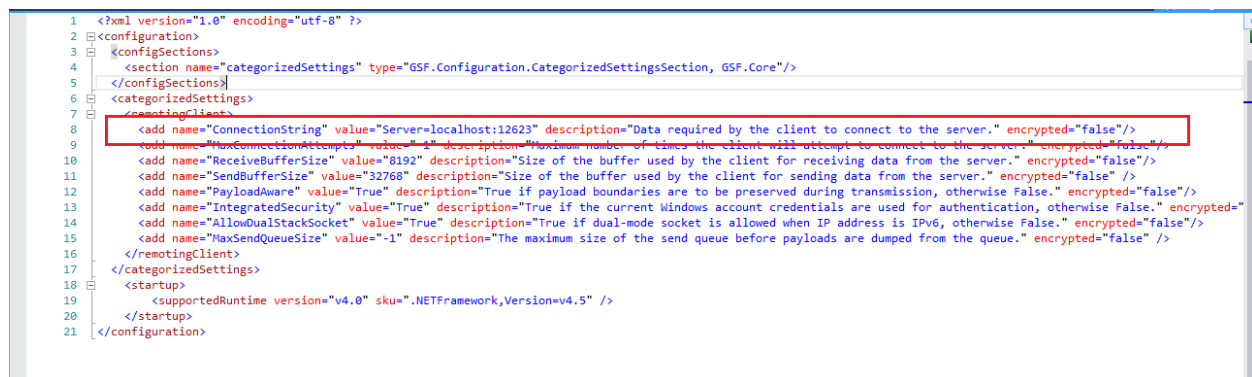


Figure 36 Edit port number in line 8

Step 6 Update Sandbox.sql file

A database table must be provided for your output data. The MAService.sql file sets up that data table. Additional changes must be made to this file if output will be displayed in the Open PQ Dashboard. To edit the file, double click on Solution Explorer >> Data >> MAService.sql as shown below.

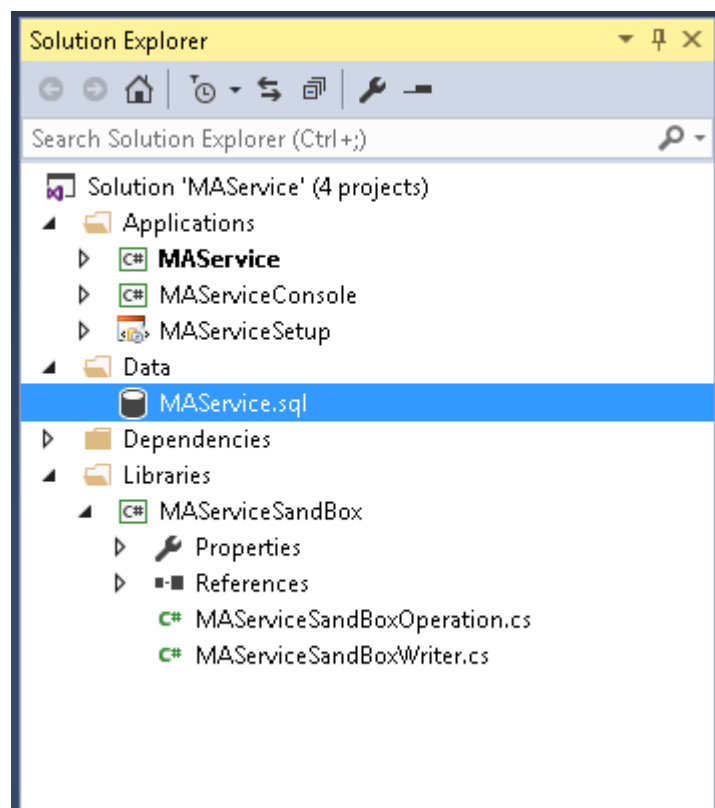


Figure 37 Solution Explorer >> Data >> Sandbox.sql

The database table for output data in this document will be called MyResult. You may choose any table name. Edit lines 5 and 7 to specify the output data table name as shown below.

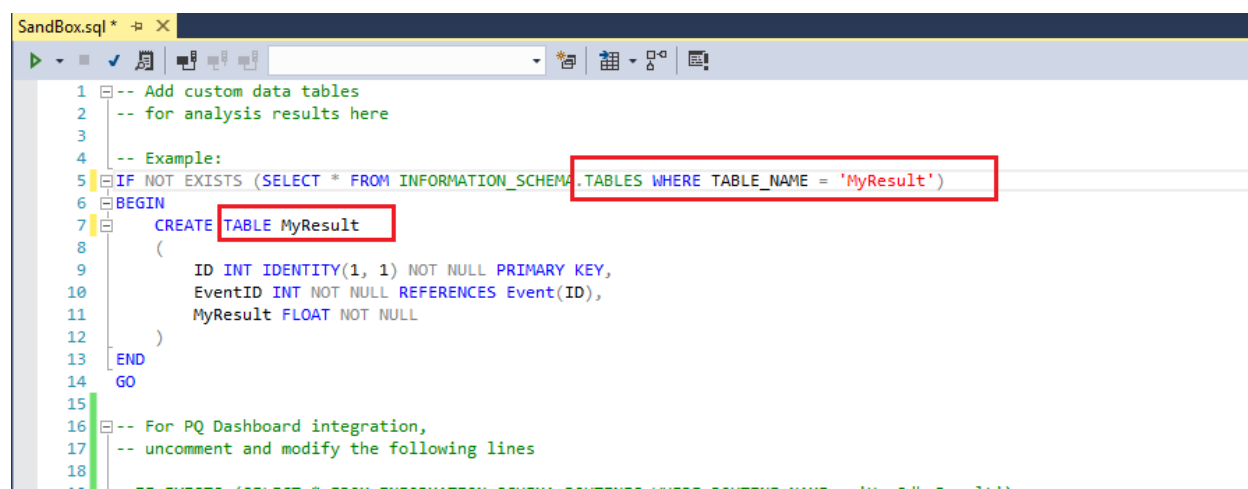


Figure 38 Specify output data table name

If PQ Dashboard integration is not planned, skip to Step 7

For PQ Dashboard integration, edit the 'HasMAServiceResult' function as shown below.

- Uncomment the function by deleting all the '--' from the beginning of each line
- Change the name of the stored procedure as noted
- Change the name of the output table as noted

After editing, the function can be used as-is or can be modified to return a value based on some other specified event criteria.

Uncomment:

```
-- For PQ Dashboard integration,
-- uncomment and modify the following lines

--IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME = 'HasMAServiceResult')
--BEGIN
--  DROP FUNCTION HasMAServiceResult
--END
--GO
--
--CREATE FUNCTION HasMAServiceResult
--(
--  @eventID INT
--)
--RETURNS INT
--AS BEGIN
--  DECLARE @hasResult INT
--
--  SELECT @hasResult = COUNT(*)
--  FROM MAServiceResult
--  WHERE EventID = @eventID
--
--  RETURN @hasResult
--END
--GO
--
--INSERT INTO EASExtension VALUES('MAService', 'HasMAServiceResult')
--GO
```

Change stored procedure name:

```

-- For PQ Dashboard integration,
-- uncomment and modify the following lines

IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME = 'HasMAServiceResult')
BEGIN
    DROP FUNCTION HasMAServiceResult
END
GO

CREATE FUNCTION HasMAServiceResult
(
    @eventID INT
)
RETURNS INT
AS BEGIN
    DECLARE @hasResult INT

    SELECT @hasResult = COUNT(*)
    FROM MAServiceResult
    WHERE EventID = @eventID

    RETURN @hasResult
END
GO

INSERT INTO EASExtension VALUES('MAService', 'HasMAServiceResult')
GO

```

Change output table name:

Edit: FROM MAServiceResult to specify the output table name: FROM MyResult

After changes:

```

16 -- For PQ Dashboard integration,
17 -- uncomment and modify the following lines
18
19 IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME = 'HasMyResult')
20 BEGIN
21     DROP FUNCTION HasMyResult
22 END
23 GO
24
25 CREATE FUNCTION HasMyResult
26 (
27     @eventID INT
28 )
29 RETURNS INT
30 AS BEGIN
31     DECLARE @hasResult INT
32
33     SELECT @hasResult = COUNT(*)
34     FROM MyResult
35     WHERE EventID = @eventID
36
37     RETURN @hasResult
38 END
39 GO
40
41 INSERT INTO EASExtension VALUES('XDASandBox', 'HasMyResult')
42 GO
43

```

Figure 39 Edited MAService.sql file to enable PQ Dashboard integration

Step 7 Your Code: Functions or External dlls

The purpose of the openEAS template is to facilitate the development of a Windows service that establishes links between openXDA and external analytic services. This template has a simple code framework in Solution Explorer >> Libraries >> Sandbox >>

MAServiceSandBoxOperations.cs as shown below. It is the user's responsibility to modify this file to include code for processing functions or external dlls and placing the results back into the database.

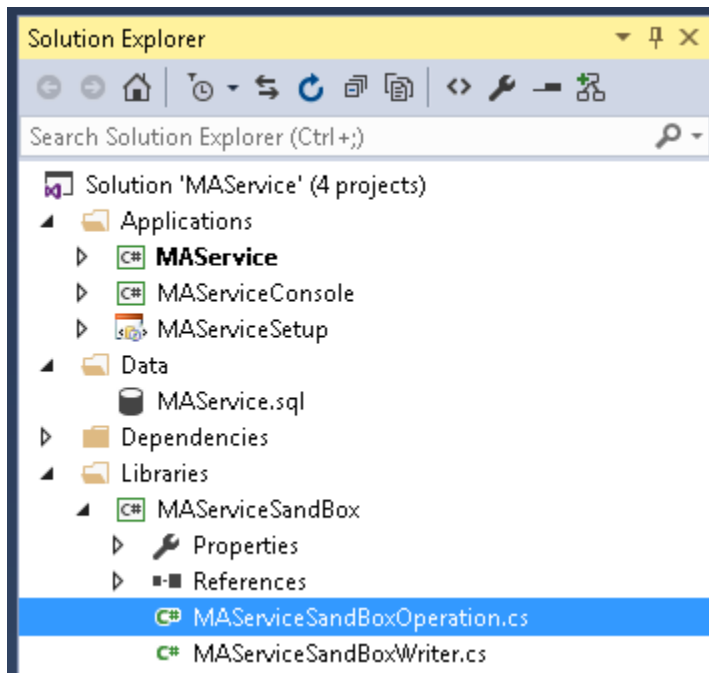


Figure 40 Location of MAServiceSandBoxOperation.cs

```

using FaultData.Database;
using FaultData.DataOperations;
using FaultData.DataSets;
using log4net;

namespace MAServiceSandBox
{
    3 references
    public class MAServiceSandBoxOperation : DataOperationBase<MeterDataSet>
    {
        1 reference
        public override void Prepare(DbAdapterContainer dbAdapterContainer)
        {
            // Prepare for data analysis
        }

        0 references
        public override void Execute(MeterDataSet meterDataSet)
        {
            Log.InfoFormat("Processing {0} waveforms from {1}...", meterDataSet.DataSeries.Count, meterDataSet.Meter.Name);

            // Execute data analysis
        }

        1 reference
        public override void Load(DbAdapterContainer dbAdapterContainer)
        {
            int resultsCount = 0;

            // Write analysis results to the database

            Log.InfoFormat("{0} results written to the database.", resultsCount);
        }

        // Used for logging messages
        private static readonly ILog Log = LogManager.GetLogger(typeof(MAServiceSandBoxOperation));
    }
}

```

Figure 41 MAServiceSandBoxOperation.cs to be modified with user developed code

The Prepare statement is generally used to prepare the connection to the database for the subsequent loads.

```

public override void Prepare(DbAdapterContainer dbAdapterContainer)
{
    // Prepare for data analysis
}

```

Figure 42 The Prepare statement

The Execute function takes input data and passes it as the meterDataSet to an internal function or external dll, then positions the resulting data to be loaded into the database.

```

public override void Execute(MeterDataSet meterDataSet)
{
    Log.InfoFormat("Processing {0} waveforms from {1}...", meterDataSet.DataSeries.Count, meterDataSet.Meter.Name);

    // Execute data analysis
}

```

Figure 43 Execute function

The Load function will load the results positioned by the Execute function into the database.

```

public override void Load(DbAdapterContainer dbAdapterContainer)
{
    int resultsCount = 0;

    // Write analysis results to the database

    Log.InfoFormat("{0} results written to the database.", resultsCount);
}

```

Figure 44 The Load function

If data from the openEAS service will not be used in the PQ Dashboard, implementation is now complete. Data from the openEAS service will be placed in the database whenever openXDA processes appropriate input data.

If data from the openEAS service will be displayed in the PQ Dashboard, continue with Step 8 below.

Step 8 PQ Dashboard Display

Data stored in the openXDA database from MAService can be displayed in the PQ Dashboard. This section describes the process at a high level. The user is responsible for developing specific details.

To create a webpage for MAService data, locate the file EASDetailsTemplate.aspx. By default, it is located in the PQDashboard install folder: C:\inetpub\wwwroot\PQDashboard. Rename a copy of EASDetailsTemplate.aspx to MASDetail.aspx as shown below.

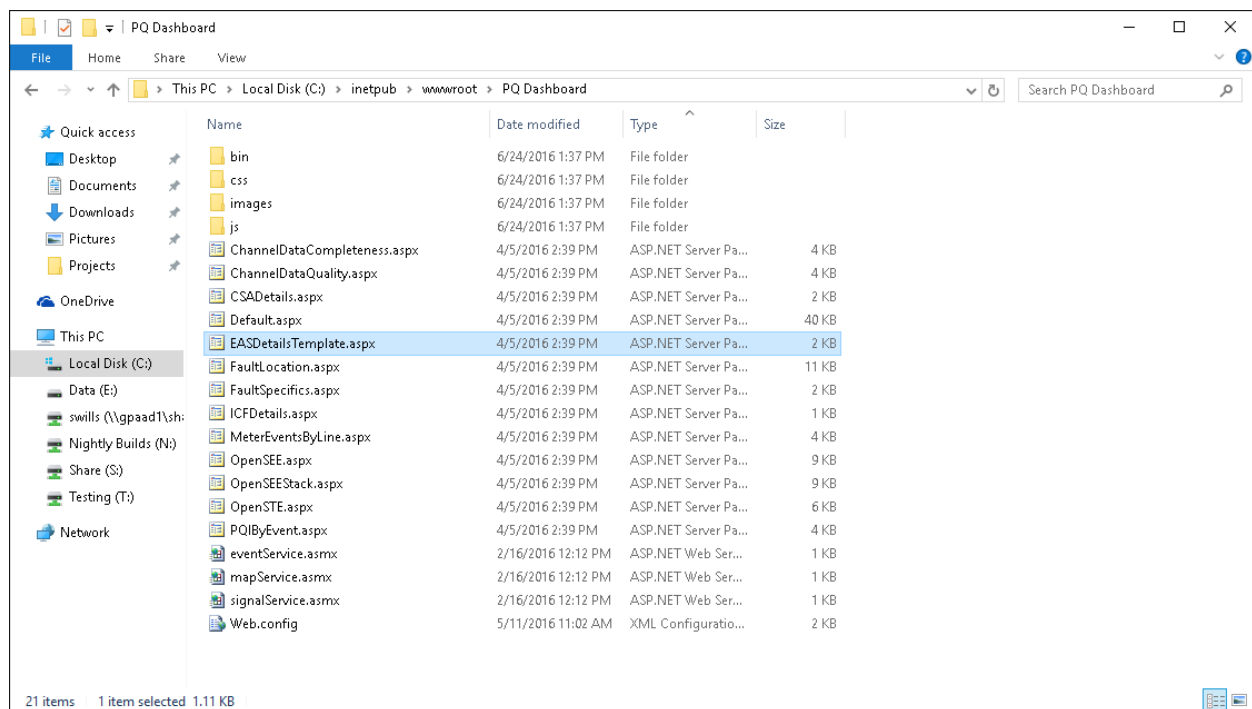


Figure 45 Location of EASDetailsTemplate.aspx

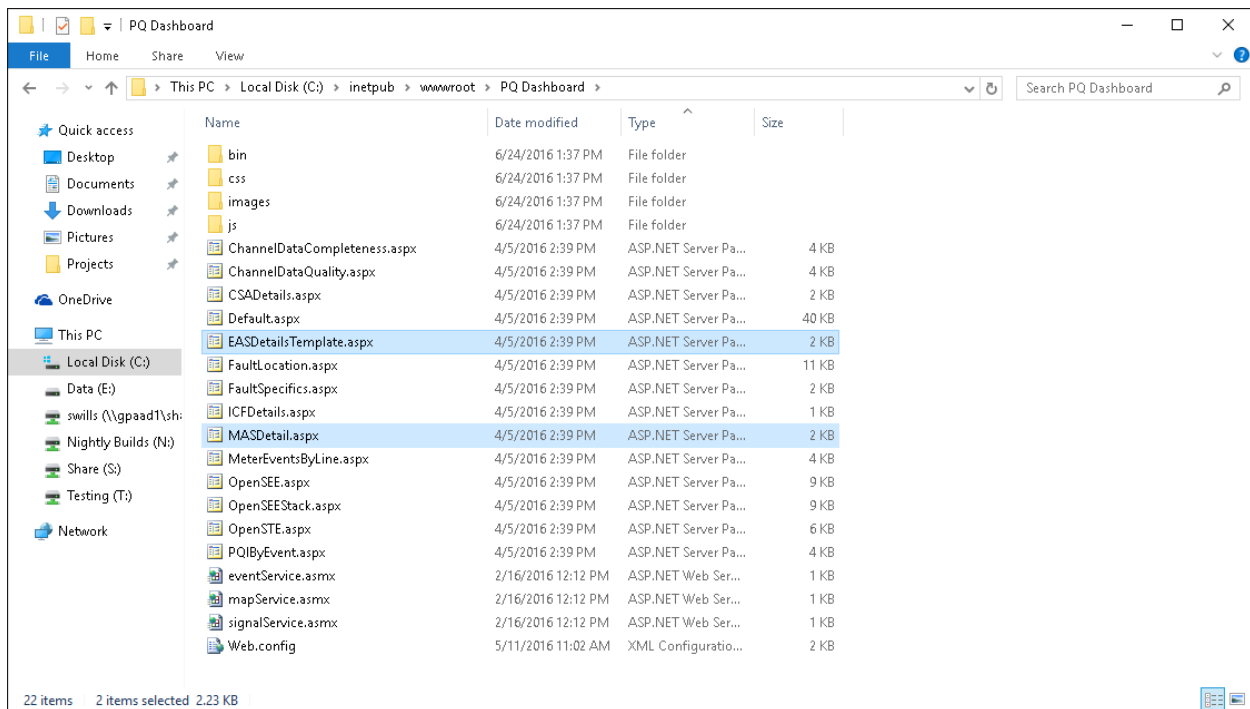


Figure 46 Location of MASDetail.aspx

Edit MASDetail.aspx and change lines 29 and 30 as shown below.

```

21      <table border="1">
22
23          <tr><td nowrap align="right"><%= entry.Key %>:</td><td nowrap><%= entry.Value %>
24
25          <%= %> %>
26      </table>
27
28      <form runat="server">
29          <div runat="server" id="ServiceName" >EAS Details</div>
30          <div runat="server" id="TableName">SandBoxResults</div>
31      </form>
32  </body>
33  </html>

```



```

23      <tr><td nowrap align="right"><%= entry.Key %>:</td><td nowrap><%= entry.Value %></td></tr>
24
25      <%= %> %>
26  </table>
27
28  <form runat="server">
29      <div runat="server" id="ServiceName" >MAS Details</div>
30      <div runat="server" id="TableName">MyResults</div>
31  </form>
32  </body>
33  </html>

```

Figure 47 Edit webpage references

An Icon will be added to the PQDashboard to indicate information from MASService.

Place an icon in the PQDashboard images folder. By default that folder is C:\inetpub\wwwroot\PQ Dashboard\images. The image in this example is mas.png as shown below.

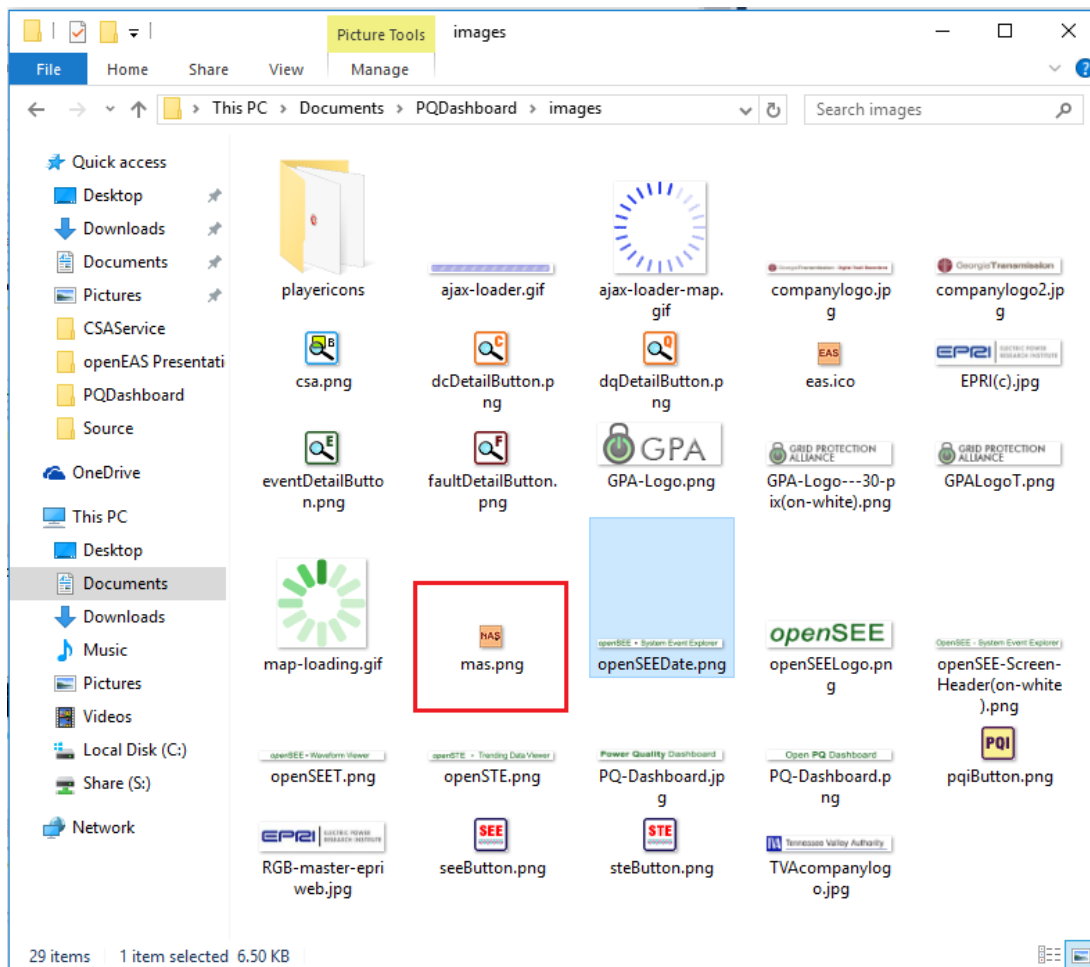


Figure 48 Place Icon in Images Folder

Javascript file modifications are necessary to display information from the MAS service in the PQ Dashboard. Locate the Open MeterEventsByLine.js file as shown below. By default it is located in C:\inetpub\wwwroot\PQ Dashboard\js

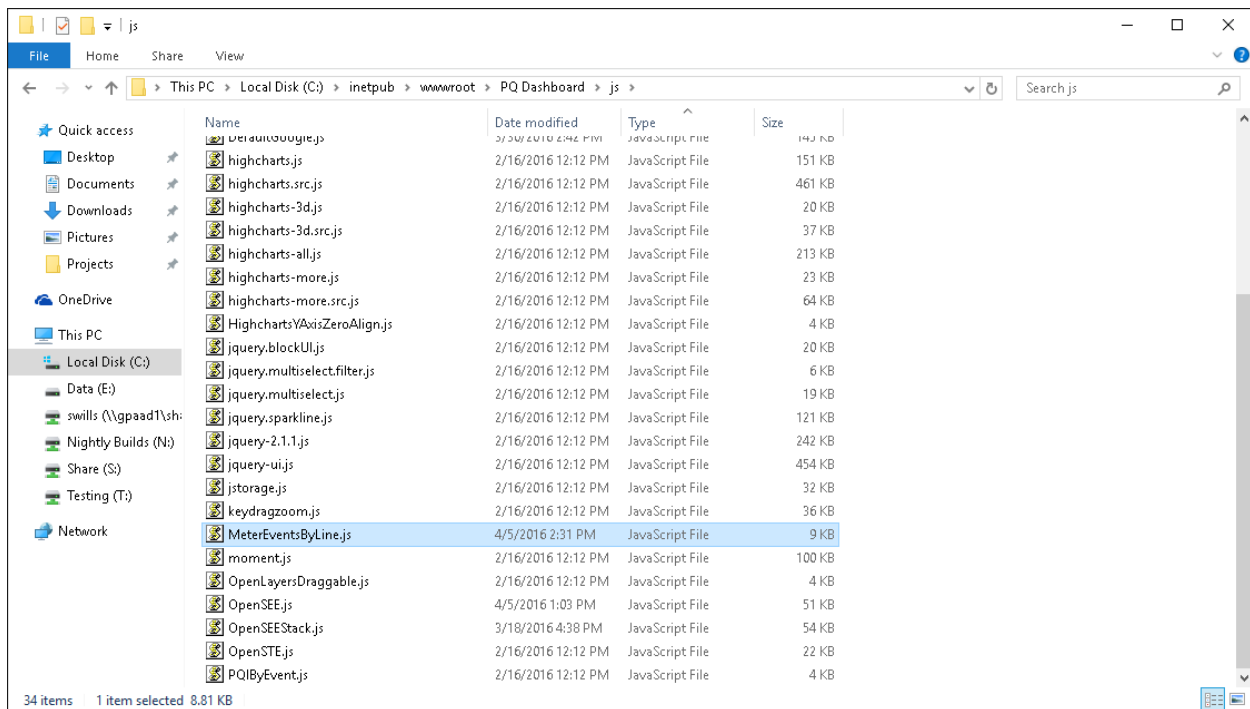


Figure 49 Default location of MeterEventsByLine.js

Edit the Open MeterEventsByLine.js file to include a datafield description for MAService as shown below. To make the changes indicated, copy line 58 to line 59, add a comma to the end of line 58, and change EASService to MAService in line 59.



Figure 50 Update Datafield name in JS file

Add a column in the PQ Dashboard detail display panel to show the MAService icon by making these changes.

- Copy Lines 82 through 85 and Paste them it directly below on line 86
- Change EASDetails.aspx to MASdetail.aspx
- Change eas.ico to mas.ico
- Change Launch EAS Details Page to Launch MAS Details Page

- Change 300 to 320 to increase display width to include new icon
- Change 200 to 220 to increase display height to include new icon
- Change EASService to MAService

See example of changes below.

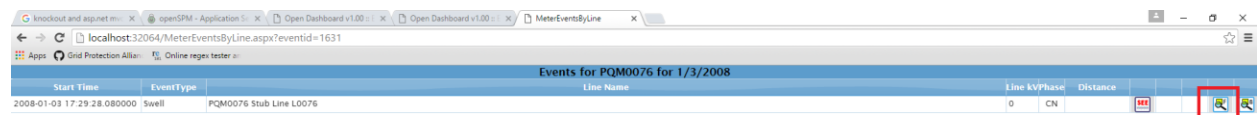
```

1 text: ... , cellrenderer: makeEASButton_html, dataField: 'PQL', width: 40, padding: 0, cellalign: 'left',
2
3 text: ' ', cellrenderer: function (row, __, value) { return makeEASDetailsButton_html(row, value, 'EASDetails.aspx', 'images/eas.ico', 'Launch EAS Details Page', 300, 200) }
4 , dataField: 'EASService', width: 40, padding: 0, cellalign: 'left'
5
6
85
86 {
87     text: ' ', cellrenderer: function (row, __, value) { return makeEASDetailsButton_html(row, value, 'MASDetails.aspx', 'images/mas.png', 'Launch MAS Details Page', 300, 200) }
88     , dataField: 'MAService', width: 40, padding: 0, cellalign: 'left'
89 }
90
91 }



```

Figure 51 JS file edits to allow display of MAS icon

Implementation of the MAService from the openEAS template is now complete. As appropriate data is processed by openXDA, the MyResult data table will be populated. The information in the table may be accessed directly through database calls, or may be displayed through the PQ Dashboard if all of the steps were followed as described above. In the PQ Dashboard, the MAS icon will appear on the right side of the Events Detail page. Clicking on the MAS icon will bring up the MAS results details in a pop-up window similar to the ICF Details shown below.



The screenshot shows a web browser window with the URL `localhost:32064/MeterEventsByLine.aspx?eventId=1631`. The page title is "Events for PQM0076 for 1/3/2008". Below the title is a table with the following data:

Start Time	EventType	Line Name	Line kV/Phase	Distance		
2008-01-03 17:29:28.080000	Swell	PQM0076 Stub Line L0076	0	CN		

The MAS icon is highlighted with a red box in the original image.

Figure 52 Location of MAS Icon

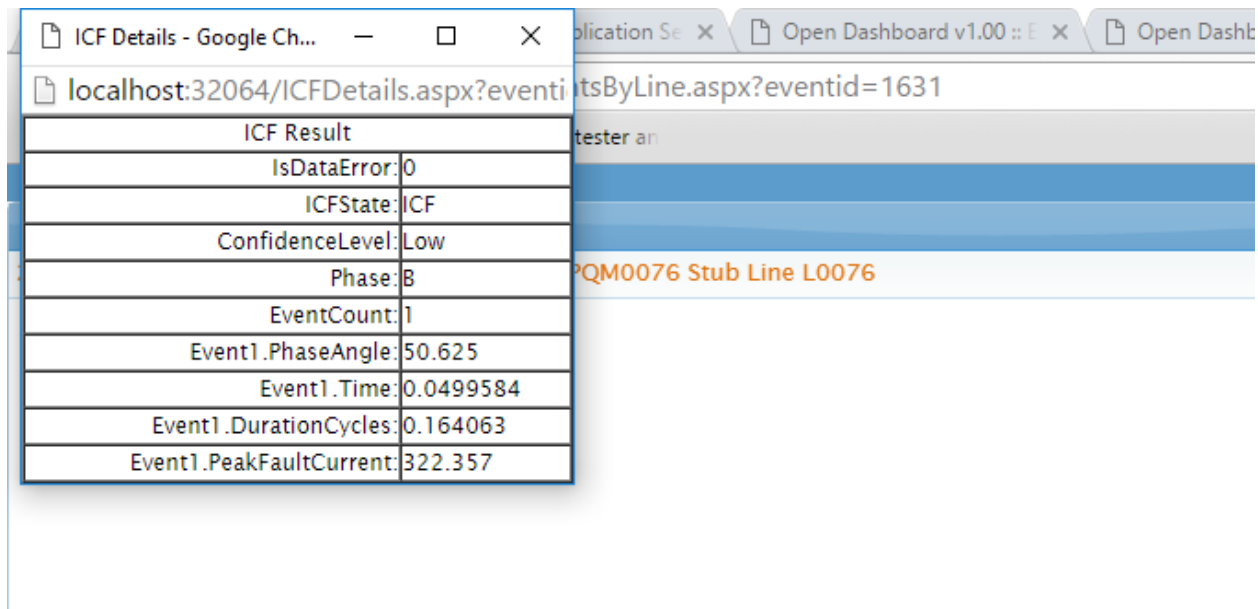


Figure 53 Example display similar to MAS output details

11

PQI INTEGRATION

Description

Power Quality Investigator (PQI) integration with openXDA and the Open PQ Dashboard provides a way to automatically compare voltage disturbances in recorded data from substation devices to ride through curves in PQI databases. A list of equipment that is “possibly effected” by an event is made available in openXDA data tables and can be displayed through the Open PQ Dashboard. Drill-down capabilities in the dashboard allow quick navigation to full resolution details of the disturbance data.

Step 1: Set up database permissions

NOTE: The integration between databases requires database administrator (dba) skills and access privileges.

To execute queries against the PQI databases, the PQ Dashboard will need db_datareader privileges to both the IndustrialPQ database and the UserIndustrialPQ database. The login created by the PQ Dashboard installer is called “PQDashboard”.

If the PQI databases are on a different database server than the openXDA database, you will need to create a “PQDashboard” login on the PQI database server, assign db_datareader

privileges to the new login, then set up a linked server on the openXDA database server using the following SQL commands, replacing **PQIServer** and **PQIPassword** as shown below with the appropriate server name and password for the PQI database server.

```
EXEC sp_addlinkedserver PQInvestigator, N'', N'SQLNCLI', N'PQIServer'
GO
EXEC sp_addlinkedsrvlogin PQInvestigator, 'FALSE', [PQDashboard], [PQDashboard],
[PQIPassword]
GO
```

Step 2: Execute the PQI Integration script

The PQI Integration script is used to alter one function and one stored procedure in the openXDA database so that they can query the appropriate data from the PQ Investigator database. The script can be viewed and downloaded from GitHub.

If you are using a linked server to access PQI databases:

[https://github.com/GridProtectionAlliance/openXDA/raw/master/Source/PQI%20Integration%20\(Linked%20Server\).sql](https://github.com/GridProtectionAlliance/openXDA/raw/master/Source/PQI%20Integration%20(Linked%20Server).sql)

If you are NOT using a linked server:

<https://github.com/GridProtectionAlliance/openXDA/raw/master/Source/PQI%20Integration.sql>

In order to enable the integration between openXDA and PQ Investigator, simply execute this SQL script on the PQ Dashboard database server.

Step 3: Populate the MeterFacility table

The openXDA database has a table called MeterFacility which contains mappings between the openXDA Meter table and the PQ Investigator Facility table. Each mapping can be viewed as a representation of the fact that a Meter is monitoring equipment at a Facility. The mappings are defined in terms of the ID column of the Meter table and the FacilityID column of the Facility table. The following query can be used to add a MeterFacility mapping by specifying the meter name and facility name.

If you are using a linked server to access PQI databases:

```
INSERT INTO MeterFacility
SELECT Meter.ID, FacilityID
FROM Meter CROSS JOIN PQInvestigator.UserIndustrialPQ.dbo.Facility
WHERE
    Meter.Name = 'My Meter Name' AND
```

```
FacilityName = 'My Facility Name'
```

If you are NOT using a linked server:

```
INSERT INTO MeterFacility
SELECT Meter.ID, FacilityID
FROM Meter CROSS JOIN UserIndustrialPQ.dbo.Facility
WHERE
    Meter.Name = 'My Meter Name' AND
    FacilityName = 'My Facility Name'
```

PQI – openXDA – Open PQ Dashboard Integration Complete

If the three steps above completed correctly, the openXDA database will now contain a possibly effected equipment list for every voltage disturbance. That data will now be visible in the Open PQ Dashboard.