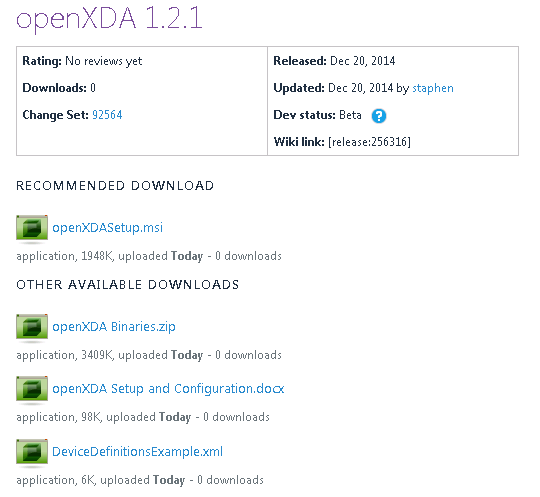
openXDA Setup and Configuration

# Install openXDA

1. Download **openXDA.msi** from <https://openxda.codeplex.com/releases/view/256316>.  
   
2. Run openXDA.msi and follow the instructions.

# Set up the database

1. Find **openXDA.sql**, located in the openXDA installation folder (**C:\Program Files\openXDA** by default).
2. Execute **openXDA.sql** in SQL Server Management Studio to create a SQL Server database for openXDA.
3. Currently, the easiest way to configure meters is to use the openFLE Device Definitions Files.

# DeviceDefinitionsMigrator

1. Open a command prompt by running **cmd.exe**.
2. Use **cd** to navigate to the openXDA installation folder (**C:\Program Files\openXDA** by default).  
    cd "C:\Program Files\openXDA"
3. The DeviceDefinitionsMigrator needs a connection string to connect to the database as well as the path to the DeviceDefinitionsFile.
   1. Example 1: DeviceDefinitionsMigrator "Data Source=MyServer; Initial Catalog=openXDA; Integrated Security=SSPI" "C:\Program Files\openFLE\DeviceDefinitions.xml"
   2. Example 2: DeviceDefinitionsMigrator "Data Source=MyServer; Initial Catalog=openXDA; User Id=admin; Password=adminpwd" "C:\Program Files\openFLE\DeviceDefinitions.xml"

# Configuration File

1. Find **openXDA.exe.config** located in the openXDA installation folder (**C:\Program** **Files\openXDA** by default).
2. Edit **openXDA.exe.config** in a simple text editor, such as Notepad. Note that you may need to run the text editor as administrator in order to save the file.
3. Locate the **ConnectionString** parameter under **systemSettings**. If you just installed openXDA, this should be the only parameter in the file.
4. Modify this parameter so that the service knows how to connect to the database you set up earlier in this process.
   1. Example 1: Data Source=MyServer; Initial Catalog=openXDA; Integrated Security=SSPI
   2. Example 2: Data Source=MyServer; Initial Catalog=openXDA; User Id=admin; Password=adminpwd

# Run the openXDA service

1. Find **openXDAConsole.exe** in the openXDA installation folder (**C:\Program Files\openXDA** by default).
2. Run **openXDAConsole.exe** to monitor the service and ensure that it is running properly.
3. Run **services.msc**.
4. The openXDA requires the **Distributed Transaction Coordinator** to run coordinates its database transactions. Locate the Distributed Transaction Coordinator in the list and make sure that it is running.
5. Locate **openXDA** in the list. Right-click openXDA and select **Start**.

# Drop a fault record

1. By default, the openXDA will monitor a folder called **Watch** in the openXDA installation folder (**C:\Program Files\openXDA** by default).
2. By default, the openXDA is configured such that the watch folder should contain a folder for each meter where the name of the folder is the same as the **AssetKey** of the meter stored in the openXDA database.
   1. SELECT AssetKey, Name FROM Meter
3. Drop the fault record into the folder associated with the meter that produced the record.
4. If the **openXDAConsole** is running, openXDA should report that it has found a fault record and indicate when it has completed its analysis.
5. Once finished, results from the fault analysis will be located in the results folder (**C:\Program Files\openXDA\Results** by default), in a subfolder named after the asset key of the meter that produced the fault record.

# Service Configuration

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.

* **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> for more information about regular expression in .NET.
* **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: 8.0
  + The per-unit threshold at which the current exceeds engineering reasonableness.
  + Events with data that exceeds engineering reasonableness will be excluded from fault analysis.
* **LowVoltageThreshold**
  + Default: 0.5
  + The per-unit threshold at which the voltage is classified as a low voltage.
  + If the voltage during a cycle is considered low, it is assumed that they will not be able to serve more than a specified amount of current.
* **MaxLowVoltageCurrent**
  + Default: 1.0
  + The per-unit threshold at which the current exceeds engineering reasonableness when the voltage is low.
  + If the current exceeds this threshold during a cycle that is considered low voltage, it is determined that this exceeds engineering reasonableness and will be excluded from fault analysis.
* **MaxTimeOffset**
  + Default: 24.0
  + 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.
* **MinTimeOffset**
  + Default: 1440.0
  + 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.
* **ResidualCurrentTrigger**
  + Default: 0.5
  + The per-unit threshold at which the residual current indicates faulted conditions.
  + If the residual current exceeds this value during any given cycle, the data in that cycle is considered to have been recorded during a fault.
* **PhaseCurrentTrigger**
  + Default: 4.0
  + The per-unit threshold at which the phase currents indicate faulted conditions.
  + If any of the phase currents exceed this value during any given cycle, the data in that cycle is considered to have been recorded during a fault.
* **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.
* **FaultSuppressionTrigger**
  + Default: 0.2
  + The threshold at which the distance between the median and mean indicates the sine wave is no longer pure and that fault detection logic should be suppressed.
  + If the RMS current is very high compared to the prefault current during any given cycle, that cycle is considered to have been recorded during a fault unless the fault suppression logic indicates otherwise.
* **MaxFaultDistanceMultiplier**
  + Default: 1.05
  + The multiplier applied to the line length to determine the maximum value allowed for fault distance in the COMTRADE results file.
  + Fault distances larger than the maximum fault distance will be snapped to the maximum fault distance in order to suppress the noise around the edges of the fault.
* **MinFaultDistanceMultiplier**
  + Default: -0.05
  + The multiplier applied to the line length to determine the minimum value allowed for fault distance in the COMTRADE results file.
  + Fault distances smaller than the minimum fault distance will be snapped to the minimum fault distance in order to suppress the noise around the edges of the fault.
* **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
  + 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.
* **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](file:///\\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**
    - FileShare.1.Name \\server1\share  
      FileShare.1.Username DOMAIN1\USER1  
      FileShare.1.Password user1pwd
    - FileShare.2.Name \\server2\share  
      FileShare.2.Username DOMAIN2\USER2  
      FileShare.2.Password user2pwd
    - Etc.
* **SMTPServer**
  + The hostname or IP address of the SMTP server used for sending emails when a fault is detected.
* **FromAddress**
  + Default: openXDA@gridprotectionalliance.org
  + The email address placed on the From line of the emails sent when a fault is detected.
* **PQDashboardURL**
  + Default: http://pqdashboard/
  + URL to the PQ Dashboard placed in email notifications to direct recipients to a page where they can view the waveforms captured by openXDA.