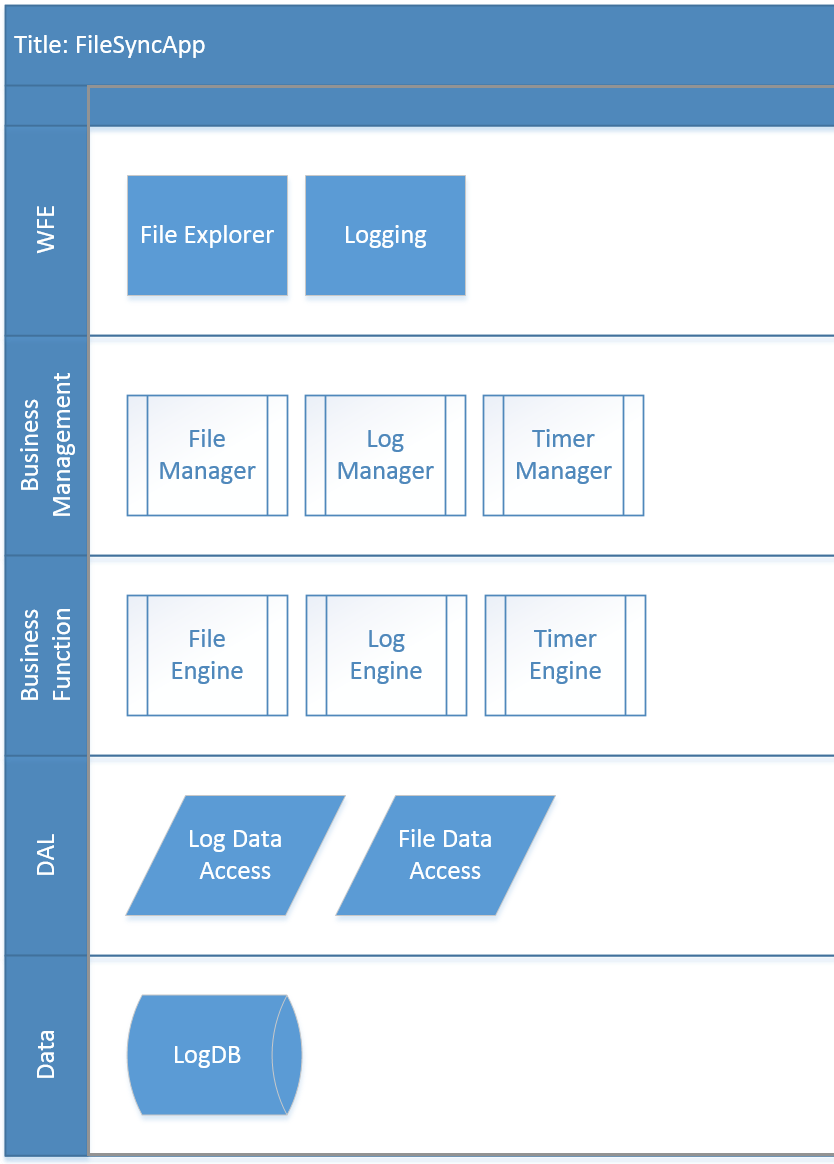
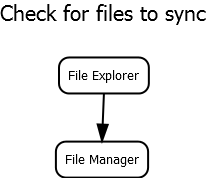
**FileSync Documentation**

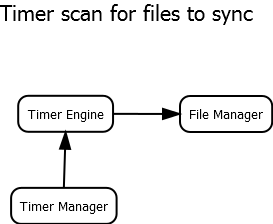
**UML DIAGRAMS**

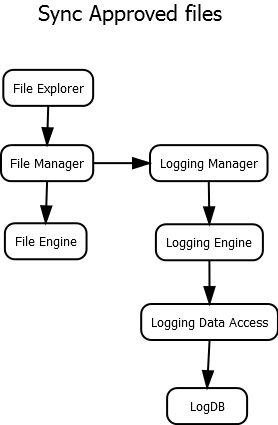
**USE CASES**

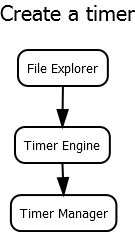
1. Display all files and folders from a single directory
2. Scan a directory for changes
3. Display found directory changes in a list
4. Request user approval to sync the directories.
5. Apply changes to target directory if approved
6. Allow the user to schedule the app to check for changes at user defined intervals.
7. In the event of a folder scan, log date/time, log that it is a scan, and Domain/current user.
8. In the event of a soft sync, log the date/time of the sync, who approved the sync, the names of all modified files for that sync, and the type of file modification.

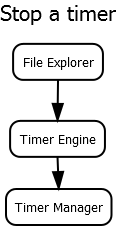
**CALL CHAINS**

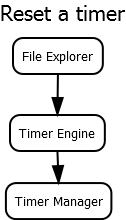


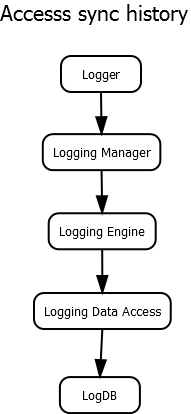












**PUBLIC CONTRACTS**

iFileExplorer();

Web Front End Stuff

iFileManager();

bool FileEngineInit(String sourceDirectory, String targetDirectory);

bool DetectChanges();

bool InitSync(bool fullSync);

bool GetApproval();

iFileEngine();

bool DetectChanges(FileSyncProvider provider);

SyncOrchestrator CreateSyncOrchestrator(FileSyncProvider source, FileSyncProvider destination, SyncOrchestrator agent);

SyncOrchestrator AssignSyncDirection(SyncOrchestrator agent);

bool Synchronize(SyncOrchestrator agent);

FileSyncProvider CreateProvider(string RootPath, FileSyncScopeFilter filter, FileSyncOptions options);

void DisposeProvider(FileSyncProvider provider);

FileSyncOptions SetOptions();

FileSyncScopeFilter CreateFilter();

FileSyncProvider SetPreviewModeFlag(FileSyncProvider provider, bool flag);

FileSyncProvider AttachSkippedChangeEventHandler(FileSyncProvider provider);

FileSyncProvider AttachDetectedChangeEventHandler(FileSyncProvider provider);

FileSyncProvider AttachApplyingCHangeEventHandler(FileSyncProvider provider);

FileSyncProvider AttachAppliedCHangeEventHandler(FileSyncProvider provider);

iTimerManager();

bool InitTimer();

bool CreateTimer(int interval = 10/\*minutes\*/);

bool StopTimer();

bool ResetTimer();

bool DisposeTimer();

iTimerEngine();

TimerLogic();

iLogger();

Web Front End Stuff

iLoggerManager();

**bool InitLogEngine();**

**bool CreateQuery();**

bool CreateSyncEntry();

bool CreateFileChangeEntry();

bool CreateProgramInitEntry();

bool CreateErrorEntry(error e);

**bool CommitLogEntry();**

iLoggerEngine();

IEnumerable<History> CreateQuery();

IEnumerable<History> LogTypeQuery(IEnumerable<History> query, int logType);

IEnumerable<History> DateRangeQuery(IEnumerable<History> query, int start, int end);

IEnumerable<History> FileOperationQuery(IEnumerable<History> query, int operation);

IEnumerable<History> UserNameQuery(IEnumerable<History> query, String name);

bool CommitLog(History log);

List<History> ExecuteQuery(IEnumerable<History> query);

History CreateSyncEntry(int interaction);

History CreateFileChangeEntry(int operation, string filePath);

History CreateProgramInitEntry(int operation, string filePath);

History CreateErrorEntry(string exception, int interaction, string filePath, int operation);

iLoggerDataAccess();

IEnumerable<History> GetQueryObj();

bool CommitLogEntry(History log);

List<History> ExecuteQuery(IEnumerable<History> result);

iLogDB();

History

-Id

-LogTime

-LogDate

-LogTypeId

-Domain

-UserName

-PCName

-FileName

-FileOperationId

-Error Msg

FIleOperation

-Id

-Description

LogType

-Id

-Description

InteractionType

-Id

-Description

Tyler’s magical work-around to get the app to work:

1. Run app
2. Break
3. File browse to QA
4. Enter credentials
5. Re-run app
6. Profit...

**Facts! :/**

www.react.rocks good site for react widgets