# Upcoming

## PADtrax – Assets

PADtrax for Assets was designed to bring asset management together with asset tracking in a system that requires minimal effort to deploy and utilize. It has been designed as a browser application in order to make the application available to users all available operating systems. PADtrax extends usability by making all application forms scale to resolutions adopted by a majority of today's available devices including smart phones and tablets.

The application has been created to resemble a digital storefront, with special effort put into creating an uncluttered environment that allows a user to easily find and manage an asset or group of assets.

PADtrax for Assets has also been designed with scalability in mind. We recognize that tracking and management of assets extends beyond asset location. Therefore, while basic common tenets of asset management are included in the delivered program, an embedded workflow engine allows for each customer to apply custom rules to assets and the activities acted upon them. (Discussed later in document)

At its most basic level, assets can be tracked by a list. A simple list of assets, where they are, and the custodian responsible for the asset is available. Tracking can be extended to barcode if desired by a customer. However, the ultimate goal for tracking an asset in PADtrax is to use RFID (radio frequency identification). By doing so, we can establish a movement history for assets as they move throughout facilities and have more opportunity to locate items quickly by seeking out specific items or running inventory audits by location, etc.

PADtrax for Assets implements a document library which can optionally be integrated with the workflow engine. Documents exist at the asset level so a user can attach instruction manuals, software and drivers, checklists, images, etc. for easy retrieval. Documents also exist at an enterprise level, making these attachments available to processes independently from assets. Such attachments might be useful when they apply to multiple assets managed through the application. Examples of such documents might include repair request forms, receipt forms, traveler documents, receipts, and contracts.

Asset details are designed to be flexible. Traditional asset management systems use a rigid structure for categories and subcategories forcing an asset to be pigeonholed inside a management system. This meant that in order to locate an asset, a user had to already know how an asset was likely to be categorized by someone else in the enterprise. Further, an asset could traditionally only exist in a single thread of categorization. PADtrax allows assets to be identified inside of a parent-child category structure but does not require it. Furthermore, an asset can be identified by many different category threads.

Each asset also has its own listing of attributes and values. Whereas some systems recommend that data be massaged to fit into fields that were not explicitly designed to store information it, PADtrax allows users to specify an unlimited number of attributes and values. This means that a user can distinguish between attributes such as color and background color without having to be bound by complex user interfaces laden with unnecessary fields, or that same user can simply look for assets with an attribute value of "red" and be presented a list of results that specifically identify which attribute for which asset has a value of "red."

Furthermore, asset details allow for assets to be easily identified in multiple ways. Always available on the user screen is a search box that allows a user to search for data in make, model, serial number, vendor, asset name, and the attribute/value listing described above.

PADtrax has also been designed using newer (M)odel(V)iew(C)ontroller development methods. This allows for contexts within the application to be captured in the URL (address) of the application. If user 1 is working on an asset and sees a problem, he can actually send an email to user 2 from inside the application that would look something like this:

http://localhost:7582/AddAsset/Edit_Asset_BasicInfo?AssetId=4

And when user 2 opens the email and clicks on the link, the user will be asked to log into the application and will then be taken directly to the application context specified in the link.

| | | |
|---|---|---|
| Other Asset details include | | |
| | Asset Name | |
| | Make | |
| | Model | |
| | Serial Number | |
| | Description | |
| Purchase details Including | | |
| | Purchase date | |
| | Purchase Vendor | |
| | Purchase Price | |
| | Part Number | |
| | Vendor Part Number | |
| | | |
| Other | | |
| | Substitute Part Number | for fulfilling asset processes |

| | | |
|---|---|---|
| | | with alternate items |
| | Dependent items | to ensure that assets are processed in sets |
| | Service Items | to allow tracking for items that are used in another, such as consumables on a copy machine where a characteristic such as cost is enough to warrant tracking the consumable item on its own |
| Service and Support | | |
| | Contract Details | Can apply different contract types with custom effective dates |

Contracts are an element of assets which can be applied to individual assets at will. For example, I might have a contract type called "Retail return period" that would apply to some of my assets. However, other types of contracts might require more careful management.

Imagine a scenario where we are managing assets at a bank. We have acquired scanners from a vendor, ABC Scanner Supply Company. However, due to inadequate service, we are moving all scanners under a contract with C&A Associates. Start this contract one month from today. We can have multiple contracts applied to these assets with different effective dates. Therefore, if a scanner should need to be sent in for maintenance, the asset manager can see who to call, the contract number covering the scanners, and anything else that might apply, such as advance loaner of replacement agreements, depot maintenance, etc.

## MORE ON WORKFLOW

The PADtrax workflow engine has been put in place to ensure that customers can apply their own processes to the management of their assets. If we think about some of the more easily identifiable things we can do with assets, we can get a relatively finite list.

Acquire asset
Inspect Asset
Maintain Asset
Distribute Asset
Collect Asset
Replace Asset
Retire Asset

However, if we consider the breadth of assets we might be managing, we can easily see how the details of each of these processes could vary greatly from one enterprise to another. Inspection schedules may vary greatly and need to have decisions made based on the feedback of the inspection. Did the inspection pass or fail? What needs to be done if it failed? Can an asset be distributed to everyone or only a certain class of people? Is certification required? How do we record this certification?

Traditionally, a worker would need to be trained how to handle a process and this worker was depended on to follow all enterprise policies completely and accurately By implementing a workflow engine, we can store the process inside the application, store and track the results in the database, and follow a defined set of custom flowcharts to match anyone's needs.

In these workflows, you can restrict custodian types, scan items into the system to attach to the workflow, attach items to the workflow and more.

The intended goal of the workflows is not to have end-users at the customer level to implement custom programming. The goal is to provide access to the core features which exist inside the application so that custom logic can be applied by our development team or distributors. The workflow engine will be continually developed and more core functionality of the application will be enabled for use by the workflow engine as we continue to develop the application.