Using Altium Documentation
Modified by Jason Howie on May 21, 2017

A distinct design solution in its own right, an *Altium Vault* works in harmony with Altium Designer to provide an elegant answer to the question of handling design data with secured integrity. An Altium Vault not only provides rock-solid, secure storage of data, but also enables re-release of data as distinctly separate revisions - essentially tracking design changes over time, without overwriting any previously released data. It also caters for the lifecycle of the data to be managed, allowing people that need to use the data to see, at-a-glance, what stage the data has reached in its 'life', and therefore what it can be safely used for.

The vault becomes both the source and destination of design elements, with each new design utilizing elements realeased to, and managed through, the vault. And by designing only with elements from a vault - vault-driven electronics design as it were - the integrity of those designs is inherently assured.

Read about The Altium Vault. Read about Accessing Your Altium Vault.

Vault Items

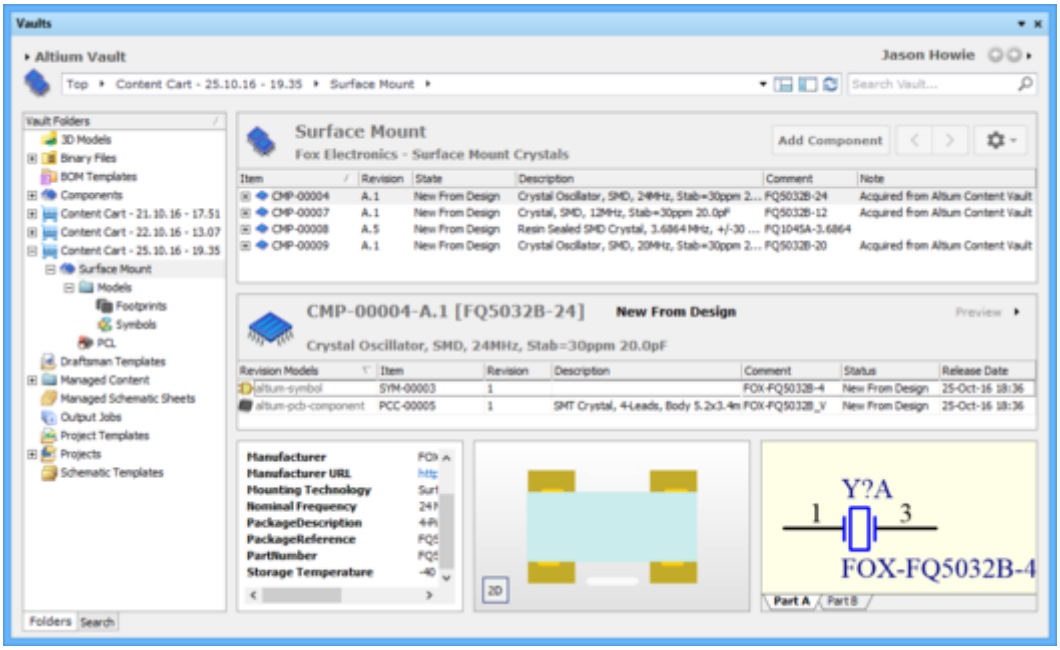
Within an Altium Vault, each design entity that can be stored, managed, and reused, is represented as a specific type of *Item*. An Item is uniquely identified within the vault and can contain any number of *Revisions*, where a revision contains the data for that Item. Each time a change is made to the data contained within a revision - which for most Item types can be edited directly within an associated temporary editor - it is committed (or re-released) into a new revision of that Item, ensuring that no existing revision can ever be overwritten, and thereby ensuring the highest integrity.

An Item can have any number of revisions, which are essentially an evolution of that Item over time. A change is made and the new data content is committed/uploaded/released into a new revision. The data stored in each revision of an item is therefore typically different. To identify between these different revisions of an Item, a revision identifier (ID) is used, which in combination with the Item ID creates a unique identifier for each release of an Item. This gives us the *Item-Revision*.

Another important aspect of an Item Revision is its *Lifecycle State*. This is another identifier that can be used to quickly assess what stage that revision has currently reached in its *life*, and what designers are therefore authorized to do with it. Where the Revision reflects design changes made to the Item, the Lifecycle State reflects the state of the item from a business perspective, such as `Planned`, `New From Design`, `For Production`, `Obsolete`, and so on.

Read about Vault Items.

The Vaults Panel



Providing the direct interface to your Altium Vault is Altium Designer's *Vaults* panel. From this panel you can perform many activities, including:

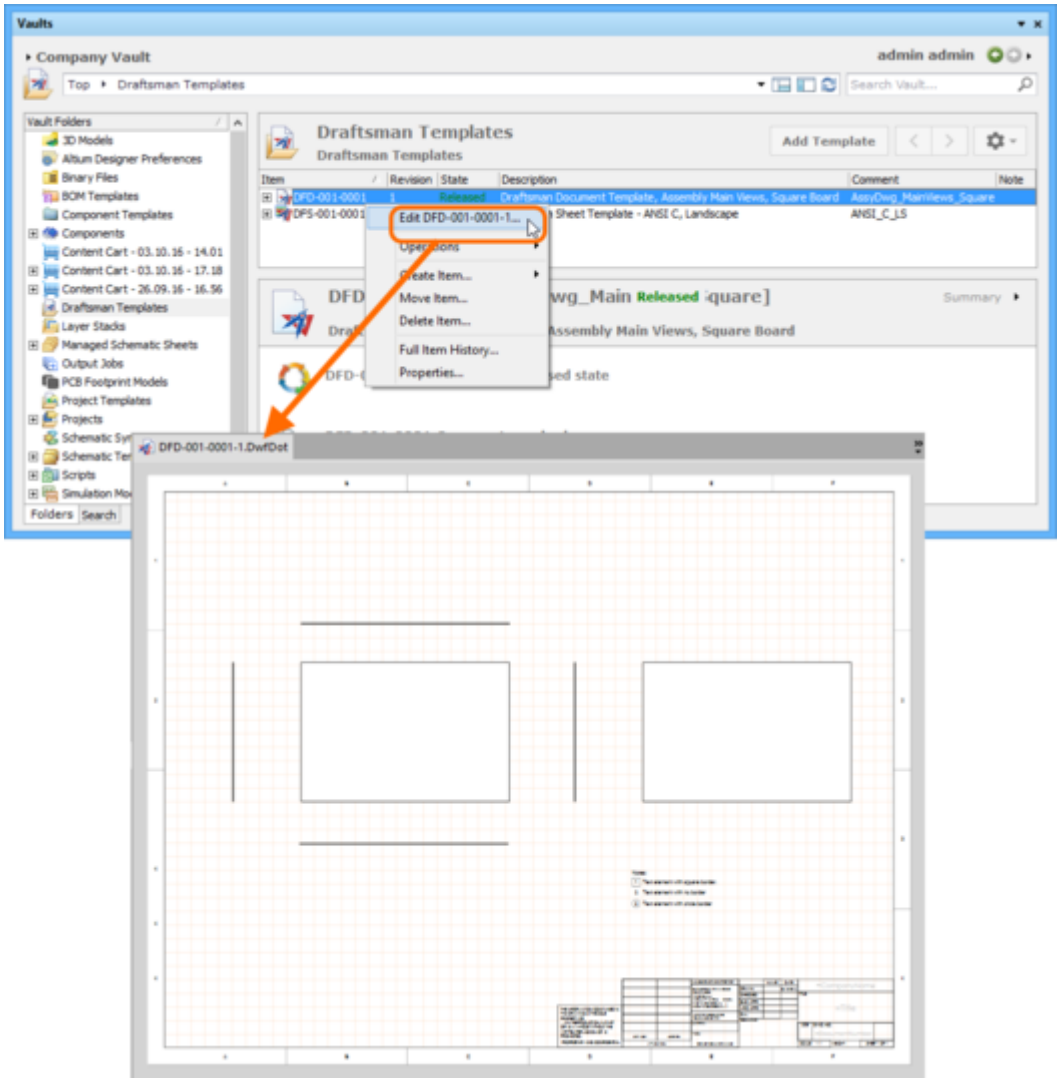
- Creating and managing the organizational structure used in the vault.
- Creating any number of Items, each representative of a design object.
- Direct editing and placement of Item Revisions.
- Reviewing and managing the lifecycle of Item revisions.
- Interrogating the usage of a particular Item revision (Where-Used).
- Browsing and managing supply chain information for Component Items.
- Downloading stored data, including data generated through the managed release of board design projects.

The Vaults panel becomes your trusty right-hand, presenting a collection of features that can really enhance your productivity when working with an Altium Vault through Altium Designer.

> And while the majority of your day-to-day working with an Altium Vault will be through the *Vaults* panel, there will also be occasion when you need to interact with the vault through its browser interface - especially for administrative purposes. For more information, see Browser-based Access & Management of an Altium Vault.

Read about Working with the Vaults Panel.

Direct Editing



Vaults provide a flexible and secure method of centralizing the storage and management of all types of design data used in Altium Designer. From the schematic model to the component, from managed sheets through to completed PCB designs, an Altium Vault delivers an ideal method of storing and managing your electronic design data.

Many design entities can be edited and released into the initial revision of a corresponding, and newly-created vault Item, courtesy of the vault's support for direct editing. Direct editing frees you from the shackles of separate version-controlled source data. You can simply edit a supported Item type using a temporary editor loaded with the latest source direct from the vault itself. And once editing is complete, the entity is released (or re-released) into a subsequent planned revision of its parent Item, and the temporary editor closed. There are no files on your hard drive, no questioning whether you are working with the correct or latest source, and no having to maintain separate version control software. The Altium Vault handles it all, with the same great integrity you've come to expect, and in a manner that greatly expedites changes to your data.

And at any stage, you can come back to any revision of a supported Item in the vault, and edit it directly. Simply right-click on the revision and choose the **Edit** command from the context menu. Once again, the temporary editor will open, with the entity contained in the revision opened for editing. Make changes as required, then commit the release of the document into the next revision of the item.

Read about Creating & Editing Items Directly in an Altium Vault.

Controlled Access

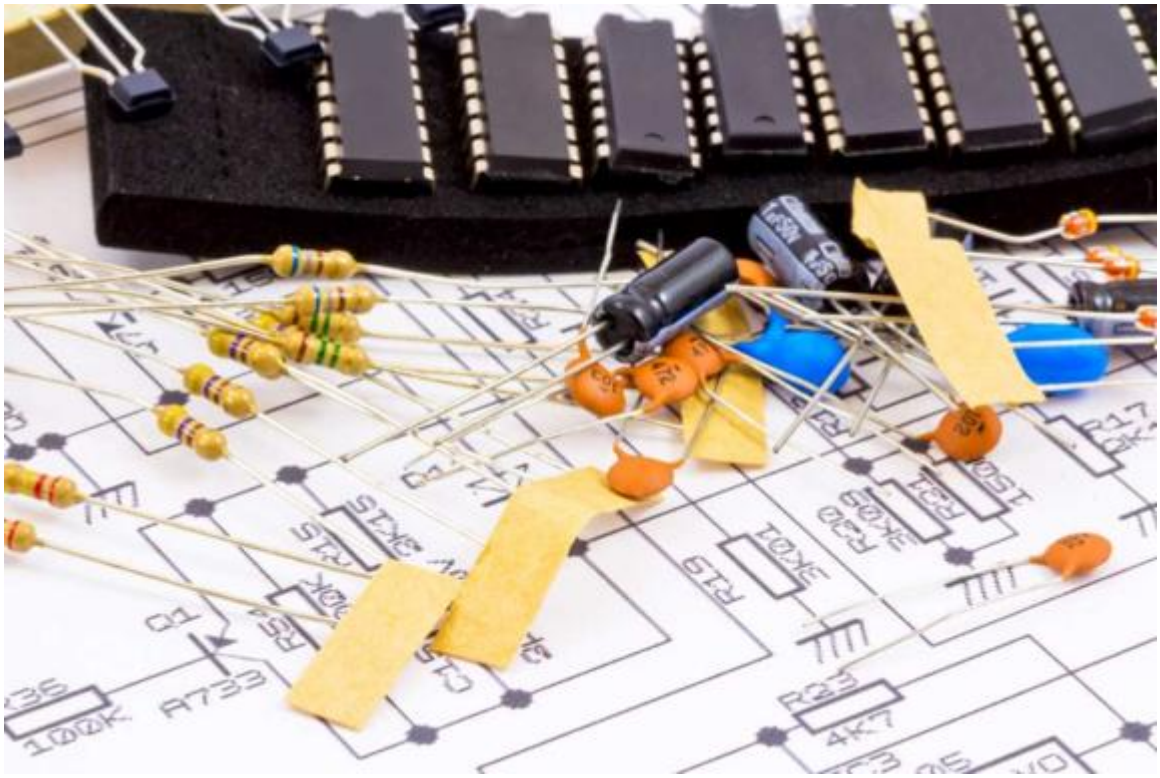
An Altium Vault provides secure handling of data with high integrity, while providing both Design Team and Supply Chain access to that data as needed. This latter aspect, of whom can access a vault, and more importantly what data they are allowed to access, is facilitated by the Altium Vault's user access control and sharing capabilities. These can be broken down into the following key areas:

- User Management
- Folder-level Sharing
- Item-level Sharing
- Item Revision-level Sharing

Folders, Items, and Item Revisions in a vault can be shared on a number of different levels, in effect defining both the level of visibility of the entity, and the level of security for access to it. This can range from being strictly private access by specified individuals or roles, through to levels for allowing anyone in the same organization to view or change that entity respectively.

Read about Controlling Access to Vault Content.

Vault Components



Altium Designer, with its unified design approach, has traditionally used a component model that extends across all aspects of the electronics design process. However, to seamlessly fit the process of electronics design into the encapsulating product development process as a whole, this model needs to evolve - extending to cover other aspects including other design processes (in particular MCAD and Industrial Design), as well as business processes (such as procurement and manufacturing) that intersect with the product development process.

This evolved object model is known as the *Unified Component Model*.

Under this modeling paradigm, the design component, as seen by the designer, is separated from the Manufacturer and/or Vendor parts. This information is not defined as part of the component. Instead, a separate vault Item - a *Part Choice List* Item - is used to map the design component to one or more Manufacturer Parts, listed in a *Part Catalog*, which in turn can be mapped to one or more Vendor parts, allowing the designer to state up-front, what real parts can be used for any given design component used in a design.

These components, along with their part choices, are stored in a target Altium Vault. A component is stored as a series of revisions of a uniquely-identifiable Component Item. Each revision is lifecycle-managed, providing collections of certified components, authorized to be re-instantiated into new design projects, manufactured into prototypes, or used for production runs. In short, a catalog of components implemented through vault-based 'libraries'.

Read about Working with Vault Components.
