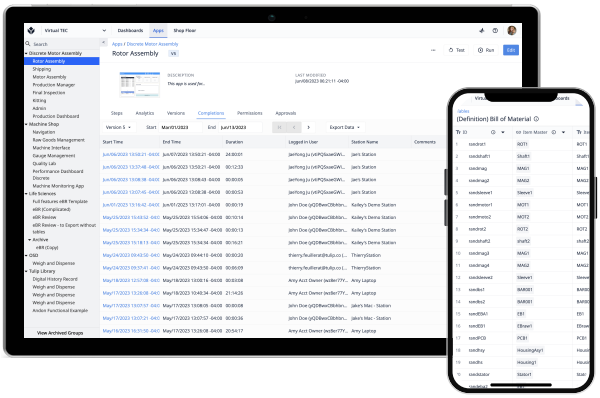
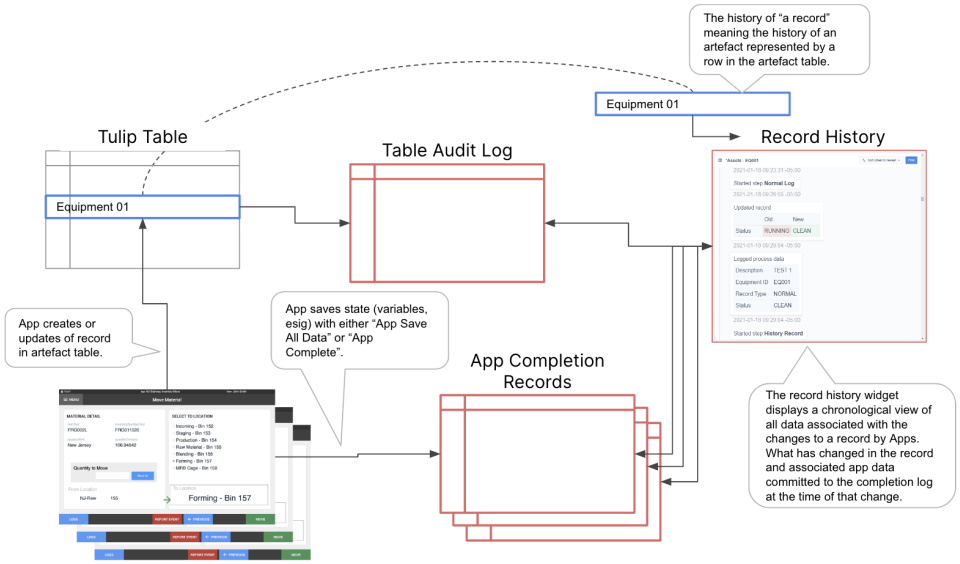
App Completions and Tables

Completions and Tulip **tables** work together to create a fully traceable process **history record**.





How do they work together?

History records are created by linking a Tulip **table record** with app **completion data**, which means at least one table record must be linked to an app and used in the **process** to capture data in **real time**.

The **Record History Widget**, a regulated industry feature, is compiled from multiple app completions. Apps should **complete** or **save** at the points in the process where data needs to be logged.

** Data in the Record History Widget is printable and compliant with 21 CFR Part 11*

What are the differences?

Completions	Tables
Records are **automatically** created at each app **save** point or upon **completion** (or cancellation) with the triggers "Save all App Data" or "Complete App"	Tables are **manually** created by an admin in the **Table Editor**; records are created using trigger logic, CSV import, automations, or manual entry
Immutable, intentionally cannot be changed or deleted; corrections are made by *adding* records	**Editable**, users with permission may change or delete records
Data is saved **locally** to the app and can be used in analyses but not trigger logic	Data is saved **globally** to the instance or workspace and can be embedded and used across many APIs and in apps, trigger logic, and automations

Tables as Artifacts

When to use tables?

Tulip apps mirror a process on the shop floor and log data as an operator executes a process from point A to point B. When an app saves or completes, it generates an immutable **Completion Record** containing a history of the app cycle. *E.g. operator name, station name, process start and end times, duration, variable values: sensor readings, quality control measurements, etc.*

If an **app** reflects a *process*, a **table** represents a *thing*.

Because apps log process data for you automatically, tables are most effective when used to store physical or operational **artifacts**—things that exist and are used in your manufacturing operations.

What are artifacts?

- **Physical Artifacts**: tangible objects or components that are used or produced during operations.
- **Operational Artifacts**: non-physical elements or components that enable or support operations.

Physical Artifact Examples

- Batches
- Equipment
- Locations
- Lots
- Machines
- Materials
- Etc.

Operational Artifact Examples

- Andons
- Defects
- Events
- Jobs
- Orders
- Tasks
- Etc.

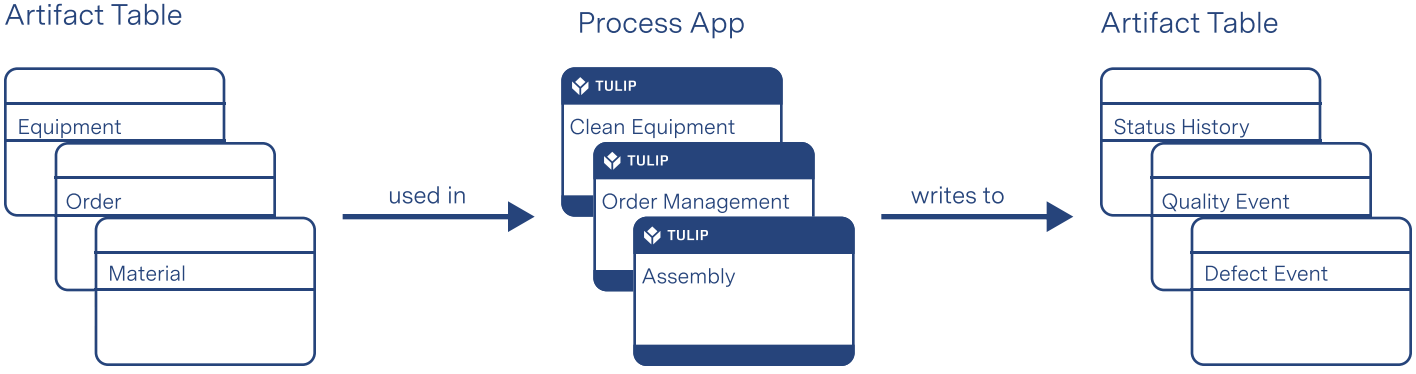


Table Fields

How are Tulip Tables organized?

When building a table in Tulip, you create **fields** which hold and organize data by **records**. Fields are the table **columns** and records are the table **rows**.

Table data can be added in a few ways:

- Stored with **trigger logic** from an app *(e.g. when an operator logs a defect, a new record in a Defects table is created)*
- **CSV import** *(e.g. loading a list of tools and current calibration dates)*
- **Automations**
- **Manual entry**

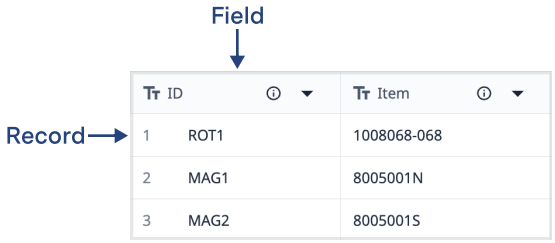


Table Data Types:

- Text
- Integer
- Number
- Datetime
- Interval
- Color
- Image URL
- File
- Object
- User
- Machine
- Station
- Boolean
- Linked Record

Required Table Fields:

Every table is required to have an **ID field**, containing a unique identifier for each record.

While this ID field can be a random string of text, tables are most effective when they represent real world physical or operational artifacts and the ID field contains the actual ID/SKU/QR Code, etc.

What fields should I create?

Table fields will depend on the process and needs of your frontline workers. However, there are generally two types of fields:

- **Status** - What is happening to the artifact in real time. *E.g. maintenance, tooling, cleaning, etc.*
- **Attributes** - Characterizations of the artifact. *E.g. quantity, color, potency, yield, etc.*

Summary of Best Practices:

- The **ID** of an artifact table should be the unique identifier of that artifact in the operational setting to maintain context as data is moved away from the original source.
- **Field** and **table names** should represent naming conventions and language used in the actual operational scenario.
- Keep “master data” at the source system and **minimize replication of data** *(e.g. a BOM in Tulip and the same BOM in ERP)*. Where possible, **data should be accessed at the source system**.

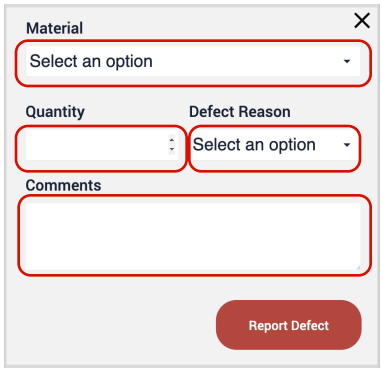
App Variables

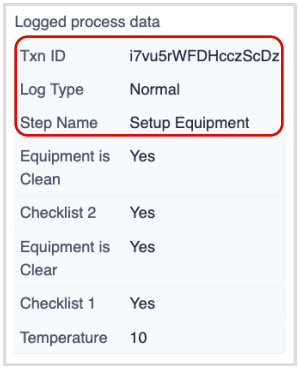
What kinds of data should I store in variables?

User or **machine** captured values such as photos, results recorded for critical process parameters, sensor readings, etc.

Units of measure that pair with captured values. It is best practice to create separate UoM variables in addition to measurement variables.

Contextual information to ensure history records are human readable. For example, step names or a unique ID captured at the start of each app cycle to enable easy search.





Variable Data Types:

- Text
- Integer
- Number
- Datetime
- Interval
- Color
- Image URL
- File
- Object
- User
- Machine
- Station
- Boolean

The **end values** of all variables are automatically saved to **completion records**. However, there are times when you may want to save variables to a table (in addition to completions) so the data can be used outside the app. To do this, the **variable data type** must match the **table field data type**.