# JIRA Object Relationships

### Intro:

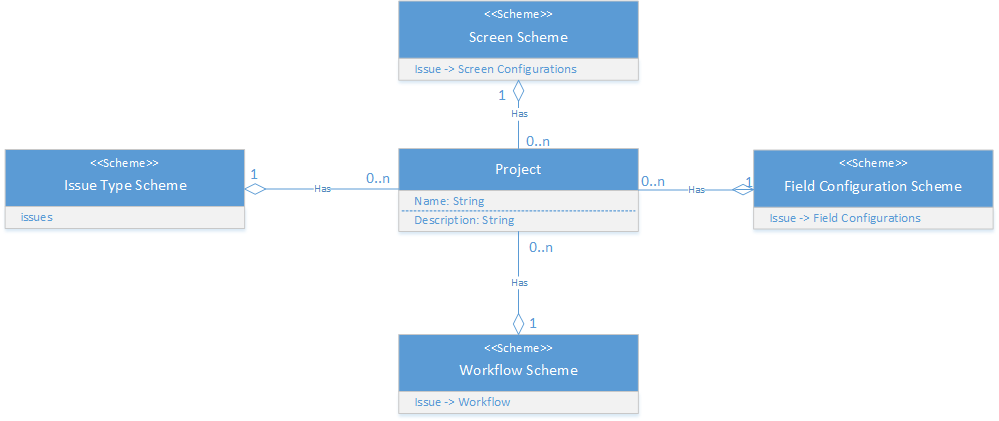
When dealing with any piece of software, it is important to understand the relationships between the abstractions (objects) in the software. This document was made to help better understand the relationships between the objects that exist within JIRA. Understanding these relationships will allow for a better understanding what a change in a JIRA instance will affect. There are 5 main types of objects in Jira. These 5 types of objects are: projects, issues, screens, fields, and workflows.

#### Project:

A project is a collection of issues in a JIRA system. The way these issues are organized and presented to the user is defined in the project schemes. Schemes can be thought of configuration files.

#### Project Schemes:

The highest-level abstraction in JIRA is a project. JIRA uses a \*\*scheme\*\* abstraction to store the relationships between project and objects. The below diagram shows the four main type of Schemes related to projects. There is a 5th scheme: Permissions but that scheme will be addressed later.



##### Breaking Down Scheme Properties:

\* \*\*Issue Type Scheme\*\* - Issue types available in the project

\* \*\*Screen Scheme\*\* - Issue -> Screen Configuration Relationships

\* \*\*Field Configuration Scheme\*\* - Issue Types -> Field Configurations Relationships

\* \*\*Workflow Scheme\*\* - Issue Types -> Workflow Relationships

#### Issue (Internally referred to as tickets. Issue, in our vernacular is a specific RAID ticket type.):

Issues are related to projects through the issue type scheme. An issue is by far the most important concept in JIRA. It is the thing that the everyday user with interact with. Issues include: epics, stories, tasks, and subtasks but can be can be custom created to represent virtually any concern or object type such as a Server, Service, Application, Risk, Decision, etc. Most of the time a JIRA user will be either opening, closing, logging work against, or editing an issue. When an issue is created, it belongs to a workflow. A workflow is a set of lifecycle stages and transitions. Issues have a field configuration which defines the data that is attached to the issue. These field can already exist in the system or the fields can be created as \*custom fields\*. The way the fields of an issue are displayed can be organized in the screen configuration.

##### Issue Actions:

\* \*\*Opening\*\*: Issues are opened when a task needs to be completed and can be assigned to the person that needs to complete it. When an issue is opened, it becomes an active object in the JIRA system.

\* \*\*Closing\*\*: It is extremely important to note that closing an issue is not the same as deleting one. Closing the issue puts the issue into a "resolved" state but the issue still exists in the JIRA system and can be queried against. Most of the time, Issues are not deleted and it is best practice to close them instead.

\* \*\*Logging Work\*\*: As work is done on a specific issue, it can be logged and linked to the issue. This allows for time tracking.

\* \*\*Editing\*\*: When issues are created, they contain fields that can be edited. These fields can be any piece of information that needs to be tracked.

\* \*\*Watchers\*\*: Users can be assigned as watchers to enable notification subscriptions for changes to the issue

\* \*\*Comments\*\*: Each issue allows for comments providing an issues centric management of discussions.

\* \*\*History\*\*: Each issue maintains a viewable history of all changes and actions associated to it providing a clear and concise audit trail.

#### Screen:

A screen is an actual UI view for the user to view fields. Screens organize fields in order and in tabs.

##### Screen Configuration:

The screen configuration is the set of UI views for an issue. These views can be configured to show the issue fields in the most appealing way. There are three operations on an issue that can have screens mapped to them: Create, Edit, and View.

#### Fields:

Fields represent actual data points associated with an issue that are stored in the database. Fields can be scoped to individual projects or issue types. It is very important to properly scope fields. If a field is scoped to an issue but does not show up on any screens in the UI views, it still exists in the database but will not be accessible. This can lead to random data filling up the database. Fields can either be made mandatory or optional. Mandatory fields will need to be filled before an issue can be created.

The field configuration defines the available screens for the field and how it will be displayed. This is also where the fields will be defined as required or optional. Along with where in the screen configuration, this is also where the description of the field can be set.

#### Workflow:

A workflow consists of Statuses and Transitions. Each status is the state the issue exists in. Each transition can be considered an action that changes an issue from one status to the other. Below is an example transition. \*To Do\* and \*Complete\* would be considered the statuses and \*Work Completed\* would be the transition for the object. These statuses would then be mapped to JIRA \*Status Categories\*. It is important to note that the object will still exist in the JIRA system even in the completed state. Until an issue is explicitly deleted, it will exist in the JIRA system.

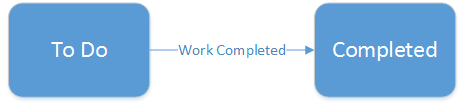
##### Workflow States:

\* \*\*Status Categories\*\*: Immutable: JQL: {statusCategory: [“To Do”, “In Progress”, “Done”]}.

\* \*\*Resolutions\*\*: Customizable: {Global: [Name, Description, Order, Default]}, JQL: {resolution: “<resolution name>”]: Provide additional data as to why the ticket was closed.

\* \*\*Status\*\*: Customizable: {Global: [Name, Category], Workflow: [(property)k=v]}, JQL: {status: “<status name>”}

\* \*\*Transitions\*\*: Customizable: {Workflow: [Name, Properties, Triggers, Conditions, Validators, Post Functions]}



#### User:

The user is the object that a person uses to interact with the system. Users belong to groups. These groups are then assigned roles in a project. A permission scheme is then used to assign project roles to responsibilities in a JIRA project.

