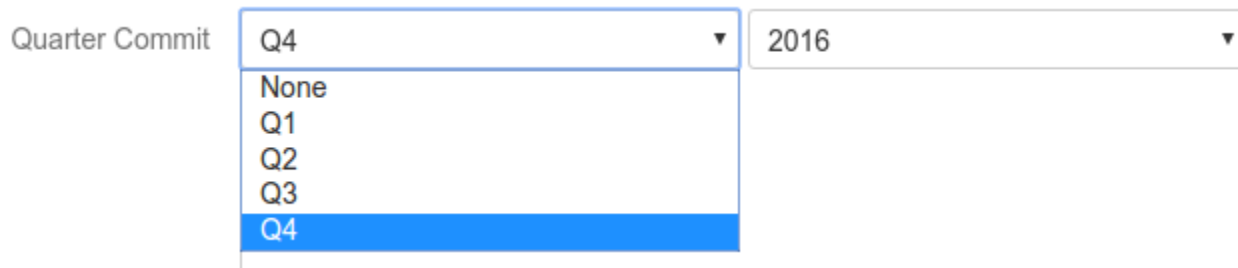


# Visualizing a Quarter Commit field

JIRA has a built-in **Due Date** field. But what happens when you don't know the exact day – you know, at best, the year *quarter* (Q1–Q4) in which the issue is due.

A client of ours solved this by defining a custom field, showing the quarter (and optionally, year) to which the issue is committed to be delivered in:



Quarter Commit

Q4 ▼

None

Q1

Q2

Q3

Q4

2016 ▼

This works, but the result is a little hard for humans to parse, as there is **very little visual distinction** between different values:

Key	Status	Quarter Commit
PRJ-980	PROPOSE	Q3 - 2016
PRJ-938	IMPLEMENT	Q2 - 2016
PRJ-937	ON HOLD	Q3 - 2016
PRJ-936	ON HOLD	Q4 - 2016
PRJ-935	ON HOLD	Q3 - 2016
PRJ-934	ON HOLD	Q4 - 2016
PRJ-933	DEFINE	Q2 - 2016
PRJ-932	IMPLEMENT	Q3 - 2016
PRJ-931	IMPLEMENT	Q2 - 2016
PRJ-930	IMPLEMENT	Q2 - 2016
PRJ-929	IMPLEMENT	Q2 - 2016
PRJ-928	IMPLEMENT	Q2 - 2016
PRJ-923	DEFINE	Q1 - 2016

## Solution – an easy-to-visualize 'view'

We solved this by creating a read-only graphical "timeline" view of the field, here seen alongside the text view:

Key	Status	Quarter Commit	Quarter Commit				
PRJ-980	PROPOSE	Q3 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-938	IMPLEMENT	Q2 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-937	ON HOLD	Q3 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-936	ON HOLD	Q4 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-935	ON HOLD	Q3 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-934	ON HOLD	Q4 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-933	DEFINE	Q2 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-932	IMPLEMENT	Q3 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-931	IMPLEMENT	Q2 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-930	IMPLEMENT	Q2 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-929	IMPLEMENT	Q2 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-928	IMPLEMENT	Q2 - 2016	... Q2	Q3	Q4	Q1	...
PRJ-923	DEFINE	Q1 - 2016	... Q2	Q3	Q4	Q1	...

I think you'll agree, it's much easier to see what's going on. The field begins with the current quarter (it's May 2016 at time of writing, so Q2), and displays 4 quarters into the future. The last issue, with the grey ellipsis, shows an issue committed to *before* the current quarter (i.e. overdue). A mouseover explains what's going on:

PRJ-923	DEFINE	Q1 - 2016	...	Q2	Q3	Q4	Q1	...
---------	--------	-----------	-----	----	----	----	----	-----

Committed to Q1 2016 (in the past)

Likewise for issues committed to more than one year in the future.

## Implementation

We utilized the indispensable [ScriptRunner for JIRA](#) plugin to create a 'script field', i.e. a field whose value is calculated programmatically. The implementation is as follows:

```
/**
 * An alternative timeline view of the 'Quarter Commit' field.
 * jeff@redradishtech.com, 4/May/16.
 */
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.customfields.option.Option
import com.atlassian.jira.issue.fields.CustomField

// ID of the cascading select custom field storing the Quarter and Year.
def QUARTER_COMMIT_CUSTOMFIELDID = 14100;

/** Get the year in which a customfield was last set in this issue, or null if never set. */
def Integer getFieldSetYear(CustomField customField) {
    def chm = ComponentAccessor.getChangeHistoryManager();
    def historyItems = chm.getChangeItemsForField(issue, customField.getName());
    if (historyItems.empty) return null;
    fieldSetDate = historyItems.reverse().first().getCreated();
    def fieldSetCal = Calendar.getInstance();
    fieldSetCal.setTimeInMillis(fieldSetDate.getTime());
    year = fieldSetCal.get(Calendar.YEAR)
    return year;
}

cf = ComponentAccessor.customFieldManager.getCustomFieldObject(QUARTER_COMMIT_CUSTOMFIELDID);
Map<String, Option> cfVal = issue.getCustomFieldValue(cf);

if (!cfVal) return;
currentYear = Calendar.getInstance().get(Calendar.YEAR) as Integer;
currentQuarter = Calendar.getInstance().get(Calendar.MONTH).intdiv(3) as Integer; // 0 to 3

issueQuarter = cfVal[null]?.getValue()?.replaceFirst("[qQ]", "").toInteger() - 1; // 0 to 3. E.g. "Q1" becomes 0
issueYear = cfVal["1"]?.getValue()?.toInteger(); // e.g. 2017

if (!issueYear) {
    // It's possible for the field's year to be unset. In that case we cunningly infer the year from the
    // date at which 'Quarter Commit' was last modified.
    issueYear = getFieldSetYear(cf);
    if (!issueYear) {
        // If for some bizarre reason we can't tell when Quarter Commit was set, default to current year.
        issueYear = currentYear
    };
};

/**
 * Display the 'before current quarter' block, grey if our issue was scheduled any time
 * before the current quarter, white otherwise.
 */
def beforeNowBlock() {
    if ((issueYear < currentYear) ||
        (issueYear == currentYear && issueQuarter < currentQuarter))
    {
        "<th style='padding-left: 0px; padding-right: 0px; background: lightgrey' title='Committed to Q${issueQuarter+1} ${issueYear?issueYear:''} (in the past)'>&hellip;</th>"
    } else {
        "<td style='padding-left: 0px; padding-right: 0px;'>&hellip;</td>"
    }
}

QUARTERS = ["Q1", "Q2", "Q3", "Q4"]

/**
 * Display a quarter's block. If the issue's year + quarter equals ours, display green, otherwise white.
 *
 * @param quarterIndex Display the n'th quarter (0 to 3).
 *
 */
```

```

def quarterBlock(quarterIndex) {
  // E.g. if it's currently Q2:
  //   - quarterIndex will be, successively, 1, 2, 3, 4
  //   - quarterIndex % 4 will be, successively, 1, 2, 3, 0
  //   - 'q' will be, successively, Q2, Q3, Q4, Q1.

  // E.g. if it's currently Q4:
  //   - quarterIndex will be, successively, 3, 4, 5, 6
  //   - quarterIndex % 4 will be, successively, 3, 0, 1, 2
  //   - 'q' will be, successively, Q4, Q1, Q2, Q3.
  def q = QUARTERS[quarterIndex % 4]

  // Imagine it's now Q4 2016. That means our field displays four blocks 'Q4 Q1 Q2 Q3', where the last 3
  // blocks
  // are for 2017, not 2016. Say we're called to render that last Q3 block (quarterIndex=6). We must
  highlight it
  // - IF year is 2017, i.e. 2016+1 = currentYear+(6/4)
  // - AND quarter is Q3, i.e. QUARTERS[6%4]
  output = (issueYear == currentYear + ((quarterIndex).intdiv(4)) && issueQuarter == quarterIndex%4) ?
    "<th style='background-color:lightgreen' title='Committed to Q${issueQuarter+1} ${issueYear}'>${q}<
  /th>" :
    "<td style='color:grey'>${q}</td>";
  output
}

/* Display the 'after our 4-quarter window' block. */
def afterQuartersBlock() {
  // E.g. for Q1 2017 when we're in Q2, don't show (currentQuarter=1, issueQuarter=0)
  if ((issueYear > currentYear) &&
      issueQuarter >= currentQuarter) {
    "<th style='padding-left: 0px; padding-right: 0px; background-color: lightgreen' title='Committed to
    Q${issueQuarter+1} ${issueYear}'>&hellip;</th>"
  } else {
    "<td style='padding-left: 0px; padding-right: 0px;'>&hellip;</td>"
  }
}

return ""
<table class="aui">
  <tbody>
    <tr>
      ${beforeNowBlock()}
      ${quarterBlock(currentQuarter)}
      ${quarterBlock(currentQuarter+1)}
      ${quarterBlock(currentQuarter+2)}
      ${quarterBlock(currentQuarter+3)}
      ${afterQuartersBlock()}
    </tr>
  </tbody>
</table>
"" as String;

```

The logic is tricky because we start with the current quarter, not always Q1, but it seems correct.

There is some ambiguity when the year is unset. For instance, if we see "Q1", does it mean Q1 this year (which has already passed), or Q2 next year? We decide by looking into the change history to see *the year in which the field itself was last touched*. If someone set "Q1" back in 2015, they meant "Q1 2015".

Here is a test suite of all possible values and their visualizations, showing commits in the past, current year, and future years.

Summary	Quarter Commit	Quarter Commit					
☰ Global Structure							
🔗 Q2, but set 1 year ago	Q2	...	Q2	Q3	Q4	Q1	...
🔗 Q1	Q1	...	Q2	Q3	Q4	Q1	...
🔗 Q2	Q2	...	Q2	Q3	Q4	Q1	...
🔗 Q3	Q3	...	Q2	Q3	Q4	Q1	...
🔗 Q4	Q4	...	Q2	Q3	Q4	Q1	...
🔗 Q1 2016	Q1 - 2016	...	Q2	Q3	Q4	Q1	...
🔗 Q2 2016	Q2 - 2016	...	Q2	Q3	Q4	Q1	...
🔗 Q3 2016	Q3 - 2016	...	Q2	Q3	Q4	Q1	...
🔗 Q4 2016	Q4 - 2016	...	Q2	Q3	Q4	Q1	...
🔗 Q1 2017	Q1 - 2017	...	Q2	Q3	Q4	Q1	...
🔗 Q2 2017	Q2 - 2017	...	Q2	Q3	Q4	Q1	...
🔗 Q3 2017	Q3 - 2017	...	Q2	Q3	Q4	Q1	...

## What next?

It would be great if one could *set* the Quarter Commit by clicking on the relevant box, but that's beyond ScriptRunner's capabilities. Ideally this field should be rewritten as a full custom field, and published, like the [traffic light custom field](#). Sponsors welcome!

As always, please [contact us](#) if you have any comments or queries.