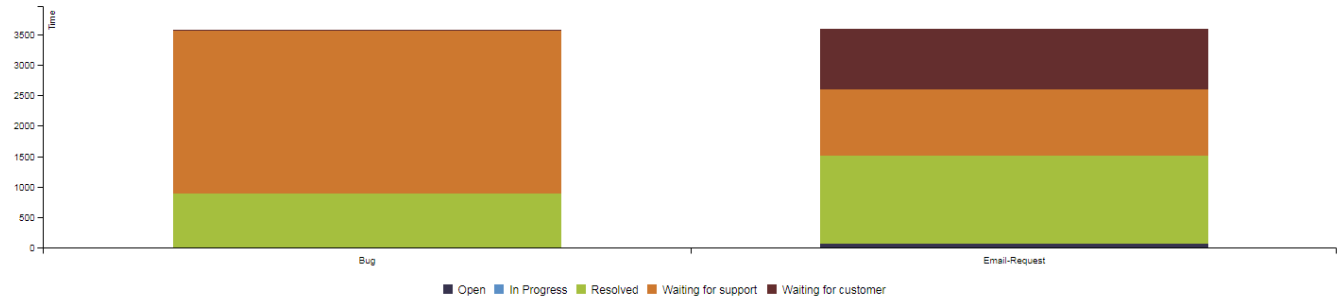
Time in status (By issue attribute)

The time in status shows the accumulated time of issues chosen via JQL in a specific status. Additional grouping is possible on the Y axis, by using the GroupBy Parameter.

Grouping by None will display the overall time in Status for the issues chosen via JQL.

Chart preview



Parameters

Parameter		Type	Default value
UI	**Code**		
JQL	`JQL`	JQL Autocomplete	
Group By	`Group`	Group By Picker	Issue Type

Layout Script

Used layout: **Easy Group By**

```
function formatTooltipAsHours(value, ratio, id, index)
{
    return value.toFixed(2) + ' h';
}

function formatTooltipAsHoursWithDays(value, ratio, id, index)
{
    var hours = parseInt(value);
    var days = parseInt(hours / 24);
    value = value - days * 24;
    if (days > 0)
    {
        return days + 'd, ' + value.toFixed(2) + ' h';
    }
    return value.toFixed(2) + ' h';
}

var c3arg = {
    onrendered: updateFrameHeight,
    data: chartData,
    axis: {
        x: {
            type: 'category', // this is needed to load string x value
            label: {
                text: chartData.custom.xLabel,
                position: 'outer-left'
            }
        },
        y: {
            label: chartData.ytype
        }
    }
};

if (chartData.custom && chartData.custom.tooltip)
{
    var tooltipFunction = eval(chartData.custom.tooltip);
    c3arg.tooltip = {
        format: {
            value: tooltipFunction
        }
    };
}

c3.generate(c3arg);
```

Data Script

```
import java.lang.reflect.Field;
import java.math.BigDecimal;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Date;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

import org.apache.log4j.Logger;
import org.apache.lucene.document.Document;

import com.atlassian.jira.component.ComponentAccessor;
import com.atlassian.jira.config.ConstantsManager;
import com.atlassian.jira.issue.DocumentIssueImpl;
import com.atlassian.jira.issue.Issue;
```

```
import com.atlassian.jira.issue.changehistory.ChangeHistoryManager;
import com.atlassian.jira.issue.history.ChangeItemBean;
import com.atlassian.jira.issue.status.Status;
import com.atlassian.jira.jql.parser.JqlParseException;
import com.atlassian.jira.jql.parser.JqlQueryParser;
import com.atlassian.query.Query;
import com.decadis.jira.xchart.api.ChartParam;
import com.decadis.jira.xchart.api.model.ChartType;

def logger = Logger.getLogger("Time In Status Chart")
def i18nHelper = ComponentAccessor.getJiraAuthenticationContext().getI18nHelper()

public static BigDecimal TimeDiffInHours(Date a, Date b)
{
  return BigDecimal.valueOf(((double) (Math.abs(a.getTime() - b.getTime()) / 1000l)) / 3600.0d);
}

String jql = JQL;

JqlQueryParser jqlQueryParser = ComponentAccessor.getComponent(JqlQueryParser.class);
Query query = null;
try
{
  query = jqlQueryParser.parseQuery(jql);
} catch (JqlParseException e)
{
  logger.warn("Bad JQL:" + jql, e);
  throw new IllegalArgumentException("Bad JQL: " + jql);
}

// for evaluation of issue history
ChangeHistoryManager changeHistoryManager = ComponentAccessor.getChangeHistoryManager();

ConstantsManager constantsManager = ComponentAccessor.getConstantsManager();

Date now = new Date();

def data = chartBuilder.newDataCollector()

def grouper = chartBuilder.getGrouper(Group);

Field documentField;
try
{
  documentField = DocumentIssueImpl.class.getDeclaredField("document");
  documentField.setAccessible(true);

  // iterate over the issues
  for ( Issue issue : chartBuilder.getFilterUtils().performSearch(query, user) )
  {

    Date since = issue.getCreated();
    Date until = now;
    String currentStatus = null;
    def groups = grouper.getGroups((Document) documentField.get(issue));

    // iterate over the issue change history of the status field
    for ( ChangeItemBean item : changeHistoryManager.getChangeItemsForField(issue, "status") )
    {
      until = item.getCreated();

      // first change item -> initial status of issue is unknown
      if ( currentStatus == null )
      {
        currentStatus = constantsManager.getStatus(item.getFrom()).getNameTranslation(i18nHelper);
      }

      for ( String group : groups )
      {
        data.addValue(TimeDiffInHours(until, since), currentStatus, grouper.getResolvedValue(group, issue));
      }
```

```
      since = item.getCreated();
      currentStatus = constantsManager.getStatus(item.getTo()).getNameTranslation(i18nHelper);
    }

    // add time of current status
    until = now;
    currentStatus = issue.getStatus().getNameTranslation(i18nHelper);
    for ( String group : groups )
    {
      data.addValue(TimeDiffInHours(until, since), currentStatus, grouper.getResolvedValue(group, issue));
    }
  }
} catch (Exception e)
{
  logger.error("Could not compute time in status", e);
  throw new RuntimeException(e);
}

data.fillMissingValues();

def chartData = chartBuilder.newChartData(i18nHelper.getText("xchart.charts.time"));
chartData.setType(ChartType.BAR.getKey());
chartData.addCustomData("tooltip", "formatTooltipAsHoursWithDays");
chartData.setYType(i18nHelper.getText("xchart.charts.time"));

List<String> statusOrder = new LinkedList<String>(data.keySet());

statusOrder.sort(new Comparator<String>()
{
    @Override
    public int compare(String o1, String o2)
    {
        Status ss1 = constantsManager.getStatusByTranslatedName(o1);
        if ( ss1 != null )
        {
           Status ss2 = constantsManager.getStatusByTranslatedName(o2);
           if ( ss2 != null )
           {
             Long s1 = ss1.getSequence();
             Long s2 = ss2.getSequence();
             return s1.compareTo(s2);
           }
        }
        return o1.compareToIgnoreCase(o2);
    }
});

chartBuilder.getChartUtil().transformResultOrdered(data, chartData, true, statusOrder.toArray(new String
[0]));

return chartData;
```

If you still have questions, feel free to refer to our support team.