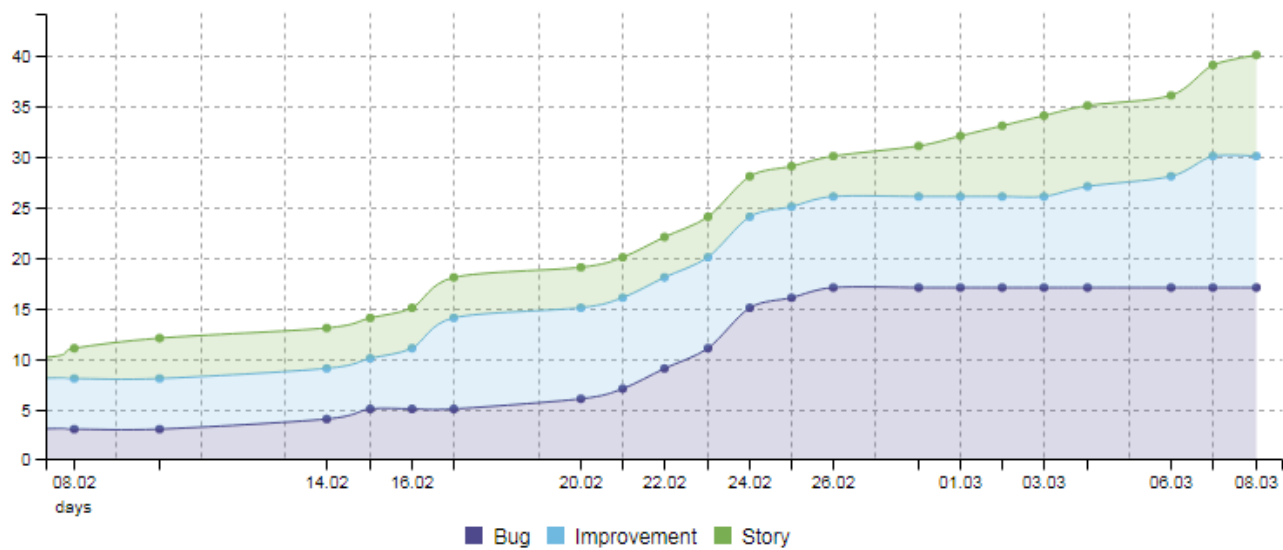


Time frame chart

The **time frame chart** is a highly configurable issue chart, which shows the number of issues over time, grouped by a specific date field on the X-axis, and by a chosen field on the Y-axis.

Chart preview



Parameters

Parameter		Type	Default value
in UI	in Code		
JQL	JQL	JQL Autocomplete	
Date Field	DateFieldParam	Date Field Picker (single field)	Date of creation
Time Period	TimePeriod	Time Period Picker	Month
From	From	Interval Picker	-1m (relative time, it goes 1 month into the past from today)
To	To	Interval Picker	+1m (same as above, but 1 month into the future, if data exists)
Group By	GroupBy	Group By Picker	Issue Type
Chart Type	ChartType	Chart Type Picker	Bar
Value Picker	ValuePicker	Value Field Picker	IssueCount
- Accumulate Groups - Accumulate Values - Zoom	AccumulateGroups	Boolean	True
	AccumulateValues	Boolean	False
	EnableZoomParam	Boolean	False

Layout Script

```
function formatQuarter(d)
{
    if (d instanceof Date)
    {
        var q = d.getMonth();
        q = parseInt(q / 3) + 1;
        return 'Q' + q;
    }
    return '';
}

function formatHalfyear(d)
{
    if (d instanceof Date)
    {
        var q = d.getMonth();
        q = parseInt(q / 6) + 1;
        return 'H' + q;
    }
    return '';
}

c3.generate({
    onrendered: updateFrameHeight,
    data: chartData,
    grid: {
        y: {
            show: true
        },
        x: {
            show: true
        }
    },
    axis: {
        x: {
            type: 'timeseries',
            label: {
                text: chartData.custom.xLabel,
                position: 'outer-left'
            },
            tick: {
                format: eval(chartData.custom.xTickFormat),
                culling: {
                    max: 25
                },
                fit: true,
                multiline: false
            }
        }
    }
})
```

Data Script

```
import java.lang.reflect.Field;

import java.math.BigDecimal;
import java.util.Date;

import org.apache.lucene.document.Document;

import com.atlassian.jira.component.ComponentAccessor;
import com.atlassian.jira.issue.DocumentIssueImpl;
```

```

import com.atlassian.jira.issue.Issue;
import com.atlassian.jira.issue.IssueFieldConstants;
import com.atlassian.jira.issue.fields.CustomField;
import com.atlassian.jira.issue.fields.FieldManager;
import com.atlassian.jira.jql.builder.JqlClauseBuilder;
import com.atlassian.jira.jql.builder.JqlQueryBuilder;
import com.atlassian.jira.jql.parser.JqlQueryParser;
import com.atlassian.query.Query;
import com.atlassian.query.operator.Operator;

import com.decadis.jira.xchart.api.ChartParam;
import com.decadis.jira.xchart.api.model.Period;
import com.decadis.jira.xchart.api.util.DateUtils;

def fieldManager = ComponentAccessor.getFieldManager();
def i18n = ComponentAccessor.getJiraAuthenticationContext().getI18nHelper();

//helper method to create the JQL Query
def createQuery(String jql, com.atlassian.jira.issue.fields.Field field, boolean isCustomField, Period
period)
{
    JqlQueryParser jqlQueryParser = ComponentAccessor.getComponent(JqlQueryParser.class);
    Query query = null;
    try {
        JqlClauseBuilder jqlClauseBuilder = JqlQueryBuilder.newClauseBuilder(jqlQueryParser.parseQuery(jql));

        Date start = chartBuilder.getDateFromParam(From);
        Date end = chartBuilder.getDateFromParam(To);

        String searchKey = field.getId();
        if ( isCustomField ) {
            searchKey = "cf[" + ((CustomField) field).getIdAsLong() + "]";
        }

        jqlClauseBuilder.defaultAnd();
        if ( start != null ) {
            jqlClauseBuilder.addDateCondition(searchKey, Operator.GREATER_THAN_EQUALS, start);
        }

        if ( end != null ) {
            jqlClauseBuilder.addDateCondition(searchKey, Operator.LESS_THAN, end);
        }

        query = jqlClauseBuilder.buildQuery();
    } catch (Exception e) {
        throw new IllegalArgumentException("Bad JQL: " + jql);
    }
    return query;
}

//here starts the main method
String dateField = DateFieldParam;
if ( dateField.isEmpty() )
{
    dateField = IssueFieldConstants.CREATED;
}

com.atlassian.jira.issue.fields.Field dateFieldObject = fieldManager.getField(dateField);
boolean isCustomField = fieldManager.isCustomField(dateFieldObject);

Period selectedPeriod = Period.fromString(TimePeriod);

Query query = createQuery(JQL, dateFieldObject, isCustomField, selectedPeriod);
def countGroup = chartBuilder.newDataCollector();
def valueExtractor = chartBuilder.getValueExtractor(ValuePicker);

if ( valueExtractor == null )
{
    throw new IllegalArgumentException("Valuepicker for " + ValuePicker + " is not supported.");
}

```

```

def groupValueExtractor = chartBuilder.getGrouper(GroupBy);
Field documentField;
try
{
    documentField = DocumentIssueImpl.class.getDeclaredField("document");
    documentField.setAccessible(true);

    for ( Issue issue : chartBuilder.getFilterUtils().performSearch(query, user) )
    {
        Document document = (Document) documentField.get(issue);
        String[] groups = groupValueExtractor.getGroups((Document) documentField.get(issue));

        for ( String group : groups )
        {
            Date dateFromField = new Date();
            if ( isCustomField ) {
                dateFromField = (Date) issue.getCustomFieldValue((CustomField) dateFieldObject);
            }
            else
            {
                switch ( dateFieldObject.getId() ) {
                    case IssueFieldConstants.CREATED:
                        dateFromField = issue.getCreated();
                        break;
                    case IssueFieldConstants.UPDATED:
                        dateFromField = issue.getUpdated();
                        break;
                    case IssueFieldConstants.RESOLUTION_DATE:
                        dateFromField = issue.getResolutionDate();
                        break;
                    case IssueFieldConstants.DUE_DATE:
                        dateFromField = issue.getDueDate();
                        break;
                    default:
                        dateFromField = new Date();
                        break;
                }
            }
            countGroup.addValue(valueExtractor.get(issue, document), group, dateUtils.getPeriodGroup(
                dateFromField, selectedPeriod));
        }
    }
} catch (Exception e) {
    System.out.println("Could not extract groups" + e);
}

if ( AccumulateValues )
{
    countGroup.accumulateGroups();
}
else
{
    countGroup.fillMissingValues();
}

def chartData = chartBuilder.newChartData(i18n.getText("common.concepts.issues"));

for ( String grpKey : countGroup.keySet() )
{
    chartData.addGroupName(grpKey, groupValueExtractor.getResolvedValue(grpKey, null));
}

chartData.setDateFormat(DateUtils.SimpleDateFormatD3);
chartData.setPeriod(selectedPeriod);
chartData.setType(ChartType);
chartData.addCustomData("enableZoom", EnableZoomParam);

chartBuilder.getChartUtil().transformResult(countGroup, chartData, AccumulateGroups);
return chartData;

```

If you still have questions, feel free to refer to our [support](#) team.