



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Knowledge Center Developer's Guide

Implicit User Feedback

12/14/2025

Implicit User Feedback

Overview

To improve search quality, gather implicit user feedback via Proactive Engagement integration.

For example, sending additional implicit user feedback automatically in cases such as:

- Customer executed the search, saw search results and left search result page in less then X seconds - > Negative feedback published for all documents in the result set.
- Customer executed the search, saw result, opened the document and left the page in less then X seconds -> Negative feedback published for particular document.
- Customer executed the search, saw result, opened the document and remained on the page for X minutes -> Positive feedback to be published for the document.
- Customer executed the search, saw result, opened the document and opened attachment to the document-> Positive feedback to be published for the document.

Customizing the Script and reconfiguring the routing strategy

1. Add additional JavaScript code to the Web Monitoring Agent on the front-end page to monitor described events.

For example:

1. Create a small JavaScript function to wrap sending feedbacks:

```
function feedbackSender
(decision, behavior, timeoutSeconds){
    this.decision = decision;//positive,negative
    this.timeoutSeconds = timeoutSeconds;
    this.behavior = behavior;//less,more,equal
    this.openTime = null;
    this.sendPositiveFeedback = function(){
        console.log("Positive feedback");
    };
    this.sendNegativeFeedback = function(){
        console.log("Negative feedback");
    };
    this.sendFeedback = function () {
        switch (decision) {
            case "positive":
                this.sendPositiveFeedback();
                break;
            case "negative":
                this.sendNegativeFeedback();
                break;
            default:
                console.log("Sorry, we are out of "
+ this.decision + ".");
        }
    };
}
```

```

    }
};
}

```

2. Add methods to monitor timeout:

```

this.onOpenPage = function () {
    this.openTime = new Date();
};
this.onLeftPage = function () {
    switch (behavior) {
        case "less":
            if ((this.openTime)&&((new Date())
- this.openTime) < this.timeoutSeconds * 1000)) {
                this.sendFeedback();
            }
            break;
        case "more":
            if ((this.openTime)&&((new Date())
- this.openTime) > this.timeoutSeconds * 1000)) {
                this.sendFeedback();
            }
            break;
        case "equals":
            if ((this.openTime)&&((new Date())
- this.openTime) == this.timeoutSeconds * 1000)) {
                this.sendFeedback();
            }
            break;
        default:
            console.log("Sorry, we are out of "
+ this.behavior + ".");
    }
}

```

3. Implement provided business cases using this function. For example:

- Case 1: Customer executed the search, saw search results and left search result page in less than X seconds -> Negative feedback published for all documents in the result set.

```

firstCaseFeedback = new feedbackSender
('negative','less',5)
firstCaseFeedback.sendNegativeFeedback=function()
{
    _gt.push(['event', {eventName: 'Feedback',
gks_Reason: 'negative', gks_docIds: strArr,
gks_sessionId: gks_sessionId,
gks_query: gks_query}]);
}
$(window).on('hashchange', function(e){
    if (window.location.hash.indexOf('/search/')
>= 0) {
        firstCaseFeedback.onOpenPage();
        arr=[];
        var $resultDiv = $('._gk-_wd-_rs');
        $resultDiv.ready(function () {
            $resultDiv.each(
            function(index, a) {
                var basicId = $(a).attr('id');
                arr.push(basicId.substring(18, basicId.length))
            }
        );
    })
}

```

```
strArr = JSON.stringify(arr);
gks_query = $('#searchContent').val();
gks_sessionId = window.localStorage.getItem('sessionId');
});
$(window).on('hashchange', function(e){
    var oldURL = e.originalEvent.oldURL;
    if (oldURL.indexOf('/search/') >= 0) {
        firstCaseFeedback.onLeftPage();
    }
});
```

- Case 2: Customer executed the search, saw result, opened the document and left the page in less then X seconds -> Negative feedback published for particular document.

```
secondCaseFeedback = new feedbackSender('negative','less',10)
secondCaseFeedback.sendNegativeFeedback=function()
{
    _gt.push(['event', {eventName: 'Feedback',
gks_Reason: 'negative', gks_docIds:
gks_docId, gks_sessionId: gks_sessionId,
gks_query: gks_query}]);
}

$(window).on('hashchange', function(e){
    if (window.location.hash.indexOf('/document/')
>= 0) {
        gks_query = $('#searchContent').val();
        gks_sessionId = window.localStorage.getItem('sessionId');
        gks_docId = window.location.hash.substr(window.location.hash.length - 36);
        secondCaseFeedback.onOpenPage();
    }
});

$(window).on('hashchange', function(e){
    var oldURL = e.originalEvent.oldURL;
    if (oldURL.indexOf('/document/') >= 0) {
        secondCaseFeedback.onLeftPage();
    }
});
```

- Case 3: Customer executed the search, saw result, opened the document and remained on the page for X minutes -> Positive feedback to be published for the document.

```
thirdCaseFeedback = new feedbackSender('positive','more',20)
thirdCaseFeedback.sendPositiveFeedback=function()
{
    _gt.push(['event', {eventName: 'Feedback',
gks_Reason: 'positive', gks_docIds:
gks_docId, gks_sessionId: gks_sessionId,
gks_query: gks_query}]);
}

thirdCaseFeedback.onOpenPage=function(){
    thirdCaseFeedback.openTime = new Date();
    setTimeout(function()
{thirdCaseFeedback.onLeftPage();},
```

```
(thirdCaseFeedback.timeoutSeconds+1)*1000);
}

$(window).on('hashchange', function(e){
    if (window.location.hash.indexOf('/document/')
    >= 0) {
        gks_query = $('#searchContent').val();
        gks_sessionId = window.localStorage.getItem
        ('sessionId');
        gks_docId = window.location.hash.substr
        (window.location.hash.length - 36);
        thirdCaseFeedback.onOpenPage();
    }
});
```

2. Create a business rule to invoke interaction on this event:

```
rule "Rule-101 Feedback"

salience 100001

    agenda-group "level0"

    dialect "mvel"

    when

        $event1: Event(eval($event1.getName().equals
        ('Feedback'))))

    then

        sendEvent($event1, ed, drools);

    end
```

3. Modify the strategy to route the interaction invoked by these events in a separate branch.
4. Use modules to send REST requests in these branches to publish feedback to the Knowledge server:

Positive feedback:

```
POST
URL: http://<host>:<port>/gks-server/v1/feedback/
knowledgefaq/documents/<docId>/
vote?relevant=true&sessionId=<sessionId>
BODY: {"query":"<query>"}
```

Negative feedback:

```
POST
URL: http://<host>:<port>/gks-server/v1/feedback/
knowledgefaq/documents/<docId>/
vote?relevant=false&sessionId=<sessionId>
BODY: {"query":"<query>"}
```