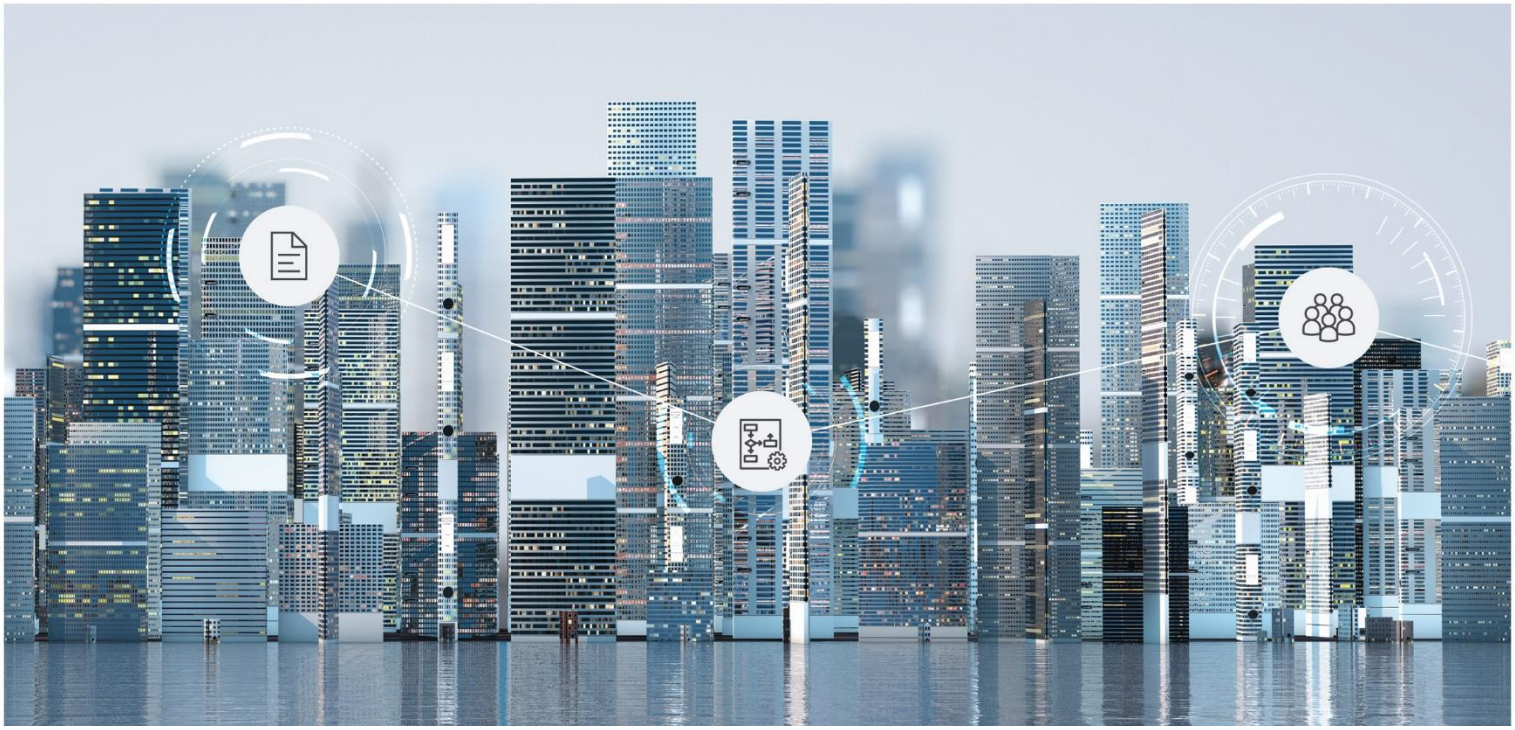
Fabasoft



White Paper

Fabasoft Folio Notifications

Fabasoft Folio 2024 Update Rollup 1

Contents

1 Introduction ____ 4
2 Software Requirements ____ 4
3 Getting Started ____ 4
4 HistoryEventType ____ 4
5 LogHistoryEvent ____ 7
6 Example ____ 7

1 Introduction

This document describes how to provide Fabasoft Folio notifications using Fabasoft app.ducx.

The necessary steps for administrating notifications are described in the administration help (https://help.folio.fabasoft.com/index.php?topic=doc/Administration-Help-Fabasoft-Folio-eng/index.htm).

2 Software Requirements

System environment: All information contained in this document implicitly assumes a Microsoft Windows environment or Linux environment.

Supported platforms: For detailed information on supported operating systems and software see the software product information on the Fabasoft distribution media.

3 Getting Started

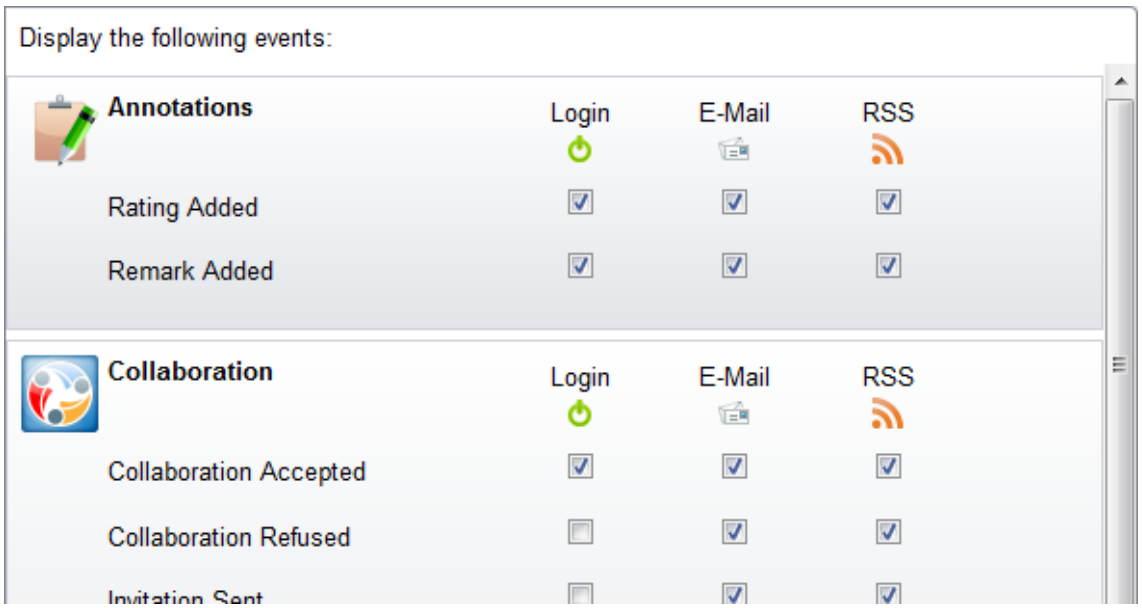
Notifications programmatically consist of the following two elements:

- *Event Type* (`FSCFOLIO@1.1001:HistoryEventType`)
  An object of object class *Event Type* is needed to represent the type of event that happened and it is used for the representation to the user.
- *Log an event to objects history* (`FSCFOLIO@1.1001:LogHistoryEvent`)
  This action creates a history entry.

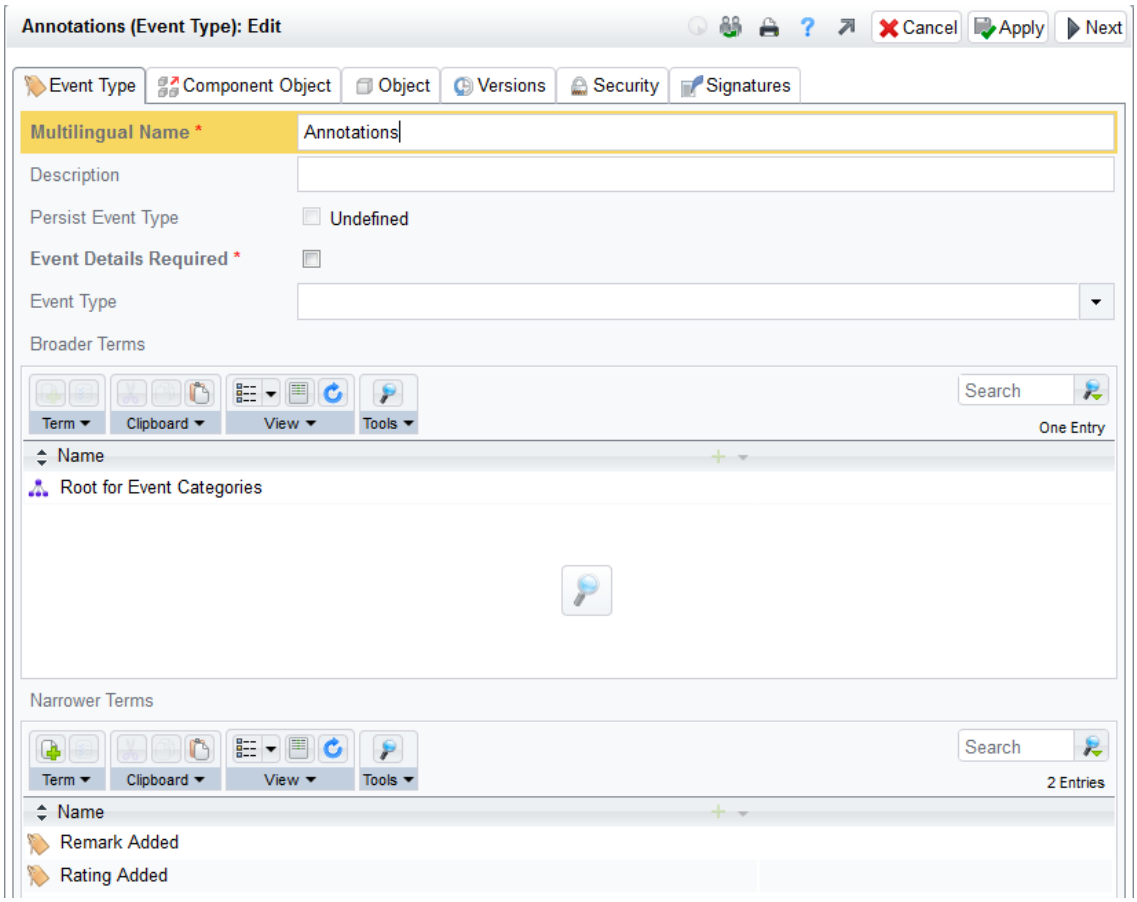
4 HistoryEventType

On the one hand objects of object class *Event Type* (`FSCFOLIO@1.1001:HistoryEventType`) define an event (e.g. "Content Changed" or "Document Removed") and on the other hand they are also used for the categorization of event types.

The following example shows the event types “Annotations” and “Collaboration” that are used as categories. The event types like “Rating Added” or “Collaboration Accepted” are assigned to the category event types.



The properties *Broader Terms* (`FSCTERM@1.1001:broadercompterms`) and *Narrower Terms* (`FSCTERM@1.1001:narrowercompterms`) are used to define which event type belongs to which category.



Event types for categories are prefixed with `EC` and event types for events are prefixed with `ET`.

Root for Event Categories (`FSCFOLIO@1.1001:EC_Root`) is the root element containing the categories in the *Narrower Terms* property. By default the following categories for event types are available:

- *Annotations* (`FSCFOLIO@1.1001:EC_Annotate`)
- *Uncategorized* (`FSCFOLIO@1.1001:EC_Other`)
- *Common Changes* (`FSCFOLIO@1.1001:EC_General`)
- *Workflow and Signatures* (`FSCFOLIO@1.1001:EC_Workflow`)
- *Collaboration* (`FSCFOLIO@1.1001:EC_Collaboration`)
- *Security* (`FSCFOLIO@1.1001:EC_Security`)
- *Newsfeed* (`FSCFOLIO@1.1001:EC_Social`)

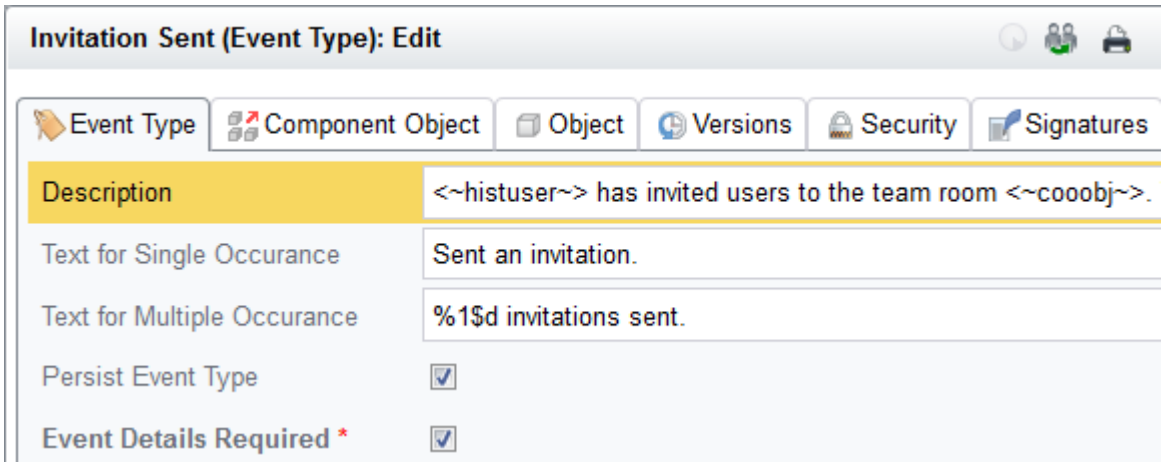
If no category is available for an event type, the *Uncategorized* category is used.

The following properties can be defined for an event type:

- `FSCFOLIO@1.1001:hetdescription`
  This text describes the event and is used for displaying the event in the welcome screen, in an e-mail or as RSS feed entry.
  The following placeholders can be used within the description text:
  - `<~histuser~>`
    The user who triggered the event.
  - `<~cooobj~>`
    The object on which the event has been triggered.
  - `<~histevent~>`
    An object related to the event, e.g. the object that has been added to a folder (`<~cooobj~>`).
  - `%s`, `%d`
    Placeholders for arguments.
- `FSCFOLIO@1.1001:hetsingletext`
  In case of an accumulated display of events this text is used instead the `hetdescription` text, if the event occurred once.
  Note: An accumulated display appears if more than three events need to be displayed in the welcome screen or in an e-mail. If more than three events occurred, the three most recent events are displayed as usual and all the other events are displayed accumulated.
- `FSCFOLIO@1.1001:hetmulitpletext`
  In case of an accumulated display of events this text is used if the event occurred more than once. In this case `%d` is a placeholder for the number of events.
- `FSCFOLIO@1.1001:hetpersist`
  Defines whether events of this type should not be deleted from an object's history in case of the maximum number of history entries has exceeded.
- `FSCFOLIO@1.1001:hetrequiredetails`
  Defines whether the `<~histevent~>` object has to be a valid object.

Note: The values for `hetdescription`, `hetsingletext` and `hetmultipletext` have to be defined in the language files.

Example:



5 LogHistoryEvent

To log an event to an object's history, the action *Log an event to objects history* (`FSCFOLIO@1.1001:LogHistoryEvent`) has to be called on the object, on which the event occurs (e.g. when renaming an object).

The following parameters can be defined:

- `type`
  The event type for which the event should be logged.
- `event`
  An object to be passed for the `<~histevent~>` placeholder.
- `description, unused`
  Deprecated parameters.
- `user`
  The user who triggered the event.
- `group, position, substuser`
  The group, position and substitution information of the user who triggered the event.
- `docstate`
  A document state, which can be passed in case a document state has changed.
- `args`
  A Fabasoft app.ducx expression returning a string list of any additional information.

6 Example

This example illustrates the use of *Event Type* (`FSCFOLIO@1.1001:HistoryEventType`) and *Log an event to objects history* (`FSCFOLIO@1.1001:LogHistoryEvent`).

In this example a new history entry is created when a trip is added to a logbook.

instances.ducx-om

```
objmodel FSCLOGBOOK@111.100
{
  /**
   * Event type for recorded trips
```

```
   */
  instance HistoryEventType ET_NewTrip {
    symbol = SymbolRecordTrip;
    hetdescription = {}
    hetsingletext = {}
    hetmultipletext = {}
    hetpersist = false;
    hetrequiredetails = false;
    broadercompterms = {
      EC_DriversLogbook
    }
  }


  /**
   * Category for event types of driver's logbook
   */
  instance HistoryEventType EC_DriversLogbook {
    symbol = SymbolLogbook;
    narrowercompterms = {
      ET_NewTrip
    }
    broadercompterms = {
      EC_Root
    }
  }


  // Add category "EC_DriversLogbook" to the root of event categories "EC_Root"
  extend instance EC_Root {
    narrowercompterms = {
      EC_DriversLogbook
    }
  }


  // Define default values for event type "ET_NewTrip"
  extend instance DefaultConfig {
    cfgnotificationsettings<notsettingstype, component, notsettingsnotification,
                            notsettingsmail, notsettingsrss> = {
      { ET_NewTrip, FSCLOGBOOK@111.100, true, true, true }
    }
  }
}
```

usecases.ducx-uc

```
usecases FSCLOGBOOK@111.100
{
  /**
   * Record a new trip in a trip log
   */
  RecordTrip(Trip trip) {
    variant TripLog {
      impl = expression {
        cooobj.ObjectLock(true, true);
        /*
         * Add trip to trip list
         */
        cooobj.trltrips += trip;
        /*
         * Log the recording of a trip in the history
         */
        cooobj.LogHistoryEvent(#ET_NewTrip, null, null, null,
                               null, null, null, null, null,
                               [cooobj.Escape(trip.trpdepartureplace,true),
                                cooobj.Escape(trip.trpdestinationplace,true)]);
      }
    }
  }
}
```

LANG_ENGLISH

Identifier	Value
ET_NewTrip.hetdescription	<~histuser~> recorded a trip in <~cooobj~> from %1$s to %2$s.
ET_NewTrip.hetmultipletext	%d trips recorded.
ET_NewTrip.hetsingletext	One trip recorded.
ET_NewTrip.mlname	Trip Recorded