Activities

Release 15.3.1.4

CONTACT Software

Jun 04, 2018

Contents

CHAPTER 1

Introduction

CONTACT Activities shows current discussions and system messages in a continuous activity stream, which are either relevant to your working context or are intended for all users. Subscribed channels are used to control what information is displayed. *CONTACT Activities* also allows objects to be shared between users.

CHAPTER 2

Activities

2.1 Subscribe Topics

Is Activity Stream Channel

Each object with a `cdb_object_id` can be a topic to which postings can be assigned. As an alternative to *Topic*, the term *Channel* is also used. In the global stream, users currently see postings only if they have subscribed to the corresponding topic. If this box is checked, the `CDB_SubscribeToChannel` or `CDB_UnsubscribeFromChannel` operations are automatically assigned to the class or removed.

2.2 System Posting

The system can automatically generate a system posting in the case of a new creation, modification or status change; this message then appears in *Activities*. System postings are generated in all active languages of the installation (see multilanguage-activate).

Note: For performance reasons, parts of the code for creating system postings are executed by a separate service. The installation and operating manual describes how to activate this service, called `System Posting Queue`. The `cs.activitystream.activitylistener` has to be enabled for jobs to be generated for the queue. This is not automatically the case, for example, for pure PowerScript applications.

The base configuration is set in the class properties (see datadict-class-properties).

The attributes concerning all actions in the Activity Stream are defined as follows:

2.2.1 General Activities configuration in the class definition

Create System Postings

In order for system postings to be created at all for objects of a class, this check box has to be selected. If it is selected, the setting also applies for derived classes. This is true regardless of whether or not the check box is selected there. If it is not selected, all other configurations have no impact on the Activity Stream.

System postings can only be created for classes with the `cdb_object_id` attribute. When classes are created or modified, the system checks whether this attribute exists. If the attribute is not present, it is added automatically. The class then usually has to be generated manually using the *Compile* function.

Rule

Here, an object rule can be referenced, which limits the amount of objects for which a system posting is to be created. For example, you could specify that postings are to be created only for tasks of a certain category.

If a rule is specified, this rule has to be fulfilled in this class and in all derived classes, so that a posting is created. If a rule is defined in a derived class, it has to apply also for all rules defined in base classes of this class.

The other attributes are described in the following chapters.

2.2.2 Generation of postings when creating an object

If the basic requirements for generating system postings are satisfied (see *General Activities configuration in the class definition* (page 2)), other settings can be used to determine whether a posting is to be generated when creating an object. The base configuration is set in the class properties (see datadict-class-properties).

Activities configuration for object creation

The relevant attributes are defined as follows:

Generate System Postings on Object Creation

If this check box is set, a posting is created when creating an object of the class. If it is selected, the setting also applies for derived classes. This is true regardless of whether or not the check box is selected there.

Text

An (error) message can be referenced here via a label. The text of the system posting is preset in this message. The evaluation is performed in the same way as for the configuration of object description in the Data Dictionary, meaning that fixed elements and attribute values can be concatenated. If no text is configured, the message `activity_obj_created` is used and the object description configured in the Data Dictionary is used as a replacement.

Channel for Creation Postings

Sometimes users want to be notified about the creation of objects by means of the activities. In the case of objects not assigned to any topic, a channel can be created, for example, the *New Ideas* channel for idea management (see *Access rights for postings* (page 4)). Then a user who wants to see new ideas in the global activities window can subscribe to this channel. If this channel is referenced in the field described here, a Creation System Posting is automatically assigned to the channel, regardless of implementing the PowerScript method `GetActivityStreamTopics`.

2.2.3 Generation of postings when there are changes

If the basic requirements for generating system postings are satisfied (see *General Activities configuration in the class definition* (page 2)), other settings can be used to determine whether a posting is to be generated when changing objects. The configuration is performed in the *Activities* tab of the attribute properties in the data dictionary (see cdb-classdef-attributes).

Activities configuration for changing an object

These attributes are defined as follows:

Create System Posting on Change

If this check box is set, a check is performed when changing an object to see if this attribute has been changed. If this is the case, a posting is generated.

Text

An (error) message can be referenced here via a label. The text of the system posting is preset in this message. The evaluation is performed in the same way as for the configuration of object description in the Data Dictionary, meaning that fixed elements and attribute values can be concatenated. If a text is configured, a separate posting is generated with the text – i.e., if multiple attributes are changed and a text is configured for each of these attributes, multiple postings are also generated.

If no text is configured, the message `activity_obj_modified` is used. Here, all changed attributes are collected whose change caused a posting, and for which no special message is configured. The object description and the list of the changed attributes with their values are used as replacements for the message.

2.2.4 Generation of postings in the case of a status change

If the basic requirements for generating system postings are met (see *General Activities configuration in the class definition* (page 2)), other settings can be used to define whether a posting is to be generated in the case of a status change. The settings are made when configuring the status definition.

Activities configuration for status change

These attributes are defined as follows:

Generate Posting on Interactive Status Change

If this checkbox is selected, a posting is generated in the case of an interactive status change to this status.

Generate Posting on Batch Status Change

If this checkbox is selected, a posting is generated in the case of an automatic status change to this status. These are the status changes performed by the system, for example, by calling status changes within user exits.

Text

A message from the (error) message management system can be referenced here via a label. The text of the system posting is preset in this message. The evaluation is performed in the same way as for the configuration of object descriptions in the Data Dictionary, meaning that fixed elements and attribute values can be concatenated.

If no text is configured, the message `activity_obj_wfstep` is used. The object description and the target state are used as replacements for the message.

2.2.5 Assignment of the topics

A posting can be assigned to multiple topics. For example, the posting that a new task was created can be assigned to the Activity of the assigned project and to the Activity of the task. The method `GetActivityStreamTopics` of the PowerScript class `cdb.objects.Object` is used for making assignments. For details, please refer to the PowerScript manual.

2.2.6 Access rights for postings

By default, the read right for postings depends on the `read` right of the object to which the posting refers. In the kernel, this is mapped similarly to a relationship access profile, i.e., the right is not evaluated for a search via `CDB_Search`. The right is evaluated at the points where postings are usually displayed, i.e., in the Enterprise-Search or in the Activities.

2.2.7 Automatic deleting of postings

If an object is deleted, the postings assigned to this object are automatically deleted with it.

2.3 Display of the Activities as tab

You can use a cdb-gui-maskconfig-dlgitemtypes-sect-cdbelinkcontrol to have the Activities displayed for a specific topic/channel. The system already has a preset mask containing just this control. This mask is called `cdb_elink_activitystream`. It can be used as a tab in a mask composition. When creating an object and when searching, the eLink application remains inactive. Therefore, it is generally advisable either to define different mask compositions for the different actions or to hide the tab using the PowerScript function `disable_registers()`. This occurs in the `Channel` class, for example:

```
class Channel(Object):
    __maps_to__ = "cdbblog_channel"
    __classname__ = "cdbblog_channel"

    def _skip_activity_register(cls, ctx):
        if ctx.action != "info" and ctx.action != "modify":
            try:
                ctx.disable_registers(["cdb_elink_activitystream"])
            except:
                # Some context adaptors does not support disable_registers
                pass

    event_map = {("*", "pre_mask"): "_skip_activity_register"}
```

2.4 Channels

For a posting to become visible to the user, there has to be a topic to which the posting is assigned. In many cases, this topic is an object, e.g., a project. In some cases, however, no suitable object exists. When this is the case, you can create a channel to which postings are assigned. An example of such a channel is the *New ideas* channel. As described in *Activities configuration for object creation* (page 3), you can configure settings so that the system postings for creating an object are assigned to a channel. The configuration for channels can be accessed in the menu tree *Administration/Configuration* → *Activities* → *Channels*. By default only administrators are permitted to create new channels there.

CHAPTER 3

Daily Activities as E-Mail

This is a user opt-in function which requires the service `cs.activitystream.daily_mail_service.DailyMailServi` to run. The service itself relies on the mail server configuration (usually found in `$CADDOK_BASE/etc/site.conf`).

If the service is up and running, it will send e-mails once a day. The default starting time is 0am and can be modified by adding the parameter "–start" to the service, specifying a value like "20:15" (formatted "%H:%M"). Don't forget to restart the service afterwards. Please note that such a change should be made during the weekend, as the service always looks at the postings and comments changed during the last 24 hours. If you change the service's starting time, some objects may be reported as new twice, or not at all.

The e-mail contains the full text of all of yesterday's postings and comments. If you want only excerpts of the text, you can modify the e-mail template.

The template can be found in `chrome/activity_digest.html` in the module directory. To customize it, you should first copy it to your custom module and overwrite the class `cs.activitystream.daily_mails.DailyMailer` to use your copy. You will have to specify the new directory the template can be found in, for example like this:

```
import os
from cdb import CADDOK
from cs.activitystream.daily_mails import DailyMailer

class CustomMailer(DailyMailer):
    __notification_template_folder__ = os.path.join(CADDOK.BASE, "email_templates")
    __notification_template__ = "activity_digest.html"
```

CHAPTER 4

Share Objects

4.1 Operation Logic

The Operation `Share (cdb_share_objects)` may be called with or without context. It opens the URL of the "Share" dialog, and appends the cdb_object_ids of any context objects.

- URL without context: `$CADDOK_WWWSERVICE_URL/share_objects`
- URL with context: `$CADDOK_WWWSERVICE_URL/share_objects?attachments=eb5af880-4be0-11e0-a01`

The dialog requires the user to select at least one attachment and one recipient. Since common roles and recipient lists may be used as recipients, the operation can only determine if at least one valid recipient was selected, when the user confirms the dialog.

After confirming the dialog, an object of class `cdb_sharing (cs.sharing.Sharing)` will be created. It is used as an activity channel, with an initial posting containing the user message and selected attachmenmts. The `cdb_sharing` object itself contains only metadata related to change control, e.g. creation date and creator (`cdb_cdate`, and `cdb_cpersno`, respectively).

Nachrichten abonnierter Kanäle für das `cdb_sharing` Objekt sowie eventuelle E-Mail Benachrichtigungen (einstellbar in den persönlichen Einstellungen) werden asynchron durch eine Message Queue erzeugt. Wartende und fehlgeschlagene Aufträge können Sie unter *Administration/Konfiguration → Teilen → Teilen-Aufträge* einsehen und ggf. neu starten. Die in der E-Mail verwendete Sprache können Sie über Dienstoption `--language` für den Dienst `cs.sharing.share_objects_server.ShareObjectsServer` festlegen.

4.2 Recipient Lists

Users may save the currently selected recipients as a personal recipient list. After providing a name, an object of class `cdb_personal_sharing_group (cs.sharing.groups.PersonalSharingGroup)` will be created, and the selected recipients will be added as members by creating entries of class `cdb_sharing_group_member (cs.sharing.groups.SharingGroupMember)`.

At this time, valid recipients are users (`angestellter`), common roles (`cdb_global_role`), and recipient lists themselves, although this limitation is not enforced when creating a recipient list.

Recipient lists may be defined for roles (for ex. "public") by administrators. If a recipient list is defined for a role, all of the role's members can use the recipient list, but only administrators may change it.

4.2.1 Object-Specific Recipient Lists

Objects to be shared may offer dynamic recipient lists (class `cdb_object_sharing_group (cs.sharing.groups.ObjectSharingGroup)`), which may be selected as recipients. These lists are predefined, and will be added at runtime to all objects matching the rule.

Lists are resolved either using a method of the object's Powerscript class, or one of its attributes. The former has to return a list of tuples (`subject_id`, `subject_type`), the latter must contain a user ID (`angestellter.personalnummer`).

4.3 Embedded Enterprise-Search

The "Share" Dialog utilizes Enterprise-Search for powering its search box. Please make sure all required services are up and running.

4.4 Setup for Custom Classes

To make objects shareable, you will first have to enable its class for the REST API by setting "REST API active" and a unique "REST Name" in the class configuration.

4.4.1 Enable Operation

To make the operation `Share` available in your custom class's context menu, enable the operation `cdb_share_objects` (*Administration/Configuration* → *Administration* → *Operations*) for your class. Then, add code corresponding to the following example to its Powerscript class:

```
from cs.sharing.share_objects import WithSharing
from cs.documents import Document

class MyClass(Document, WithSharing):
    pass
```

4.4.2 Set Up Object-Specific Recipient Lists

Object-specific recipient lists can be set up like this:

1. Create a new recipient list or select an existing one at *Administration/Configuration* → *Share* → *Recipient Lists*
2. The recipient list's object rule must contain a predicate with at least one term for your class. Only objects matching the rule will show the recipient list in the dialog.
3. If the recipient list is not a simple attribute containing a user ID (`angestellter.personalnummer`), add code to the Powerscript class to resolve it. For example:

```
from cdb.objects import Object

class myObjectsClass(Object):
    def getCustomRecipients(self, sharingGroup):
        "Empfängerliste auflösen/resolve recipient list"
        return [(self.cdb_cpersno, "Person"),
                (self.subject_id, self.subject_type),
                ("Administrator", "Common Role")]
```