Documents

Release 15.2.3.7

CONTACT Software

Oct 29, 2018

Contents

CHAPTER 1

Introduction

Managing, utilizing and ensuring the availability of documents are key tasks in any company. In view of the vast number of documents, interfaces and systems involved, conventional archives on network drives or in email systems lead to a dead end. *CONTACT Documents* provides a clear overview using intelligent document logistics – even in complex processes – while at the same time supporting team work.

CHAPTER 2

The standard access system

2.1 Roles

The roles listed below are used by `cs.documents` to define the access to documents. Details are explained in the *Access Control System* (page 2) chapter of this documentation.

- `public`: Read access to documents
- `Documentation`: Users that should be able to create or change documents
- `Administrator`: Nearly full access to all documents

2.2 Access Control System

The standard access system grants the `read` access to every user with the role `public` which allows the user to display the attributes of a document and to view derived files. This access is granted with the *Access Control Domain* ‘‘cs.documents.Document‘.

The *Access Control Domain* `cs.documents: Documents - Unprotected` grants the `read_file` access to every user with the role `public` which allows the user to read all files. In the standard this affects all documents of the class `document`. Documents of derived classes like `model` are not affected by this *Access Control Domain*. You might change the predicate `cs.documents: Unprotected documents` to define which documents should grants the access to the non derived files to `public`.

Users that have to create, modify or edit documents should have the role `Documentation`. These role grants nearly full access for documents with the status `Draft` (ACD `cs.documents: Documents (DRAFT)`) and read access - including the original files - for documents in revision (ACD `cs.documents: Documents (REVISION)`, review (ACD `cs.documents: Documents (REVIEW)`, for blocked or obsolete documents (ACD `cs.documents: Documents (BLOCKED, OBSOLETE)`) and for released documents (ACD `cs.documents: Documents (RELEASED)`).

The standard rejects any attempt to change a released document. The *Access Control Domain* `cs.documents: Documents (RELEASED)` defines an exclusive `save` access granted to nobody.

You might adjust the predicates of the access control domains if you do not use the standard object lifecycles `doc_standard` and `doc_approve`.

Note that the access system is influenced by other modules. At this time `cs.shared` provides an access control domain that grants full access on documents to the administrator.

CHAPTER 3

Defining the default document class

The class can be configured using the `ddc` (default document class) property. The class is used if an action is run on documents where the class is not specified by any context. Creating a new document by dragging from the Microsoft Windows desktop and dropping into the empty area of the main window in the system client or the file editing window provides an example of one type of this context-free action. If the property is not set, the `document` class is used.

This means that setting this property is useful if different instances of access to document management have been configured for different user groups.

CHAPTER 4

Dashboard Widgets

4.1 `my-documents` - *Last modified Documents*

The documents that appears in this widget are defined by the *Object Rule* `cs-web-dashboard: My Documents`.

CHAPTER 5

Ident mechanism

A document is uniquely identified in the system using a number and an index. However, there are situations where the actual document number together with the version is not unique. For example, a document could consist of several sheets. In this case, it is necessary to separate the technical document number (`zeichnung.z_nummer`) from the document number that appears in the *Document No.* field.

The ident mechanism provides an option to address this issue. In order to use the mechanism, the `iden` property has to be set to `true`. In addition, the class `document` has to contain the `ident` attribute. This is then assigned to the document number in the masks. If the ident mechanism is active, the user identifies a document using the ident the system used to generate the document number for the entry in the database instead of the original document number. The number is generated before calling the standard user exits at the `pre` execution time.

The calculated document number consists of a part of the ident and sequential number. The `ipre` property specifies how many digits of the ident are taken over in the document number. The `isuf` property determines how many digits the sequential number is to consist of, which are then attached to the number.

CHAPTER 6

Batch operations

6.1 *Change Status*

The standard solution of *cs.documents* (page 7) contains an implementation that allows you to change the status for several documents in one step.

From a technical point of view the batch operation has 4 parameters

- `param1`: The integer value of status as stored in `z_status`.
- `param2`: The status name
- `param3`: A flag that determines if documents that are referenced by this document should be included to the batch operation. If the flag is set and the status of a document changes while executing the batch operation the system tries to change the status of the referenced documents in the same way. If the status of a document is already the target status the status of referenced documents is not changed. The parameter is not visible in the standard mask that parametrizes the batch operation. If the status of referenced documents depends on the status of the referencing document you should implement that dependency in your customizing and the behaviour should not depend on the way the user changes a status.
- `param4`: If the value is `1` the document will be unlocked as part of the after the status has changed.

CHAPTER 7

PowerScript API

PowerScript is the development environment of CONTACT Elements, which enables you to develop add ons, user exits, stand alone scripts or complex applications. The scope of this documentation is the description of the classes and functions provided by this pacakge.

Contents:

7.1 cs.documents

7.1.1 Dialog Hooks

The module contains some dialog hooks used by *cs.documents* (page 7)

cs.documents.std_hooks.**CheckItemReference**(*hook*)
Check if an item reference is obligatory for a document with the given category.

cs.documents.std_hooks.**OLCCheckReferencedHook**(*hook*)
Hook that checks if referenced documents have an appropriate state. If not, the user will be asked if he wants to continue the action.

Note that this hook will only work with the standard definitions of the object lifecycle.

7.1.2 Document

class cs.documents.**Document**(***values*)

CalculateOLC(*ctx*)
The function is called to calculate the object life cycle from the attributes provided by ctx.dialog or self. You can overwrite this function to adapt the system behaviour. If there is no rule, you should return `None`. The function is used to prevent object modifications that implies a change of the object life cycle if the document state is not `0`.

classmethod CalculateSourceOID(*ctx*)
Called by the framework to set the `source_oid` attribute during the creation of a document. The default returns the `cdb_object_id` of the template if you use *Create from template* and the `cdb_object_id` of the copy source if the `CDB_Copy` operation is used.

CreateIndex(*new_index=''*, ***kwargs*)
Creates and returns a new version of `self`. You can use `new_index` to predefine the value of `z_index`. If `new_index` is emtpy it is generated using the index schema of your installation.

GetActivityStreamTopics(*posting*)
Returns the Activity stream topics where a posting to the document should occure. The default implementation assigns a posting to the project and the object itself.

classmethod GetDefaultErzSystem()
Retrieve the default that has to be used to initialize the attribute `erzeug_system` from the personal setting `cs.documents.default_cad`.

classmethod GetInitialCopyValues()
The function is called for `CDB_Copy` to retrieve values as `dict` that should not be copied from the previous version. For interactive operations these attributes are set during the `pre_mask` call. For batch operations the attributes are set in `pre`.

classmethod GetInitialCreateValues()
The function is called for `CDB_Create` to retrieve initial values as `dict`. For interactive operations these attributes are set during the `pre_mask` call. For batch operations the attributes are set in `pre` if no value is supplied before.

classmethod GetInitialIndexValues()
The function is called for `CDB_Index` to retrieve values as `dict` that should not be copied from the previous version. For interactive operations these attributes are set during the `pre_mask` call. For batch operations the attributes are set in `pre`.

GetOLCRelevantAttributes()
The function should return a list of attributes that are relevant for *CalculateOLC* (page 7) to determine the object life cycle.

GetReadonlyAttrsIfOLCStarted()
Returns a list of attributes that should be set readonly if the document has left the initial state. The default implementation calls *GetOLCRelevantAttributes* (page 8).

GetReviewer()
Returns a list of personal numbers of the persons that should review or have reviewed the document. This is e.g. used to implement the recipient list *Reviewer* of the sharing. The default tries to map the value of the attribute `pruefer` to the personal no.

HandleUnlockForeignLock(*f*, *previous_locker*, *ctx*)
The framework calls this function as part of the `post` user exit of the `CDB_Unlock` action of the class `cdb.objects.Object.cdb_file.CDB_File` if the user has unlocked a file locked by the user identified by `previous_locker`. The function generates a sharing to send this information to the previous locker. You might overwrite the function to adapt this behaviour.

IsValidInteractiveTargetStatus(*target_status*)
Called by the framework as an additional check if a target status should be selectable by an user. If you return `False` the standard implementation ensures that the status is excluded in the target status catalog of the operation `CDB_Workflow`. Trying to set the status of `self` to `target_status` using a batch operation will also fail. You can overwrite this function to exclude status from interactive use.

PreviousIndex
A `cdb.objects.ReferenceMethod_1` that returns the document that represents the previous version of `self` usually by selecting all versions and sorting them using the configuration of the property `ixsm` and `z_index`. This is quite expensive - if you know your indexing schema you should use an implementation that do not need DB-Statements.

Note that this function will not work for a 'delete'-'post' user exit if your sorting uses any other attribute from `zeichnung` than `z_index` for sorting.

copyDoc(***kwargs*)
Returns a copy of self using `makeNumber` to generate a new document number. The function uses `cdb.objects.Object.Copy` which means at this time that there is no access check and no user

exits are called. The function also copies the non derived files of the document using `cdb.objects.cdb_file.CDB_File.copy_file`. At this time this function does not reset the locking information.

The function might be replaced in future versions - you should use `cdb.objects.operations.operation` for the regular behaviour.

get_template_preset_attributes()
Return a list of attribute names that are to be copied from the template document, when the operation *New from template* is called. You may overwrite this method to customize the list of attributes.

resolveReferencedDocuments(*depth=0*)
Returns a list of all documents that are referenced by `self` by navigating `DocumentReferences` until depth is reached or a recursion occures. If depth is `0` all references will be returned. The result does not contain any duplicates and will be sorted in a way that if `D1` references `D2` that `D1` is located before `D2` in the result as long a there are no recursions.

If a document reference points to an invalid document this reference will be ignored.

List of Figures

List of Tables

Python Module Index

d