Crux: BRE/WFE - Task #17851

WSC: WFinstanceAbort()

15/02/2024 09:40 AM - Sachin Divekar

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Kanchan Barve	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Sprint 29 Feb		

Description

History

#1 - 17/02/2024 05:27 PM - Shuvam Misra

- Status changed from New to In Progress

#2 - 17/02/2024 05:27 PM - Shuvam Misra

- Target version set to Sprint 29 Feb

#3 - 20/02/2024 11:39 AM - Kanchan Barve

- Assignee set to Kanchan Barve

#4 - 20/02/2024 12:27 PM - Kanchan Barve

WFinstanceAbort()

Request

```
{
    "id": 853
    "entityid": "6cf3093c-b078-11ee-bc69-13755d95865a"
}
```

where **any one, but not both** of these parameters will be present.

Processing

The call will be open to any authenticated user from one of the whitelisted IP addresses. No specific capability is required.

All records matching the specified parameter will be deleted from wfinstance. All records whose parent field matches one of the records being deleted will also be deleted.

This call is useful when a workflow traversal needs to be aborted, either because the entity itself is being purged from the application, or because the traversal needs to be restarted for some reason with a fresh call to WFinstanceNew().

Response

The data block in the response will be an empty block if the call is successful.

#5 - 20/02/2024 08:02 PM - Kanchan Barve

- % Done changed from 0 to 40

package wfinstance

import (
"github.com/gin-gonic/gin"
"github.com/jackc/pgx/v5/pgtype"
"github.com/remiges-tech/alya/service"
"github.com/remiges-tech/alya/wscutils"
"github.com/remiges-tech/crux/db/sqlc-gen"
"github.com/remiges-tech/crux/server"
"github.com/remiges-tech/crux/types"
)

```
// AbortWFInstance rquest format
type WFInstanceAbortRquest struct {
ID      *int32  json:"id"
EntityID *string json:"entityid"
}

func GetWFInstanceAbort(c *gin.Context, s *service.Service) {
lh := s.LogHarbour
lh.Debug0().Log("AbortWFInstance request received")

var (
    request  WFInstanceAbortRquest
    id       int32
    entityid string
)

isCapable, _ := types.Authz_check(types.OpReq{
    User: USERID,
}, false)

if !isCapable {
    lh.Info().LogActivity("Unauthorized user:", USERID)
    wscutils.SendErrorResponse(c, wscutils.NewErrorResponse(server.MsgId_Unauthorized, server.ErrCode_Unauthor
ized))
    return
}

// Bind request
err := wscutils.BindJSON(c, &request)
if err != nil {
    lh.Debug0().LogActivity("error while binding json request error:", err)
    return
}
//validate request
if request.ID != nil && request.EntityID != nil {
    lh.Debug0().Log("Both ID and EntityID cannot be present at the same time")
    wscutils.SendErrorResponse(c, wscutils.NewErrorResponse(server.MsgId_Invalid, server.ErrCode_InvalidReques
t))
    return
}
if request.ID == nil && request.EntityID == nil {
    lh.Debug0().Log("Either ID or EntityID must be present")
    wscutils.SendErrorResponse(c, wscutils.NewErrorResponse(server.MsgId_Invalid, server.ErrCode_InvalidReques
t))
    return
}

//Handle the request based on the presence of ID or EntityID
if request.ID != nil {
    id = *request.ID
} else {
    entityid = *request.EntityID
}
lh.Debug0().LogActivity("present values :", map[string]any{"ID": id, "EntityId": entityid})

query, ok := s.Dependencies["queries"].(*sqlc.Queries)
if !ok {
    lh.Log("Error while getting query instance from service Dependencies")
    wscutils.SendErrorResponse(c, wscutils.NewErrorResponse(server.MsgId_InternalErr, server.ErrCode_DatabaseE
rror))
}

error := query.DeleteWfInstance(c, sqlc.DeleteWfInstanceParams{
    ID:       pgtype.Int4{Int32: id, Valid: id != 0},
    Entityid: pgtype.Text{String: entityid, Valid: !types.IsStringEmpty(&entityid)},
})
if error != nil {
    lh.LogActivity("error while deleting wfinstances based on id  :", error.Error())
    return
}


wscutils.SendSuccessResponse(c, &wscutils.Response{Status: wscutils.SuccessStatus})

}
```

```
-- name: DeleteWfInstance :exec
WITH deleted_parents AS (
DELETE FROM public.wfinstance
WHERE
(id = sqlc.narg('id')::INTEGER OR entityid = sqlc.narg('entityid')::TEXT)
RETURNING parent
)
DELETE FROM public.wfinstance
WHERE parent IN (SELECT parent FROM deleted_parents)
```

#6 - 21/02/2024 02:29 PM - Kanchan Barve

- % Done changed from 40 to 50

#7 - 21/02/2024 06:45 PM - Kanchan Barve

- % Done changed from 50 to 80

- modified delete query and added test cases

#8 - 26/02/2024 02:43 PM - Kanchan Barve

- Status changed from In Progress to Testing

#9 - 27/02/2024 03:03 PM - Kanchan Barve

- % Done changed from 80 to 100

- Status changed from Testing to Closed

#10 - 27/02/2024 07:05 PM - Kanchan Barve

- Added request validation.
- Added Data change Log.
- Modified Delete query to return all data for data change log.
- Testing with new Changes done.