LOGALI

Teoría

EML – Autorizaciones

ABAP RESTful – Arquitectura Cloud

Contenido

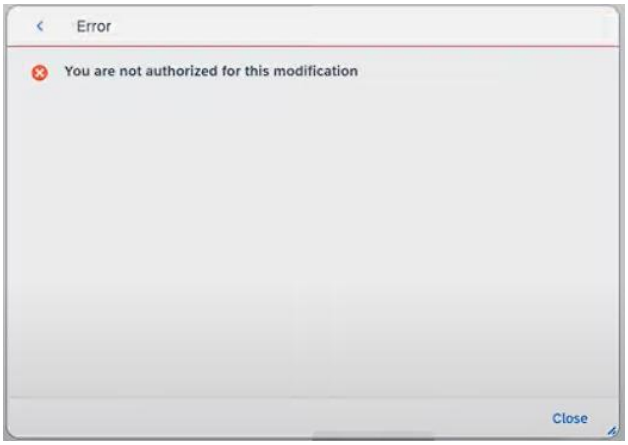
1. EML – Autorizaciones

Las autorizaciones son un componente crucial para asegurar que solo los usuarios adecuados puedan realizar determinadas operaciones sobre los datos de negocio.

1.1. Autorización Global – Creación

Las autorizaciones globales en RAP (ABAP RESTful Application Programming) son un mecanismo fundamental que permite controlar de manera centralizada el acceso a las operaciones sobre los datos de una aplicación. Estas autorizaciones determinan si un usuario tiene permiso para ejecutar acciones como crear, actualizar o eliminar registros, sin depender de los datos específicos de una instancia.

Las autorizaciones globales se aplican a toda la aplicación y son independientes de los datos específicos de las entidades. Están diseñadas para proporcionar un nivel de control más amplio y se definen en el Behavior Definition del modelo RAP. Este tipo de autorización evalúa si un usuario tiene los permisos adecuados para realizar operaciones específicas globales en la aplicación, como la creación, eliminación, actualización de registros existentes e incluso acciones personalizadas.



Implementación de Autorizaciones Globales:

La definición de las autorizaciones globales se definen en el Behavior Definition asociada a una entidad raíz por medio de la sentencia authorization master añadiendo la propiedad global:

authorization master (global, instance)

La implementación de la lógica global de autorización se implementa en la clase Behavior Pool asociada al Behavior Definition en el método **get_global_authorization.**

Es importante agregar las propiedades de autorizaciones **global** e **instance**, que se utilizarán para controlar los niveles de acceso de los usuarios, antes de generar la clase del Definition Pool. Esto es necesario porque la clase declara los métodos pertinentes si previamente se especificaron por medio de la sentencia authorization master. En el caso contrario se debe realizar este proceso utilizando la ayuda del framework a través del botón  para generar la lógica de las autorizaciones globales. El método **get_global_authorization,** se declara de la siguiente forma:

```
METHODS get_global_authorizations FOR GLOBAL AUTHORIZATION
  IMPORTING REQUEST requested_authorizations FOR
root_entityname or alias RESULT result.
```

Condiciones de Autorización:

El método **get_global_authorization,** es fundamental para verificar las autorizaciones globales del usuario en una aplicación. A continuación, se describen los parámetros clave de este método y sus funciones:

- **requested_authorizations**: Este parámetro es una estructura que contiene las autorizaciones solicitadas para la operación actual. Incluye información sobre el tipo de operación que se desea realizar (creación, actualización, eliminación, etc.). Estas operaciones pueden ser identificadas en el método mediante la evaluación de los campos de la estructura, utilizando las constantes **mk** de la interfaz **if_abap_behv**, con el valor **on**. La

estructura dentro del método get_global_authorization posee los siguientes campos que identifican la operación solicitada:

- **%create:** Identifica la operación para crear registros por medio de un endpoint.
- **%delete**: Identifica la operación para eliminar registros por medio de un endpoint.
- **%update**: Identifica la operación para actualizar registros por medio de un endpoint.
- **%action**: Este campo representa la acción específica las validaciones personalizadas entre otros aspectos definidos en el Behavior Definition.

- **result:** Este parámetro es una estructura que se utiliza para devolver el resultado de la solicitud de autorización. Indica si la operación solicitada ha sido autorizada o denegada. La estructura dentro del método get_global_authorization, posee los mismos campos que la estructura requested_authorizations con la diferencia que permite establecer la autorización de las operaciones utilizando la estructura de constantes **auth** de la interfaz **if_abap_behv**, con los valores **allowed** y **unauthorized**.
- **reported:** Este parámetro se utiliza para reportar el estado de la solicitud de autorización. Puede incluir mensajes o logs que indiquen el resultado de la verificación de autorizaciones.

Al establecer las configuraciones de autorización para las acciones dentro de la tabla **result**, se mostrará un mensaje de falta de autorización según la operación solicitada. Estos mensajes pueden ser personalizados para ofrecer información más específica y útil al usuario.

Además, se puede identificar el nombre técnico del usuario que solicita la operación por medio del método estático **get_user_technical_name** de la clase **cl_abap_context_info**. Para especificar los usuarios con las autorizaciones pertinentes para realizar las operaciones dentro de la interfaz de usuario.

Ejemplo:

```
METHOD get_global_authorizations.
  DATA(lv_technical_name) = cl_abap_context_info=>get_user_technical_name( ).
  IF requested_authorizations-%create EQ if_abap_behv=>mk-on.
    IF lv_technical_name EQ 'User Name' .
      result-%create = if_abap_behv=>auth-allowed.
    ELSE.
      result-%create = if_abap_behv=>auth-unauthorized.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

1.2. Autorización Global – Actualización

De igual forma que para la operación de crear, es posible verificar las autorizaciones globales para la operación de actualización, permitiendo o denegando el acceso basado en la autorización definida para el usuario dentro del método get_global_authorizations. Para esto se utilizan las estructuras requested_authorizations para identificar las operaciones solicitadas para las modificación de los registros de la entidad raíz por una operación update realizada a través de un endpoint o al presionar un botón como Edit, y result para devolver las autorizaciones pertinentes.

Ejemplo:

```
DATA(lv_technical_name) = cl_abap_context_info=>get_user_technical_name( ).

IF requested_authorizations-%update EQ if_abap_behv=>mk-on or
     requested_authorizations-%action-Edit EQ if_abap_behv=>mk-on.
    IF lv_technical_name EQ 'User Name' .
      result-%update = if_abap_behv=>auth-allowed.
      result-%action-Edit = if_abap_behv=>auth-allowed.
    ELSE.
      result-%update = if_abap_behv=>auth-unauthorized.
      result-%action-Edit = if_abap_behv=>auth-unauthorized.
    ENDIF.
  ENDIF.
```

1.3. Autorización Global – Eliminación

De igual forma que para la operación de crear, es posible verificar las autorizaciones globales para la operación de eliminación, permitiendo o denegando el acceso basado en la autorización definida para el usuario dentro del método get_global_authorizations. Para esto se utilizan las estructuras requested_authorizations para identificar las operaciones solicitadas para las modificación de los registros de la entidad raíz por una operación delete realizada a través de un endpoint y result para devolver las autorizaciones pertinentes.

Ejemplo:

```
DATA(lv_technical_name) = cl_abap_context_info=>get_user_technical_name( ).

  IF requested_authorizations-%delete EQ if_abap_behv=>mk-on.
   IF lv_technical_name EQ 'User Name' .
     result-%delete = if_abap_behv=>auth-allowed.
   ELSE.
     result-%delete = if_abap_behv=>auth-unauthorized.
   ENDIF.
  ENDIF.
```

1.4. Autorizaciones de Instancia – Actualización

Las autorizaciones de instancia permiten controlar el acceso a las operaciones de actualización o eliminación sobre registros específicos dentro de los datos de una aplicación. Estas autorizaciones se basan en los valores específicos de los datos de cada registro y determinan si un usuario tiene permiso para realizar ciertas operaciones.

Implementación de Autorizaciones de instancia:

Las autorizaciones de instancia se aplican a registros existentes, permitiendo o denegando operaciones de actualización o eliminación en función de los valores de los datos.

La definición de las autorizaciones globales se definen en el Behavior Definition asociada a una entidad raíz por medio de la sentencia authorization master añadiendo la propiedad instance:

```
authorization master ( global, instance )
```

La implementación de la lógica de instancia de autorización se implementa en la clase Behavior Pool asociada al Behavior Definition en el método **get_instance_authorizations.**

Es importante agregar las propiedades de autorizaciones **global** e **instance**, que se utilizarán para controlar los niveles de acceso de los usuarios, antes de generar la clase del Definition Pool. Esto es necesario porque la clase declara los métodos pertinentes si previamente se especificaron por medio de la sentencia authorization master. En el caso contrario se debe realizar este proceso utilizando la ayuda del framework a través del botón  para generar la lógica de las autorizaciones globales. El método **get_instance_authorizations,** se declara de la siguiente forma:

```
METHODS get_instance_authorizations FOR INSTANCE
AUTHORIZATION
    IMPORTING keys REQUEST requested_authorizations FOR
root_entityname or alias RESULT result.
```

Condiciones de Autorización:

El método **get_instance_authorizations,** es fundamental para verificar las autorizaciones de instancia del usuario en una aplicación. A continuación, se describen los parámetros clave de este método y sus funciones:

- **keys**: Este parámetro es una tabla interna que se utiliza para especificar las claves técnicas de los registros que se están autorizando. Es decir, las claves que identifican de manera única a cada registro en la entidad que se está controlando.
- **requested_authorization**: Este es un parámetro es una estructura que indica qué tipo de autorización se está solicitando. Puede ser una operación de actualización (%update), eliminación (%delete), o cualquier otra operación permitida por la definición de comportamiento. La estructura dentro del método get_instance_authorizations posee los siguientes campos que identifican la operación solicitada:

- **%delete**: Identifica la operación para eliminar registros.
- **%update**: Identifica la operación para actualizar registros.
- **%action**: Este campo representa la acción específica las validaciones personalizadas entre otros aspectos definidos en el Behavior Definition.

- **result**: Este parámetro es una tabla interna que de salida se utiliza para almacenar los resultados de la autorización. Se debe llenar con la estructura de autorización que indica si se ha concedido o denegado el acceso a la operación solicitada. La tabla interna dentro del método get_instance_authorizations, posee los mismos campos que la estructura requested_authorizations con la diferencia que permite agregar otros campos para establecer la autorización de las operaciones utilizando la estructura de constantes **auth** de la interfaz **if_abap_behv**, con los valores **allowed** y **unauthorized**. Dichos campos son los siguientes:
  - **%tky:** Este campo representa la clave técnica del registro. Es una referencia única que identifica al registro dentro de la entidad.
  - **%is_draft:** Este campo indica si el registro está en estado borrador (draft). Es útil para determinar si el registro es una versión temporal que aún no ha sido confirmada.
  - **%key:** Este campo contiene la clave principal del registro. Es la clave que se utiliza para identificar de manera única al registro en la base de datos.
  - **%op:** Este campo especifica la operación que se está autorizando.
  - **%pky:** Este campo contiene la clave principal del registro en el contexto de la entidad. Es similar al campo %key, pero se utiliza específicamente para la entidad en cuestión.

  - **ComponentUUID:** Este campo contiene el identificador único UUID del componente que está siendo autorizado de la entidad raíz.

- **failed**: Este parámetro es una tabla interna de salida que se utiliza para indicar si hubo algún error durante el proceso de autorización. Si se produce un error, este parámetro puede ser utilizado para manejar la situación adecuadamente.

- **reported**: Este parámetro es una tabla interna que se utiliza para reportar cualquier mensaje o información adicional sobre el proceso de autorización.

La operación de creación no se maneja mediante autorizaciones de instancia, ya que un registro no existe en el momento de su creación. Estas se gestionan con autorizaciones globales.

El proceso comienza con la lectura de los datos de las entidades mediante la sentencia **read entities of**, para identificar los registros específicos a los que se les asignará la autorización pertinente. Luego, a través de una iteración, se asignan valores a la tabla **result** utilizando la clave técnica **%tky**, para permitir o restringir las operaciones solicitadas. Esto se hace utilizando la estructura **requested_authorizations** y asignando a los campos **%delete, %update** y **%action,** el valor de las constantes de autorización **auth** de la interfaz **if_abap_behv**, con los valores **allowed** y **unauthorized**.

Ejemplo:

```
METHOD get_instance_authorizations.
* Declaration of necessary variables
  DATA: lv_update_requested TYPE abap_bool,
        lv_update_granted   TYPE abap_bool.

* Read root entity entries updated
  READ ENTITIES OF root_entity_name IN LOCAL MODE
  ENTITY root_entity_alias
  ALL FIELDS WITH CORRESPONDING #( keys )
  RESULT DATA(lt_root_entity)
  FAILED failed.

* Identify current operation to be authorized
  lv_update_requested = COND #( WHEN requested_authorizations-%update =
if_abap_behv=>mk-on OR
```

```
                    requested_authorizations-%update = if_abap_behv=>mk-on
                  THEN abap_true
                  ELSE abap_false ).
  CHECK lv_update_requested EQ abap_true.

* Iterate through the root entity records
  DATA(lv_technical_name) = cl_abap_context_info=>get_user_technical_name( ).
  LOOP AT lt_root_entity INTO DATA(ls_root_entity).
   IF lv_technical_name EQ 'User Name' AND ls_root_entity-component EQ 'Value'.
    lv_update_granted = abap_true.
   ELSE.
    lv_update_granted = abap_false.
   ENDIF.

* Set authorizations to root entity records
   APPEND VALUE #( LET upd_auth = COND #(
                          WHEN lv_update_granted EQ abap_true
                           THEN if_abap_behv=>auth-allowed
                           ELSE if_abap_behv=>auth-unauthorized  )
                           IN
                           %tky = ls_root_entity-%tky
                           %update = upd_auth
                           %action-Edit = upd_auth ) TO result.
  ENDLOOP.
ENDMETHOD.
```

1.5. Autorizaciones de Instancia – Eliminación

De igual forma que para la operación de modificar, es posible verificar las autorizaciones de instancia para la operación de eliminación, permitiendo o denegando el acceso basado en la autorización definida para el usuario dentro del método get_instance_authorizations. Para esto se utilizan las estructuras requested_authorizations para identificar las operaciones solicitadas para la eliminación de los registros de la entidad raíz por una operación delete y se utiliza la tabla result para devolver las autorizaciones pertinentes.

Ejemplo:

```
METHOD get_instance_authorizations.
* Declaration of necessary variables
  DATA: lv_delete_requested TYPE abap_bool,
        lv_delete_granted   TYPE abap_bool.
```

```
* Read root entity entries updated
  READ ENTITIES OF root_entity_name IN LOCAL MODE
  ENTITY root_entity_alias
  ALL FIELDS WITH CORRESPONDING #( keys )
  RESULT DATA(lt_root_entity)
  FAILED failed.

* Identify current operation to be authorized
  lv_delete_requested = COND #( WHEN requested_authorizations-%delete =
if_abap_behv=>mk-on
                        THEN abap_true
                        ELSE abap_false ).
  CHECK lv_delete_requested EQ abap_true.

* Iterate through the root entity records
  DATA(lv_technical_name) = cl_abap_context_info=>get_user_technical_name( ).
  LOOP AT lt_root_entity INTO DATA(ls_root_entity).
   IF lv_technical_name EQ 'User Name' AND ls_root_entity-component EQ 'Value'.
     lv_delete_granted = abap_true.
   ELSE.
     lv_delete_granted = abap_false.
   ENDIF.

* Set authorizations to root entity records
    APPEND VALUE #( LET del_auth = COND #(
                              WHEN lv_delete_granted EQ abap_true
                               THEN if_abap_behv=>auth-allowed
                               ELSE if_abap_behv=>auth-unauthorized  )
                               IN
                               %tky = ls_root_entity-%tky
                               %delete = del_auth ) TO result.
  ENDLOOP.
ENDMETHOD.
```

Es posible ampliar el método **get_instance_authorizations** para agregar autorizaciones a registros específicos para todas las operaciones, ya sean de modificación, eliminación u otras permitidas por el método.