Automatically Generate and Assign Authorizations in SAP BW/4HANA or BPC Embedded

Table of Contents

Access Only Authorized Data

Business Analytics applications nowadays expose huge amounts of data to large groups of users. In either reporting, analytical tasks or business planning scenarios we would like to give access only to the data users are supposed to see. Authorized data is normally defined in a set of business rules. These rules may be complex and involve dynamic organizational structure that evolves over time.

It would be nice to have proper authorizations applied in BI reports, Business Planning or transactional applications, where business complexity is realized via BW Analysis Authorizations. What would be especially super nice to have is that these authorizations are generated and assigned to users automatically!

Business Planning applications on SAP BPC embedded (S/4HANA) can also have a huge win with automated Analysis Authorizations assignments. In multi-user multi-departmental planning scenarios users may potentially lock each other. In cases where planning access is restricted to the department user is responsible for, other users will not be locked, and can perform business planning activities for other departments at the same time.

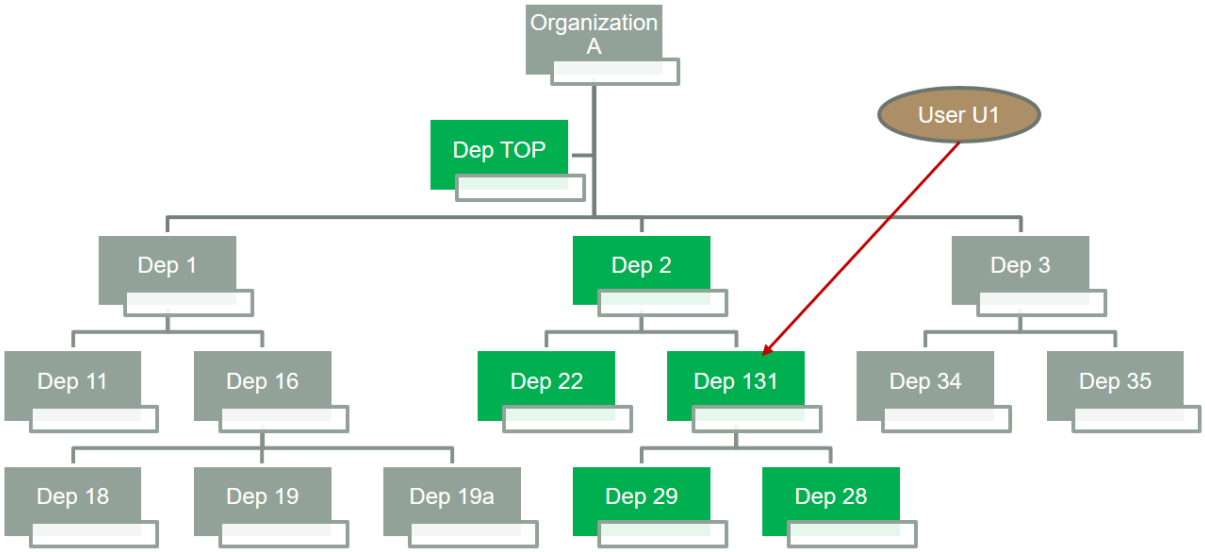
In the post below I will go through steps needed to automate Analysis Authorizations generation, and review how you can save on redundant manual processes required in maintaining complex organizational structures, user and authorization assignments.

Authorizations Logic and Data Sources

Let us imagine we have 2 datasets maintained in a separate application (e.g. LDAP) for users and authorizations. First dataset specifies Organization's structure by defining child-parent relationships

between units and departments. The second one defines user assignments to one or several nodes in the Organizational structure (departments).

Each user assigned to a department can see all levels below, departments reporting to the same parent, and summarized data for all parental departments. The following chart illustrates this scenario (green nodes from the Org Structure are visible to User U1):



*Rules defining data access by user can be summarized in the **Business Rules 1,2,3**:*

1. *All nodes below user's department (BW AA Type 1)*
2. *All neighbors reporting to the same parent, but not below them (BW AA Type 4, +1)*
3. *Parent of parent (recursively) at a total level (BW AA Type 0)*

Business Content for Analysis Authorizations

SAP has provided standard Business Content in BW/4HANA for generating users and Analysis Authorizations. We can simply install this content, populate relevant ADSOs with appropriate details and generate users and authorizations using program: RSEC_GENERATE_AUTHORIZATIONS.

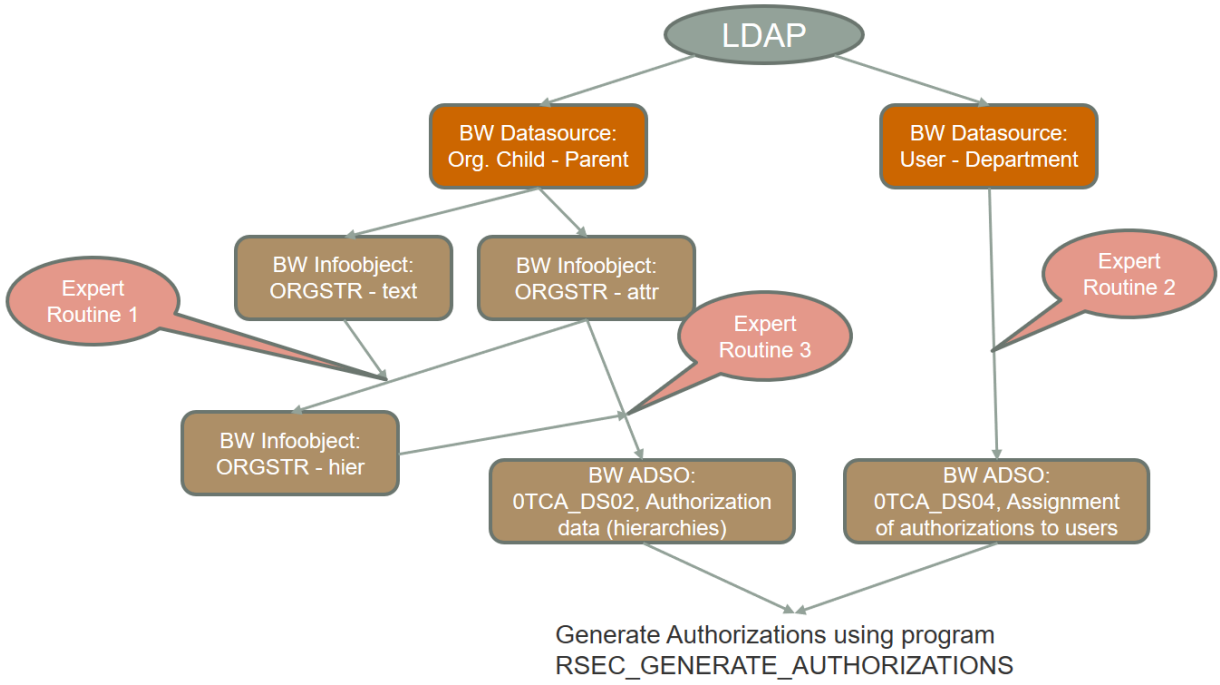
For more details on how to use Business Content see SAP Help:
https://help.sap.com/doc/saphelp_qim100/1.0/en-US/46/8bbd2738fc429ee10000000a1553f7/frameset.htm

Data Flow in BW

First step in delivering on these requirements will be to generate a hierarchy based on the Child-Parent relationships. This hierarchy may be refreshed daily or more frequently. It can be used both for

reports/dashboards navigations and for restricting user access.

If we create an authorization relevant infoobject ORGSTR with master data, texts and hierarchies the following data flow can be used for automating Analysis Authorizations generation:



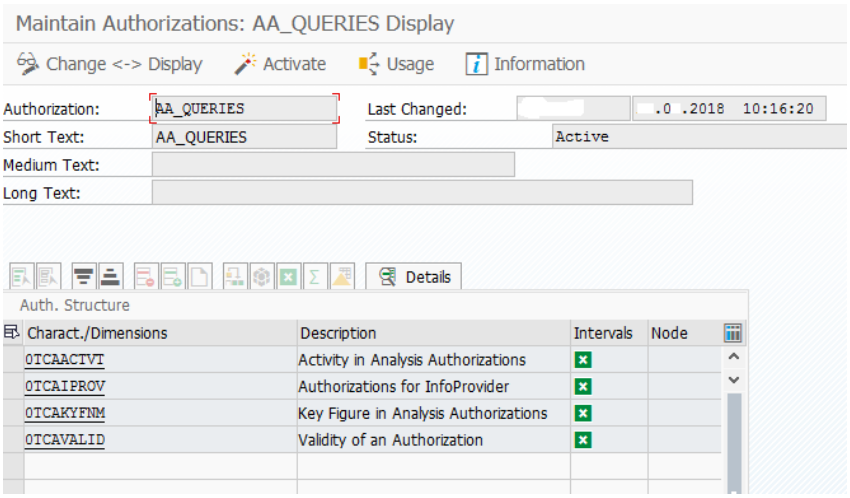
Logic defined in the expert routines allows generating hierarchy, users, relevant authorizations and assignments. The following routines are key in the data flow:

- *Sample Expert Routine 1 for hierarchy generation is given in Appendix 1.*
- *Sample Expert Routine 2 for Authorizations assignments to users is given in Appendix 2.*
- *Expert routine 3 generates authorizations based on business rules 1,2,3*

User Assignments to Departments

If we have users to department mapping coming from an external source like LDAP we can assign proper Analysis Authorizations using naming conventions. These assignments may change frequently, and relevant authorizations will be added or removed automatically.

Technically, in Expert Routine 2 each user must get an authorization assigned for each business rule and department (AA1D131, AA2D131, AA3D131) they are responsible for. Additionally, a standard BW queries user authorization (AA_QUERIES) must be assigned to users:



If user U1 is assigned to department 131 he has access to the part of the Organizational structure that is defined by the Business Rules 1,2,3. This means he can see all structures reporting to Department 131, departmental totals reporting to the same parent, and totals for all parents. The other hierarchy nodes are simply not visible to user U1.

Final Words

Analysis Authorizations is one of the useful and powerful features in SAP BW. Generating complex analysis authorizations can be fully automated. External systems can be used for maintaining organizational structure changes. The rest can be done by scripts that would ensure data visibility by user and department follows corporate policies. This approach is especially practical in large organizations with regular structure updates, and in cases of temporary user assignments to departments, during job changes, holidays, replacements, etc.

The automation logic is beneficial both in a standalone SAP BW/4HANA data warehouse, as well as in SAP BPC embedded applications on S/4HANA, e.g. for Cost Center planning or Revenue planning scenarios where departmental data access restrictions are crucial.

About the Author

Sergei Peleshuk has over 15 years of experience implementing BI technologies for global clients in retail,

distribution, fast-moving consumer goods (FMCG), insurance, and energy industries. He has helped clients to design robust BI reporting and planning capabilities, leading them through all project phases: from analysis of requirements to building BI roadmaps, technical architecture, and efficient BI teams. Sergei is an expert in SAP Business Warehouse (SAP BW), SAP HANA, BPC, BusinessObjects, SAP Analytics Cloud, and SAP Lumira. You may contact Sergei at peleshuk@biportal.org

Appendix 1 – Expert Routine 1 – Generate Org hierarchy

```
* Generate Hierarchy from attribute Parent
* T1 ------------------
DATA: rp1 TYPE _ty_s_TG_1.
rp1-H_HIENM = 'ORGA'.
rp1-H_STARTLEV = '1'.
INSERT rp1 INTO TABLE RESULT_PACKAGE_1.

* T3 and T4 ------------------ Prepare Hierarchy Text
DATA: rp4 TYPE _ty_s_TG_4.
DATA rp6 TYPE _ty_s_TG_6.
TYPES:
 BEGIN OF ty_aht,

   NODENAME TYPE RSSHNODENAME,
   TXTSH TYPE RSTXTSH,
   TXTMD TYPE RSTXTMD,

 END OF ty_aht.

FIELD-SYMBOLS: <RESULT_FIELDS> TYPE _ty_s_TG_3.
DATA hid TYPE RSHIEID.

* Retrieve Hierarchy for ORGA
SELECT SINGLE HIEID FROM RSHIEDIR INTO hid
WHERE IOBJNM = 'ORGSTR' AND HIENM = 'ORGA'.

*Retrieve texts
DATA aht TYPE HASHED TABLE OF ty_aht WITH UNIQUE KEY NODENAME.
FIELD-SYMBOLS: <aht> TYPE ty_aht.

SELECT /BIC/ORGSTR AS NODENAME, TXTSH, TXTMD
FROM /BIC/TORGSTR
INTO CORRESPONDING FIELDS OF TABLE @aht.

* T3 ------------------ Hierarchy Structure
TYPES:
 BEGIN OF ty_ah,
   NODEID TYPE /BIC/OIORGSTR, "RSHIENODID,
   PARENTID TYPE /BIC/OIORGSTR, "RSPARENT,
   NEXTID TYPE /BIC/OIORGSTR,
   level      TYPE /BIC/OIORGLEV,
 END OF ty_ah.

DATA ah TYPE STANDARD TABLE OF ty_ah.
FIELD-SYMBOLS: <ah> TYPE ty_ah.
DATA ah3 TYPE ty_ah.

SELECT /BIC/ORGSTR AS NODEID,
     /BIC/ORGPARENT AS PARENTID,
     /BIC/ORGNEXT AS NEXTID,
```

```
    /BIC/ORGLEV AS level
FROM /BIC/PORGSTR
INTO CORRESPONDING FIELDS OF TABLE @ah
WHERE OBJVERS = 'A'
AND /BIC/ORGPARENT <> ' '.

DATA: nid TYPE N LENGTH 8.
DATA: rp TYPE _ty_s_TG_3.
FIELD-SYMBOLS: <rp> TYPE _ty_s_TG_3.
DATA a TYPE C LENGTH 32.

* Insert Top node
 rp-H_IOBJNM = '0HIER_NODE'. rp-H_PARENTID = ' '.
 rp-H_NODEID = '1'.
 rp-H_HIERNODE = rp-/BIC/ORGSTR = 'TOP'.
 INSERT rp INTO TABLE RESULT_PACKAGE_3.

* Generate lines for lowest level
 nid = 2.  rp-H_IOBJNM = 'ORGSTR'.
LOOP AT SOURCE_PACKAGE ASSIGNING <SOURCE_FIELDS>
 WHERE /BIC/ORGSTR <> ' '.
* Check if this is the lowest level
* (Is there a node with the current node as parent?)
 READ TABLE ah ASSIGNING <ah>
 WITH KEY PARENTID = <SOURCE_FIELDS>-/BIC/ORGSTR.
 IF sy-subrc <> 0.
  rp-H_NODEID = nid.
  nid = nid + 1.
  a = rp-/BIC/ORGSTR = <SOURCE_FIELDS>-/BIC/ORGSTR.
  SHIFT a RIGHT BY 27 PLACES. TRANSLATE a USING ' 0'.
  rp-H_HIERNODE = a.
  INSERT rp INTO TABLE RESULT_PACKAGE_3.
 ENDIF.
ENDLOOP.

* Assign parent node ids
 rp-H_IOBJNM = '0HIER_NODE'.  rp-H_PARENTID = ' '.

LOOP AT RESULT_PACKAGE_3 ASSIGNING <RESULT_FIELDS> WHERE /BIC/ORGSTR <>
'TOP'.
 READ TABLE ah ASSIGNING <ah>
 WITH KEY NODEID = <RESULT_FIELDS>-/BIC/ORGSTR.
 IF sy-subrc = 0 AND <ah>-PARENTID <> ' '.
* Check if Parent already exists in the new hierarchy
     READ TABLE RESULT_PACKAGE_3 ASSIGNING <rp>
     WITH KEY /BIC/ORGSTR = <ah>-PARENTID
           H_IOBJNM = '0HIER_NODE'.
     IF sy-subrc = 0.
      <RESULT_FIELDS>-H_PARENTID = <rp>-H_NODEID.
     ELSE.
* Add upper node to Result Package
```

```
        rp-H_NODEID = nid.
        nid = nid + 1.
        a = rp-/BIC/ORGSTR = <ah>-PARENTID.
        SHIFT a RIGHT BY 27 PLACES. TRANSLATE a USING ' 0'.
        rp-H_HIERNODE = a.
        INSERT rp INTO TABLE RESULT_PACKAGE_3.
        <RESULT_FIELDS>-H_PARENTID = rp-H_NODEID.
      ENDIF.
 ELSE.
* If no parents assígn to Top node
   <RESULT_FIELDS>-H_PARENTID = '1'.
 ENDIF.
ENDLOOP.

SORT RESULT_PACKAGE_3 BY H_NODEID /BIC/ORGSTR ASCENDING.
DELETE ADJACENT DUPLICATES FROM RESULT_PACKAGE_3 COMPARING H_NODEID.

* T4 ---------------------------- Populate Hierarchy text
 rp4-LANGU = 'EN'.
* Insert Text for the Top node
   rp4-H_HIERNODE = 'TOP'.
   rp4-TXTSH = rp4-TXTMD = rp4-TXTLG = 'TOP'.
   INSERT rp4 INTO TABLE RESULT_PACKAGE_4.

LOOP AT RESULT_PACKAGE_3 ASSIGNING <RESULT_FIELDS> WHERE H_IOBJNM =
'0HIER_NODE'.
* Add English text
 READ TABLE aht ASSIGNING <aht>
 WITH TABLE KEY NODENAME = <RESULT_FIELDS>-/BIC/ORGSTR.
 IF sy-subrc = 0.
   rp4-H_HIERNODE = <RESULT_FIELDS>-H_HIERNODE.
   rp4-TXTSH = <aht>-TXTSH.
   rp4-TXTMD = rp4-TXTLG = <aht>-TXTMD.
   INSERT rp4 INTO TABLE RESULT_PACKAGE_4.
 ENDIF.
ENDLOOP.

SORT RESULT_PACKAGE_4 BY H_HIERNODE ASCENDING.
*$*$ end of routine - insert your code only before this line       *-*
 ENDMETHOD.                "expert_routine
```

Appendix 2 – Expert Routine 2 – Authorizations by User

```
 METHOD expert_routine.
*=== Segments ===
   FIELD-SYMBOLS:
     <SOURCE_FIELDS>   TYPE _ty_s_SC_1.
   DATA:
     RESULT_FIELDS     TYPE _ty_s_TG_1.
```

```
*$*$ begin of routine - insert your code only below this line       *-*
... "insert your code here
* Pick all relevant Authorizations by user

* Auth Roles table
TYPES:
 BEGIN OF ty_ar,
   TCTAUTH TYPE C LENGTH 12,
 END OF ty_ar.
DATA ar TYPE STANDARD TABLE OF ty_ar
      WITH NON-UNIQUE DEFAULT KEY.
FIELD-SYMBOLS <ar> TYPE ty_ar.

SELECT TCTAUTH FROM /BI0/ATCA_AD022
INTO CORRESPONDING FIELDS OF TABLE ar.

* BW users table
TYPES:
 BEGIN OF ty_usr,
   BNAME TYPE C LENGTH 12,
 END OF ty_usr.
DATA usr TYPE SORTED TABLE OF ty_usr
      WITH NON-UNIQUE KEY BNAME.
FIELD-SYMBOLS <usr> TYPE ty_usr.

SELECT BNAME FROM USR01
INTO CORRESPONDING FIELDS OF TABLE usr.

RESULT_FIELDS-TCTADTO = '99991231'.
RESULT_FIELDS-TCTADFROM = '19000101'.

LOOP AT SOURCE_PACKAGE ASSIGNING <SOURCE_FIELDS>.
 RESULT_FIELDS-TCTUSERNM = <SOURCE_FIELDS>-BENSCHL(12).
 TRANSLATE RESULT_FIELDS-TCTUSERNM TO UPPER CASE.

 READ TABLE usr TRANSPORTING NO FIELDS
 WITH TABLE KEY
 BNAME = RESULT_FIELDS-TCTUSERNM.
* If BW user in the system generate authorizations
 IF sy-subrc = 0.
   LOOP AT ar ASSIGNING <ar> WHERE TCTAUTH+7(5) =
   <SOURCE_FIELDS>-BAUMSCHL.
     RESULT_FIELDS-TCTAUTH = <ar>-TCTAUTH.
     INSERT RESULT_FIELDS INTO TABLE RESULT_PACKAGE.
   ENDLOOP.
* Add authorization for query access
   RESULT_FIELDS-TCTAUTH = 'AA_QUERIES'.
   INSERT RESULT_FIELDS INTO TABLE RESULT_PACKAGE.
 ENDIF.
ENDLOOP.
```