BOPF Authorization Checks

SAP AG, 2012

Disclaimer

This presentation outlines our general product direction and should not be relied on in making a purchase decision. This presentation is not subject to your license agreement or any other agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or to develop or release any functionality mentioned in this presentation. This presentation and SAP's strategy and possible future developments are subject to change and may be changed by SAP at any time for any reason without notice. This document is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP assumes no responsibility for errors or omissions in this document, except if such damages were caused by SAP intentionally or grossly negligent.

Agenda

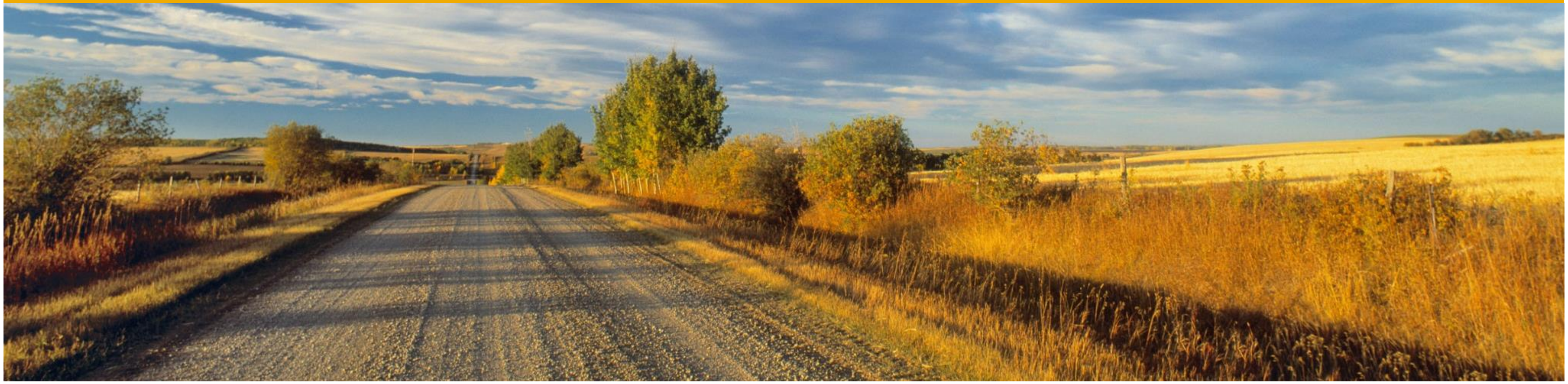
Introduction

Creating Authorization Objects and Fields

Assigning Authorization Objects and Mapping Fields

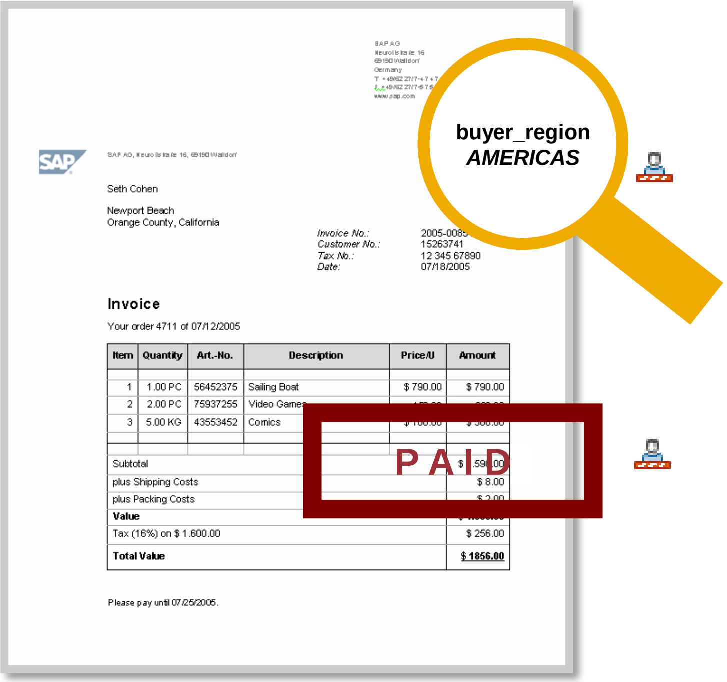
Role Configuration Examples

Runtime



Introduction

Introduction


buyer_region
AMERICAS
PAID

SAP AG
Neurottstraße 16
69190 Walldorf
Germany
T +49/62 27/7-4 7 4 7
F +49/62 27/7-5 7 5
www.sap.com

SAP AG, Neurottstraße 16, 69190 Walldorf

Seth Cohen

Newport Beach
Orange County, California

Invoice No.:	2005-0085
Customer No.:	15263741
Tax No.:	12 345 67890
Date:	07/18/2005

Invoice

Your order 4711 of 07/12/2005

Item	Quantity	Art.-No.	Description	Price/U	Amount
1	1.00 PC	56452375	Sailing Boat	$ 790.00	$ 790.00
2	2.00 PC	75937255	Video Games	[illegible]	[illegible]
3	5.00 KG	43553452	Comics	[illegible]	[illegible]
Subtotal					$ 1.590.00
plus Shipping Costs					$ 8.00
plus Packing Costs					$ 2.00
Value					[illegible]
Tax (16%) on $ 1.600.00					$ 256.00
Total Value					**$ 1856.00**

Please pay until 07/25/2005.

Motivation

- Applications require an authorization concept for their data and the *operations* on their data, so that *display* and *update* activities are allowed for authorized persons only.
- In this example, invoices with the `buyer_region AMERICAS` must be visible only for authorized persons. Moreover, invoices must be changed to status `PAID` only by authorized persons (action `invoice_paid`).

Introduction

BOPF offers a generic authorization concept for applications built on top of business objects, so that **each *display* and *update* activity can be protected by an authorization check**.

Therefor, the application **only has to**…

- **create an authorization object**
- **assign the authorization object to** the appropriate business object **node(s)**
- **map the authorization fields to** the appropriate **node data fields.**

The application does not have to implement specific check coding as long as the application accepts the generic authorization implementation.

The generic authorization concept **is based on the well-known concept of authorization objects and the `authority-check` statement.** Regarding the authorization objects, a BOPF-specific field *pattern* is required. Details will follow…

Basically, all BOPF service requests are authorization-relevant (`RETRIEVE, RETRIEVE_BY_ASSOCIATION, CONVERT_ALTERN_KEY, MODIFY, DO_ACTION, QUERY,` …) and are therefore checked for authorization by the generic authorization implementation.

Introduction

The generic authorization concept differentiates between *static* and *instance-based* authorization checks…

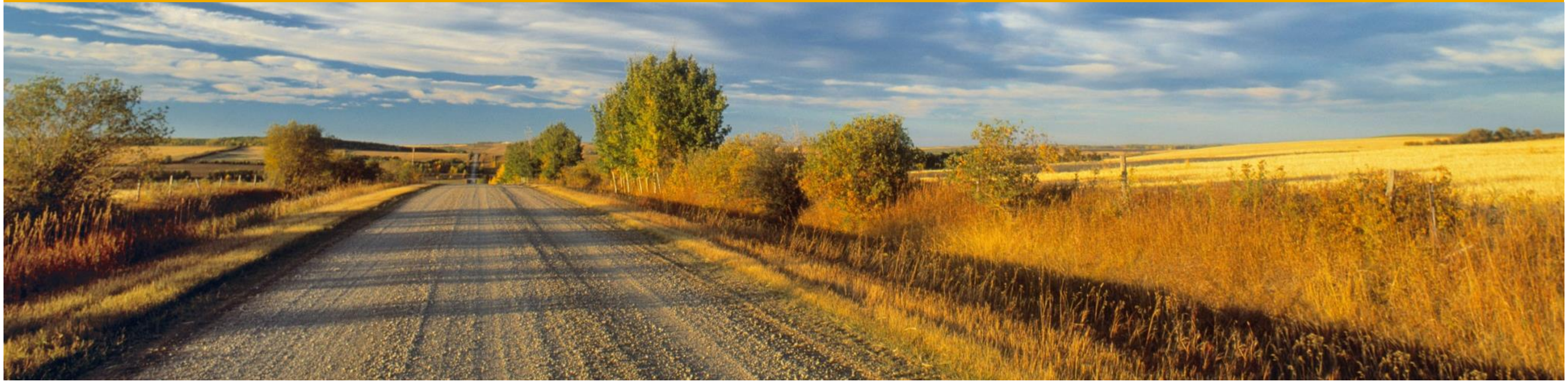
- **Static check**: Checks if the user has permission to perform a sepcific activity (e.g. `DISPLAY`)
- **Instance-based check**: Evaluates the node data and checks if the user has permission to display or change a data row where an authorization-relevant field has a concrete value

Therefore our first example “invoices with the `buyer_region AMERICAS` must be visible only for authorized persons” is handled by an instance-based check, whereas the second example “invoices must be changed to status `PAID` only by authorized persons” is handled by a static check.

Introduction

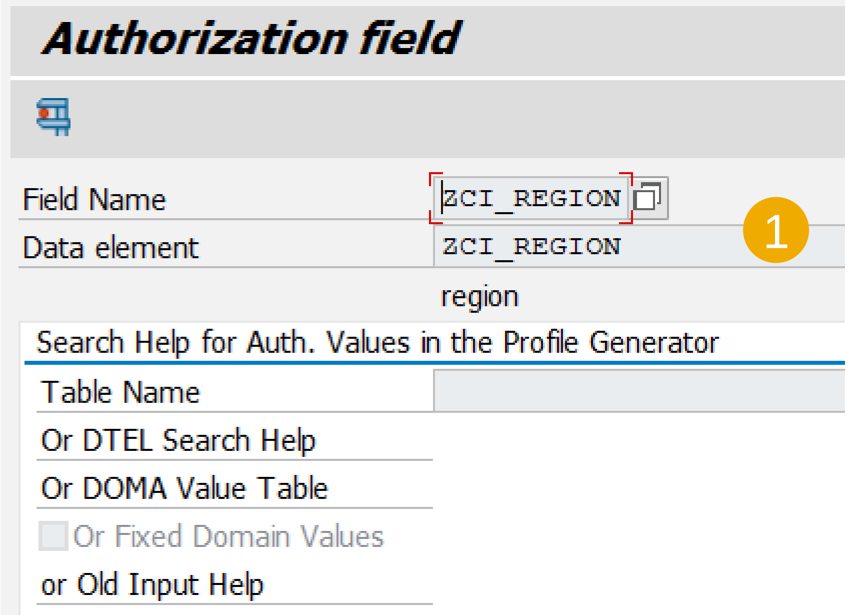
The static check utilizes the first field ACTVT and the second field BO_SERVICE of an authorization object. The instance-based check utilizes all authorization fields, especially the application-specific ones like BUYER_REGION.

Even if authorization objects and fields are assigned to a concrete node, at runtime the static and instance-based checks will be propagated along the composition tree. **E.g. authorization checks configured to ROOT node will be propagated to ITEM node.**



Creating Authorization Objects and Fields

Creating Authorization Fields



Authorization field

Field Name	ZCI_REGION
Data element	ZCI_REGION
	region

Search Help for Auth. Values in the Profile Generator

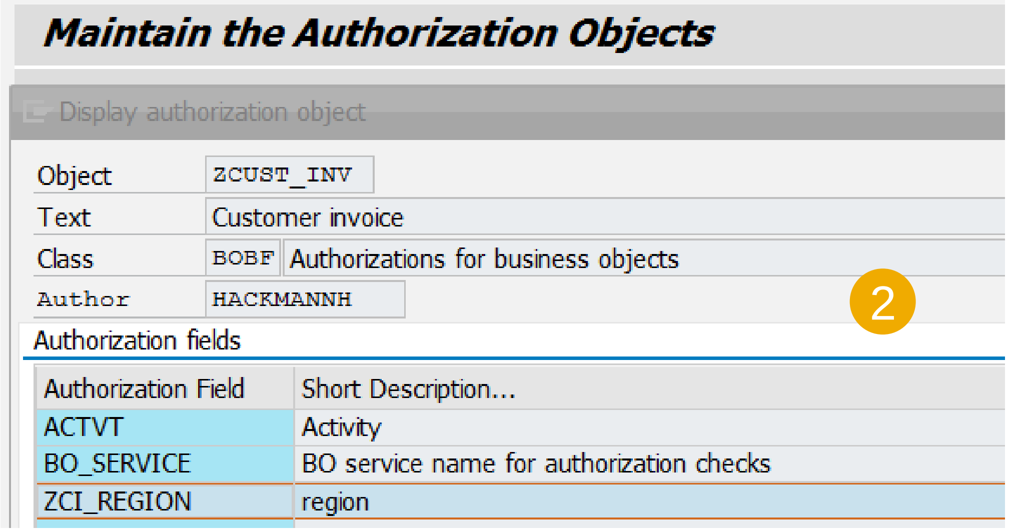
Table Name	
Or DTEL Search Help	
Or DOMA Value Table	
□ Or Fixed Domain Values	
or Old Input Help	

Use transaction SU20 to display existing authorization fields and to create new authorization fields.

You do not always need to create new authorization fields since a huge variety of authorization fields already exists and may be re-used in the new authorization object.

Hint: The “generic” authorization fields ACTVT and BO_SERVICE are already delivered by SAP.

Creation of Authorization Objects



Maintain the Authorization Objects

Display authorization object

Object	ZCUST_INV	
Text	Customer invoice	
Class	BOBF	Authorizations for business objects
Author	HACKMANNH	

Authorization fields

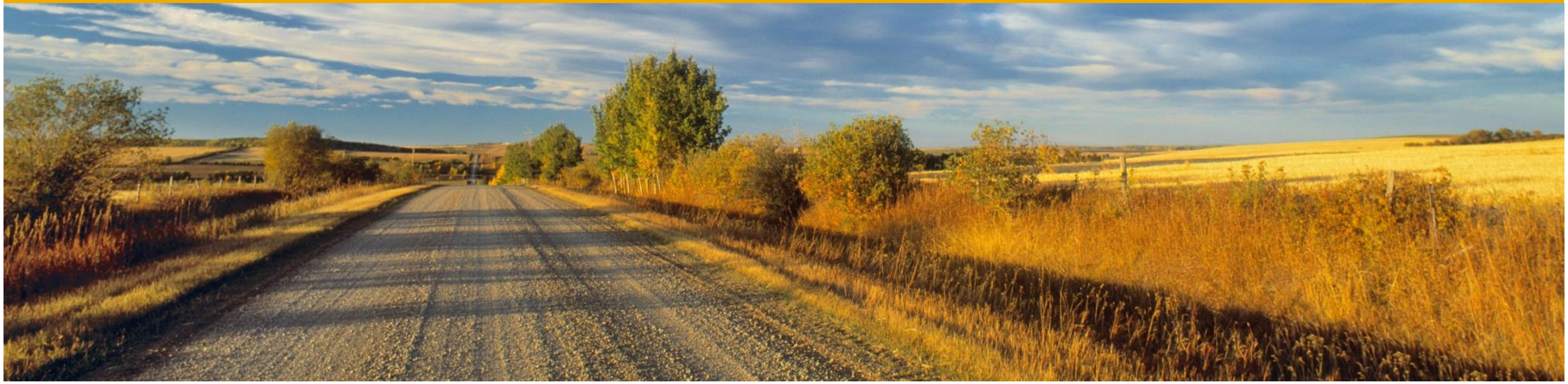
Authorization Field	Short Description...
ACTVT	Activity
BO_SERVICE	BO service name for authorization checks
ZCI_REGION	region

Use transaction SU21 to create new authorization objects.

To configure an authorization for a business object, you need at least one authorization object.

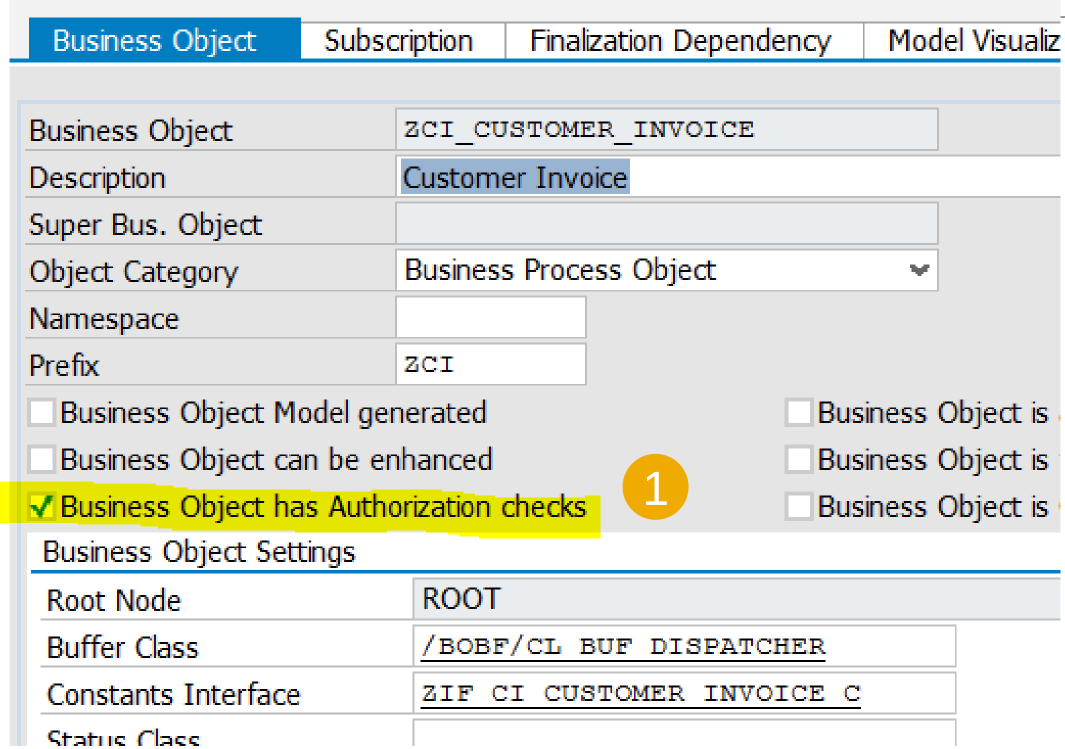
Even you can assign several authorization objects to one business object (many authorizations objects may be assigned to one node), it is good practice to start with one authorization object for the whole business object. In the lifecycle of the business object, you might have to introduce new authorization objects since the existing authorization object cannot be changed any longer.

Hint: The authorization object must consist of the fields ACTVT at the first position and BO_SERVICE at the second one. As of the third position, application-specific fields can be configured.



Assigning Authorization Objects and Mapping Fields

Authorization Configuration – BO Level



Business Object | Subscription | Finalization Dependency | Model Visualiz

Business Object	ZCI_CUSTOMER_INVOICE
Description	Customer Invoice
Super Bus. Object	
Object Category	Business Process Object
Namespace	
Prefix	ZCI

□ Business Object Model generated

□ Business Object can be enhanced

☑ Business Object has Authorization checks

□ Business Object is

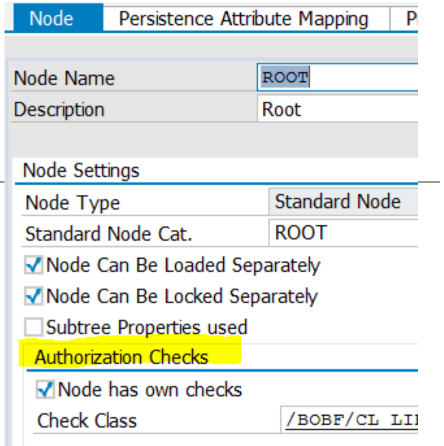
□ Business Object is

□ Business Object is

1

Business Object Settings

Root Node	ROOT
Buffer Class	/BOBF/CL_BUF_DISPATCHER
Constants Interface	ZIF_CI_CUSTOMER_INVOICE_C
Status Class	



Node | Persistence Attribute Mapping | P

Node Name	ROOT
Description	Root

Node Settings

Node Type	Standard Node
Standard Node Cat.	ROOT

☑ Node Can Be Loaded Separately

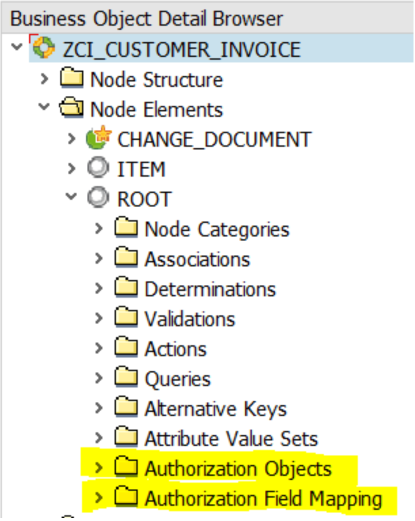
☑ Node Can Be Locked Separately

□ Subtree Properties used

Authorization Checks

☑ Node has own checks

Check Class	/BOBF/CL_LI



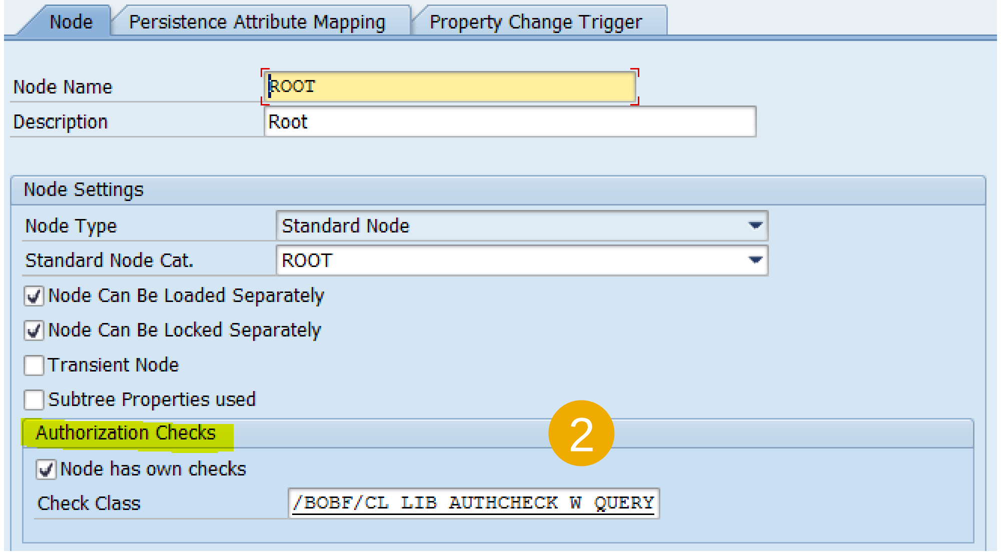
Business Object Detail Browser

- ZCI_CUSTOMER_INVOICE
  - Node Structure
  - Node Elements
    - CHANGE_DOCUMENT
    - ITEM
    - ROOT
      - Node Categories
      - Associations
      - Determinations
      - Validations
      - Actions
      - Queries
      - Alternative Keys
      - Attribute Value Sets
      - Authorization Objects
      - Authorization Field Mapping

First of all, the “main” authorization flag must be marked in the business object settings (transaction `BOBF`).

Hint: If this flag is not marked, all authorization configuration sections on node level are set to invisible.

Authorization Configuration – Node Level



Node Name: ROOT
Description: Root

Node Settings

Node Type: Standard Node
Standard Node Cat.: ROOT

☑ Node Can Be Loaded Separately
☑ Node Can Be Locked Separately
□ Transient Node
□ Subtree Properties used

Authorization Checks

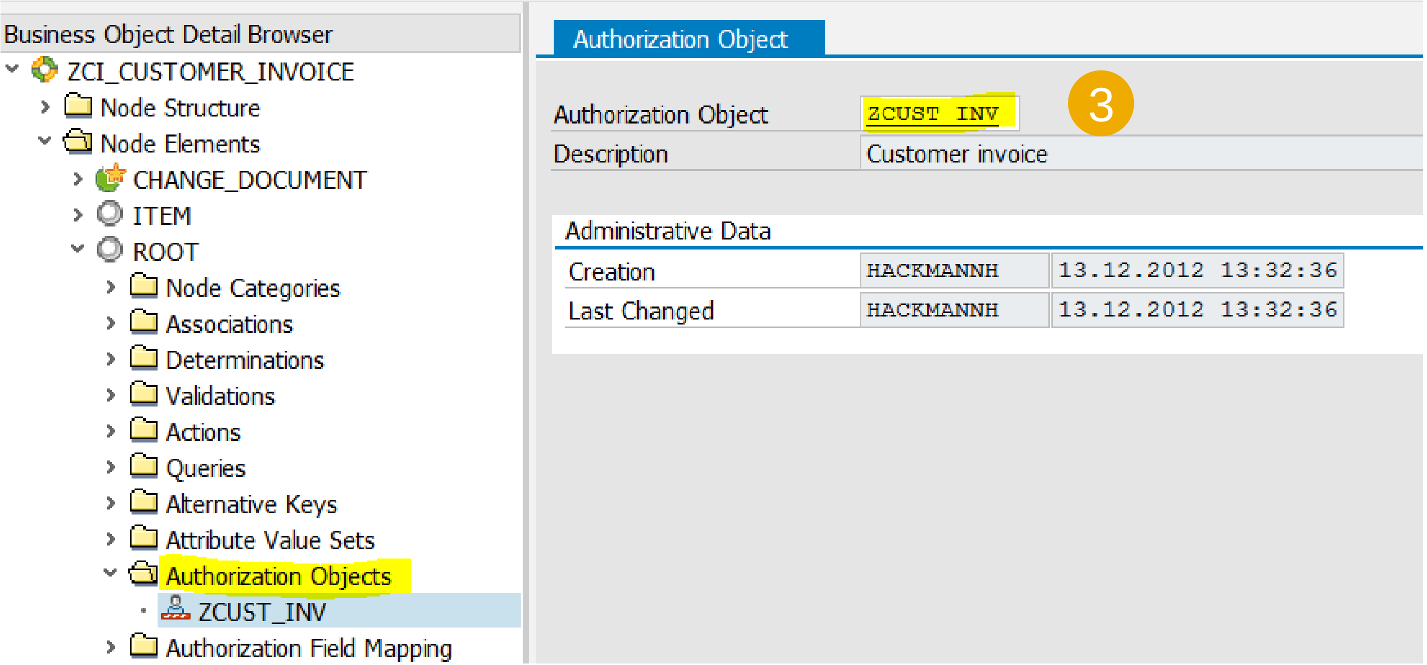
☑ Node has own checks
Check Class: /BOBF/CL_LIB_AUTHCHECK_W_QUERY

You must decide on which node(s) the authorization are performed. In our example, the ROOT node is suitable. For many other business objects, the ROOT node may also be the first choice.

The flag “Node has own checks” must be marked and thus, the library class `/BOBF/CL_LIB_AUTHCHECK_W_QUERY` is pre-filled as check class.

In most cases, the library class must meet the application requirements. But an application may also decide to implement a different behavior.

Authorization Configuration – Node Level



Business Object Detail Browser

- ZCI_CUSTOMER_INVOICE
  - Node Structure
  - Node Elements
    - CHANGE_DOCUMENT
    - ITEM
    - ROOT
      - Node Categories
      - Associations
      - Determinations
      - Validations
      - Actions
      - Queries
      - Alternative Keys
      - Attribute Value Sets
      - Authorization Objects
        - ZCUST_INV
      - Authorization Field Mapping

Authorization Object

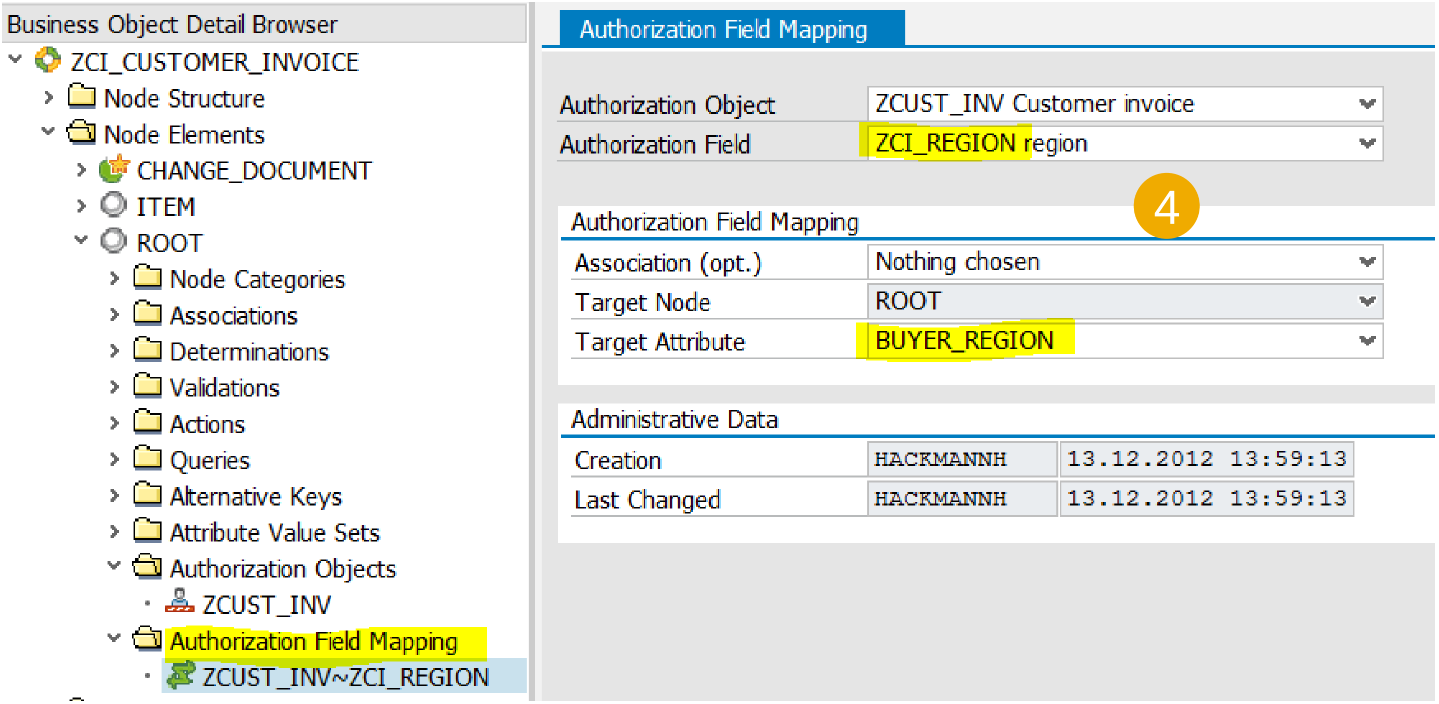
Authorization Object	ZCUST_INV
Description	Customer invoice

Administrative Data

Creation	HACKMANNH	13.12.2012 13:32:36
Last Changed	HACKMANNH	13.12.2012 13:32:36

First of all, you assign the authorization object `ZCUST_INV` to the ROOT node.

Authorization Configuration – Node Level



Business Object Detail Browser

- ZCI_CUSTOMER_INVOICE
  - Node Structure
  - Node Elements
    - CHANGE_DOCUMENT
    - ITEM
    - ROOT
      - Node Categories
      - Associations
      - Determinations
      - Validations
      - Actions
      - Queries
      - Alternative Keys
      - Attribute Value Sets
      - Authorization Objects
        - ZCUST_INV
      - Authorization Field Mapping
        - ZCUST_INV~ZCI_REGION

Authorization Field Mapping

Authorization Object	ZCUST_INV Customer invoice
Authorization Field	ZCI_REGION region

Authorization Field Mapping

Association (opt.)	Nothing chosen
Target Node	ROOT
Target Attribute	BUYER_REGION

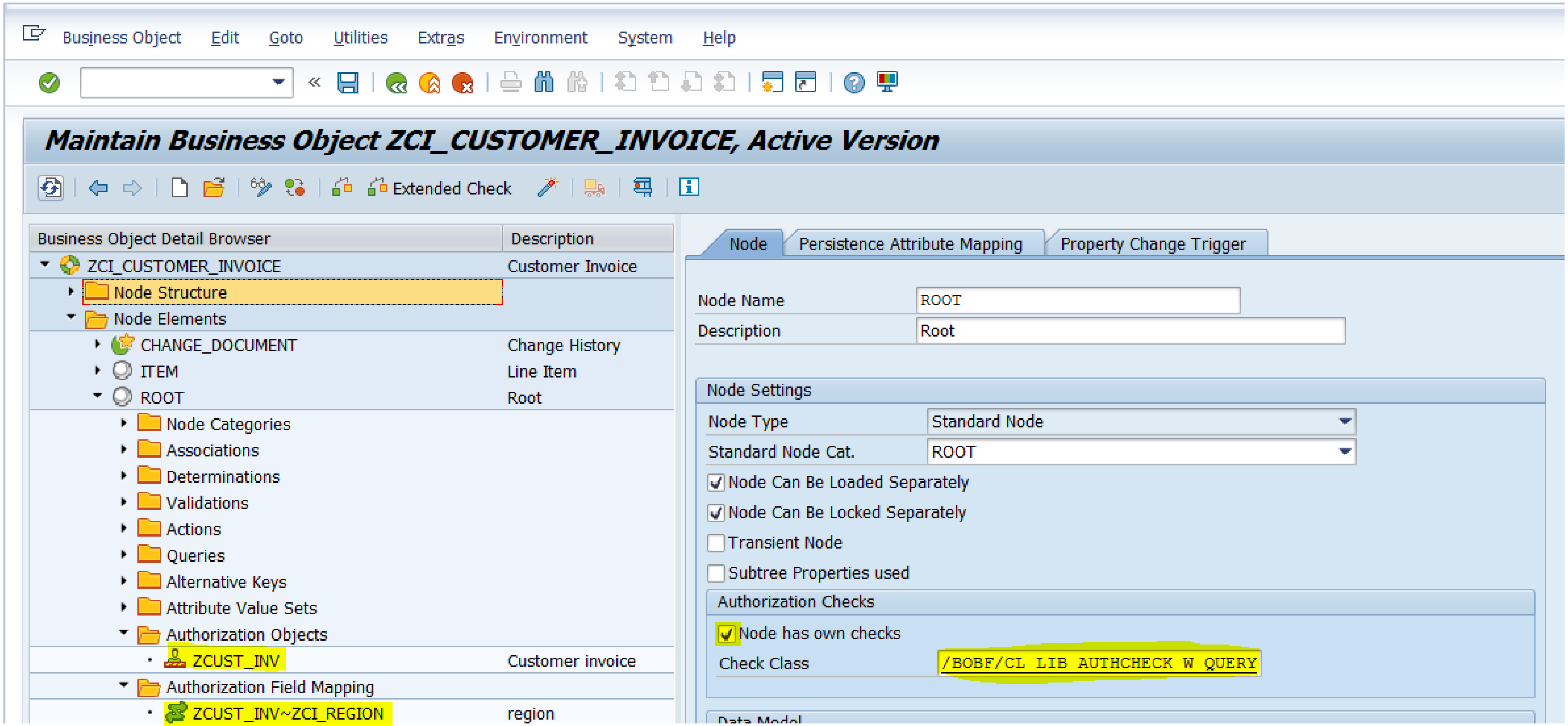
Administrative Data

Creation	HACKMANNH	13.12.2012 13:59:13
Last Changed	HACKMANNH	13.12.2012 13:59:13

Secondly, you assign the authorization field `ZCI_REGION` to the data field `BUYER_REGION` of the ROOT node.

Authorization Configuration – Node Level

Overview - the complete authorization configuration for the ROOT node…



Maintain Business Object ZCI_CUSTOMER_INVOICE, Active Version

Business Object Detail Browser	Description
ZCI_CUSTOMER_INVOICE	Customer Invoice
Node Structure	
Node Elements	
CHANGE_DOCUMENT	Change History
ITEM	Line Item
ROOT	Root
Node Categories	
Associations	
Determinations	
Validations	
Actions	
Queries	
Alternative Keys	
Attribute Value Sets	
Authorization Objects	
ZCUST_INV	Customer invoice
Authorization Field Mapping	
ZCUST_INV~ZCI_REGION	region

Node Name: ROOT

Description: Root

Node Settings

Node Type: Standard Node

Standard Node Cat.: ROOT

☑ Node Can Be Loaded Separately

☑ Node Can Be Locked Separately

☐ Transient Node

☐ Subtree Properties used

Authorization Checks

☑ Node has own checks

Check Class: /BOBF/CL_LIB_AUTHCHECK_W_QUERY

Data Model

Implementing application specific customizing

By entering an application specific class that inherits from /BOBF/CL_LIB_AUTHCHECK_W_QUERY, it is possible to redefine the behavior in the following way:

- **Redefine static fields activity and bo_service**
  example: For querying node data, check activity display instead of query
- **Suppress authority propagation**
  example: for retrieving items, check authority only on item node instead of item and root node
- **Send application specific messages in case authorization is denied**

The methods of the interfaces /BOBF/IF_LIB_AUTH_CUSTOM_GEN and /BOBF/IF_LIB_AUTH_CUSTOMIZER are meant to be redefined by applications for that purpose

Overwriting the standard authorization concept

In rare cases, it is necessary to implement a completely different authorization concept, e.g. if Access Control Lists should are used instead of standard authority objects. To achieve this, an application specific class can be defined that inherits from the abstract superclass /BOBF/CL_FRW_AUTHORITY_CHECK. Then, all methods have to be implemented.

Further assistance for the implementation can be found in the system documentation of the class /BOBF/CL_FRW_AUTHORITY_CHECK.

Authorization checked queries

For performance reasons, BOPF runs authorization checks directly on the database when executing generic queries (queries without an implementation class). To achieve this, BOPF uses SADL Query and provides an authority condition provider for it. The authority condition provider is returned by the get_query_condition_provider method of the authorization check class.

Restriction: For instance, if an authorization field mapping uses an implemented association, this implementation can‘t be executed on the database. The authority check therefore needs to be completely modeled.

Regarding implemened queries, the query implementation is responsible to return authorization checked results. BOPF only runs static checks for these.

The method /BOBF/CL_LIB_Q_SUPERCLASS->QUERY_USING_SADL can be called by a query implementation to get authorization checked results for a business object node.

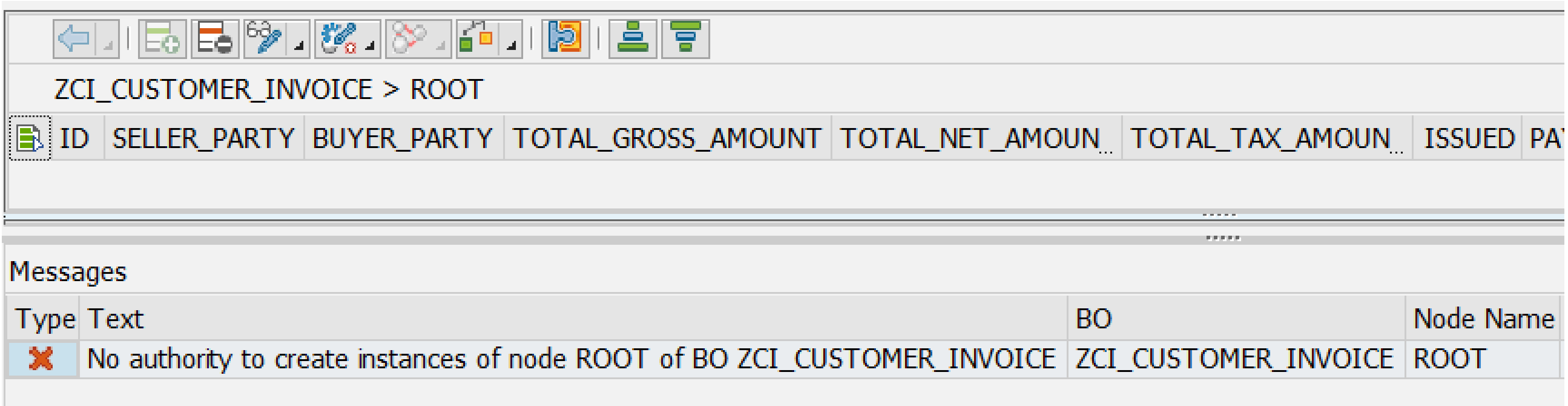
In rare cases, applications may want to define an own SADL authority condition provider. A how to guide to implement such a provider can be found on SCN http://scn.sap.com/docs/DOC-51476

Authorization Configuration - Result

After the previous steps the authorization configuration is done and active.

This means, that at runtime all service requests (`RETRIEVE`, …) for the business object are checked by BOPF regarding authorization. It is therefore necessary to configure appropriate roles (profiles) and assign them to the users.

If no role (profile) is configured and assigned to the user, this user consequently has no authorizations for the BO and, thus, e.g. a CREATE activity on the ROOT node would fail in the test shell as follows:

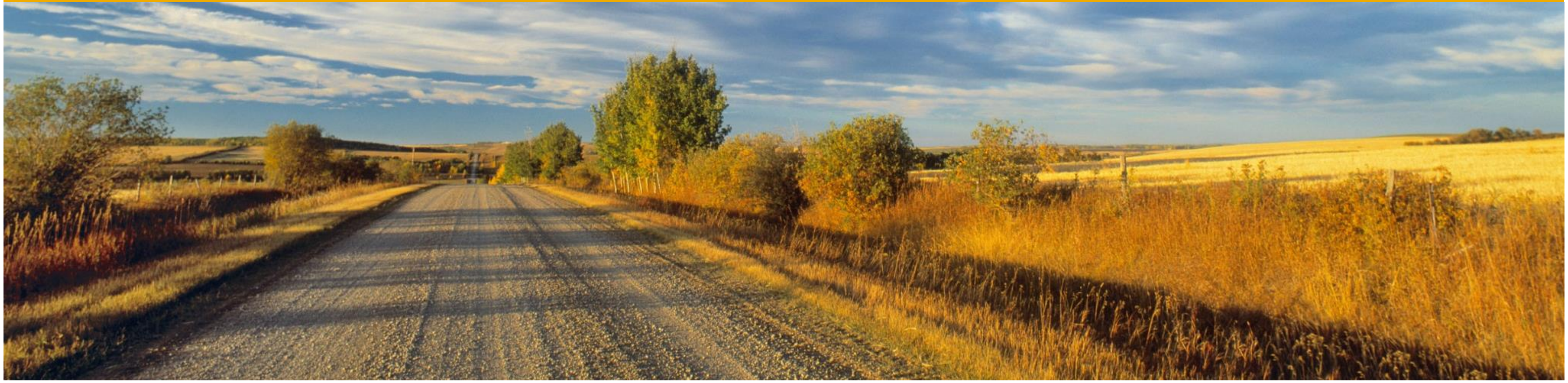


ZCI_CUSTOMER_INVOICE > ROOT

ID	SELLER_PARTY	BUYER_PARTY	TOTAL_GROSS_AMOUNT	TOTAL_NET_AMOUN...	TOTAL_TAX_AMOUN...	ISSUED	PA

Messages

Type	Text	BO	Node Name
✗	No authority to create instances of node ROOT of BO ZCI_CUSTOMER_INVOICE	ZCI_CUSTOMER_INVOICE	ROOT

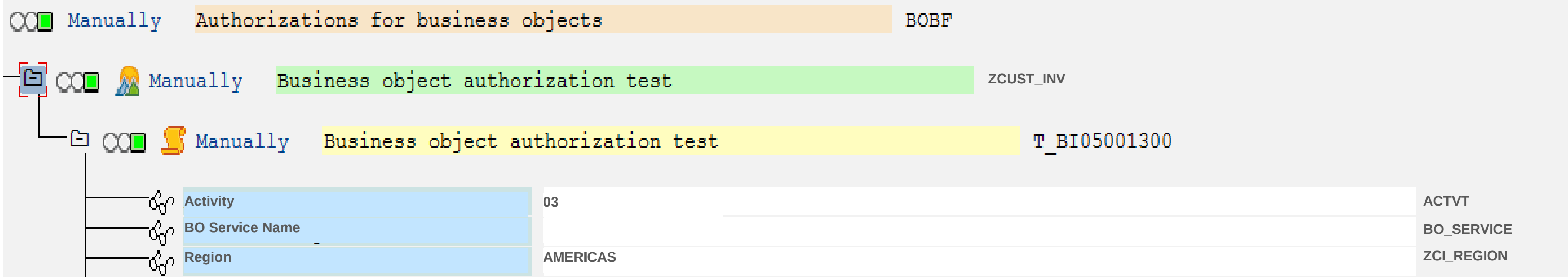


Role Configuration Examples

Authorization Configuration – Role for DISPLAY

To allow a group of users to display invoices for the buyer_region AMERICAS, configure the following role (see screenshot) in transaction PFCG.

Assign the role to the users afterwards.



Manually Authorizations for business objects BOBF

Manually Business object authorization test ZCUST_INV

Manually Business object authorization test T_BI05001300

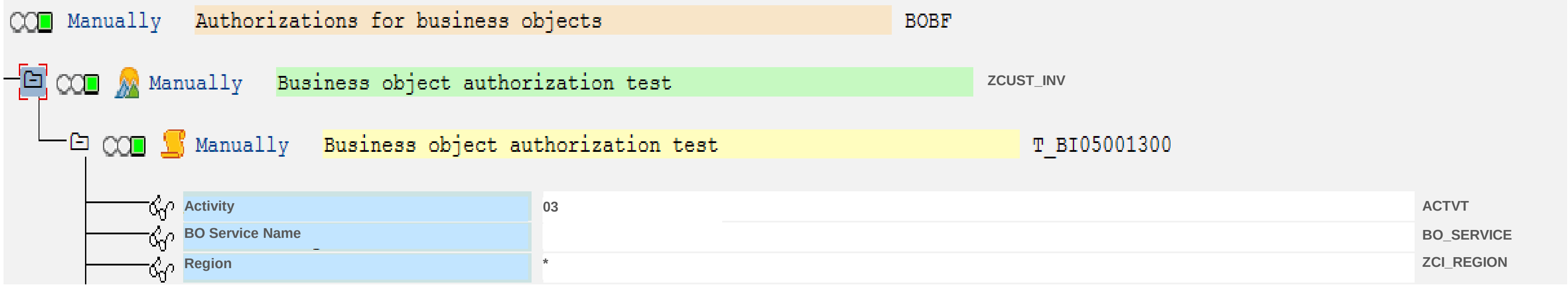
Activity	03	ACTVT
BO Service Name		BO_SERVICE
Region	AMERICAS	ZCI_REGION

Recap: The authorization field ZCI_REGION is mapped to the data field buyer_region on the ROOT node.

Hint: 03 is the code for the activity DISPLAY.

Authorization Configuration – Role for DISPLAY

To allow a group of users to display invoices for all `buyer_regions,` configure the following role:



Manually Authorizations for business objects BOBF

Manually Business object authorization test ZCUST_INV

Manually Business object authorization test T_BI05001300

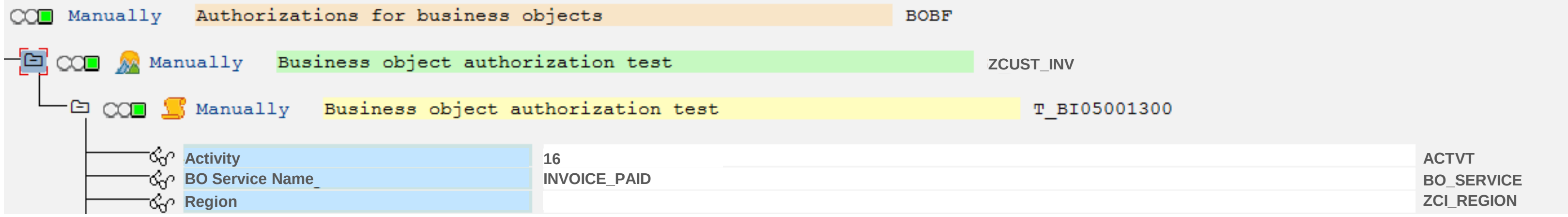
Activity	03	ACTVT
BO Service Name		BO_SERVICE
Region	*	ZCI_REGION

Hint: `03` is the code for the activity `DISPLAY.`

Authorization Configuration – Role for EXECUTE

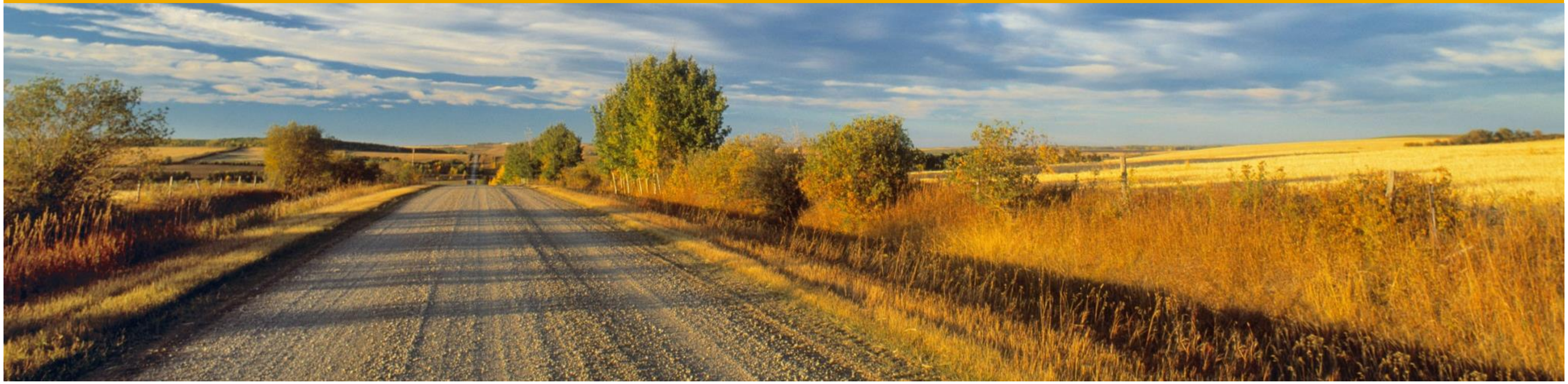
To allow a group of users to call the action `invoice_paid`, configure the following role (see screenshot) in transaction `PFCG`.

Assign the role to the users afterwards.



```
Manually   Authorizations for business objects      BOBF
  Manually   Business object authorization test       ZCUST_INV
    Manually   Business object authorization test       T_BI05001300
      Activity            16              ACTVT
      BO Service Name_    INVOICE_PAID    BO_SERVICE
      Region                              ZCI_REGION
```

Hint: `16` is the code for the activity `EXECUTE`.



Runtime

Runtime

The **generic authorization runtime implementation consists of BOPF** (service manager, authority handler) **and the authority check library class**. The latter one can be substituted by an application-specific check class.

At runtime, BOPF **evaluates each service request and delegates the** static or complete (static and instance) **authority check request to the check class** that usually is the library class.

The **authority check library class evaluates the authorization configuration** (recap: authorization object is assigned to node and authorization field is mapped to node field) **and maps the check request to `authority-check` statement calls**.

The **authority check library class tries to reduce the number of calls of `authority-check`** to a minimum by **buffering** and by the **concept of *equivalence groups***.

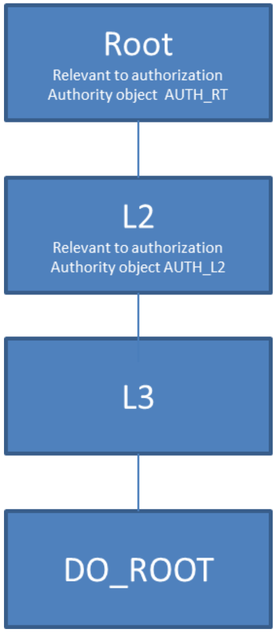
The BO implementation itself is considered to be privileged. Therefore, implementaion classes of validations, determinations etc. always have access to the data.

Runtime - Propagation to Subnodes

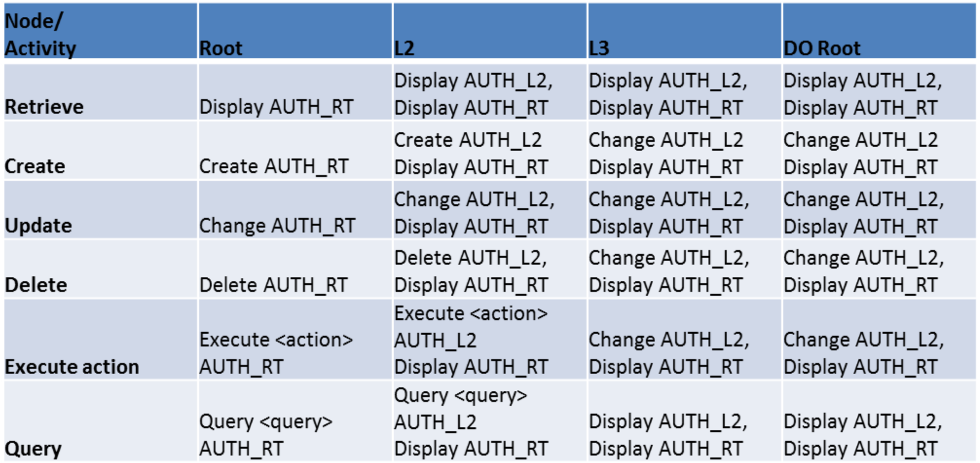
Authorizations of higher-level nodes **apply implicitly to their compositions.** E.g. a user is not allowed to display ITEM instances, if he does not have the display authority for the ROOT node.

The following table lists the checks that are done per activity in such a case. E.g. for a `CREATE` activity on node L2 the `CREATE` authorization on L2, and the `DISPLAY` authorization on ROOT are checked.

BO Model



Authority checks per node/activity



Node/ Activity	Root	L2	L3	DO Root
Retrieve	Display AUTH_RT	Display AUTH_L2, Display AUTH_RT	Display AUTH_L2, Display AUTH_RT	Display AUTH_L2, Display AUTH_RT
Create	Create AUTH_RT	Create AUTH_L2 Display AUTH_RT	Change AUTH_L2 Display AUTH_RT	Change AUTH_L2 Display AUTH_RT
Update	Change AUTH_RT	Change AUTH_L2, Display AUTH_RT	Change AUTH_L2, Display AUTH_RT	Change AUTH_L2, Display AUTH_RT
Delete	Delete AUTH_RT	Delete AUTH_L2, Display AUTH_RT	Change AUTH_L2, Display AUTH_RT	Change AUTH_L2, Display AUTH_RT
Execute action	Execute <action> AUTH_RT	Execute <action> AUTH_L2 Display AUTH_RT	Change AUTH_L2, Display AUTH_RT	Change AUTH_L2, Display AUTH_RT
Query	Query <query> AUTH_RT	Query <query> AUTH_L2 Display AUTH_RT	Display AUTH_L2, Display AUTH_RT	Display AUTH_L2, Display AUTH_RT

Activity Codes & Core Services

What are the activity codes for a certain core service?

Core Service	Activity Code
MODIFY	Create 01 / Change 02 / Delete 06
RETRIEVE	Display 03
RETRIEVE_BY_ASSOCIATION	Display 03
RETRIEVE_CODE_VALUE_SET	Display 03
RETRIEVE_DEFAULT_ACTION_PARAM	Display 03
RETRIEVE_DEFAULT_NODE_VALUES	Display 03
RETRIEVE_DEFAULT_QUERY_PARAM	Display 03
RETRIEVE_PROPERTY	Display 03
QUERY	Query AF
CHECK_ACTION	Display 03
CHECK_AND_DETERMINE	Change 02
CHECK_CONSISTENCY	Check 39
CONVERT_ALTERN_KEY	Display 03
DO_ACTION	Execute 16

Thank you

© 2012 SAP AG. All rights reserved.