BOPF Properties

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

Static Properties

Dynamic Properties

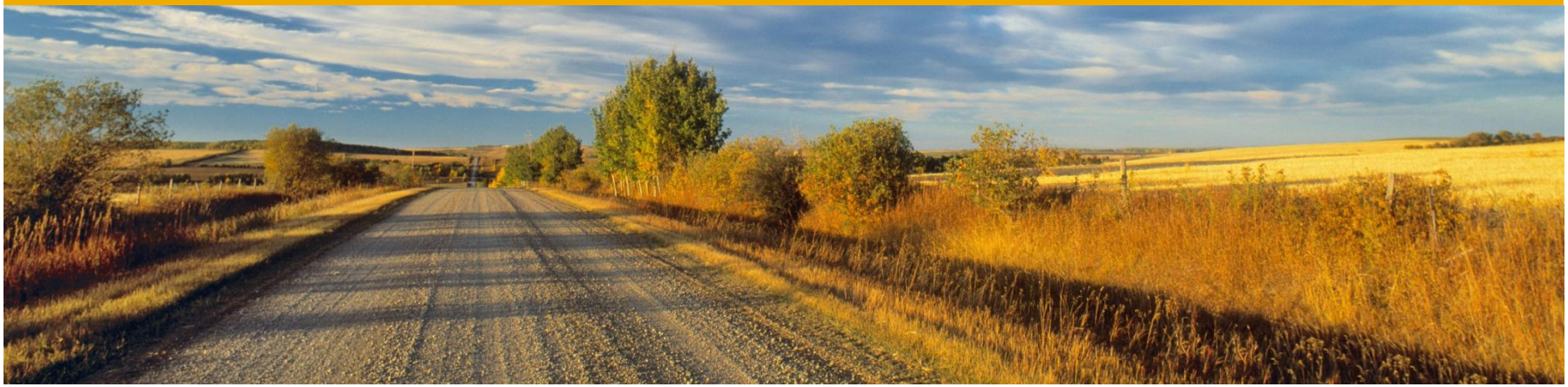
- Lock Dependency
- S&AM Dependency
- Sub-Tree Properties
- Application-Specific Properties

Priority of Properties

Property Change Notifications

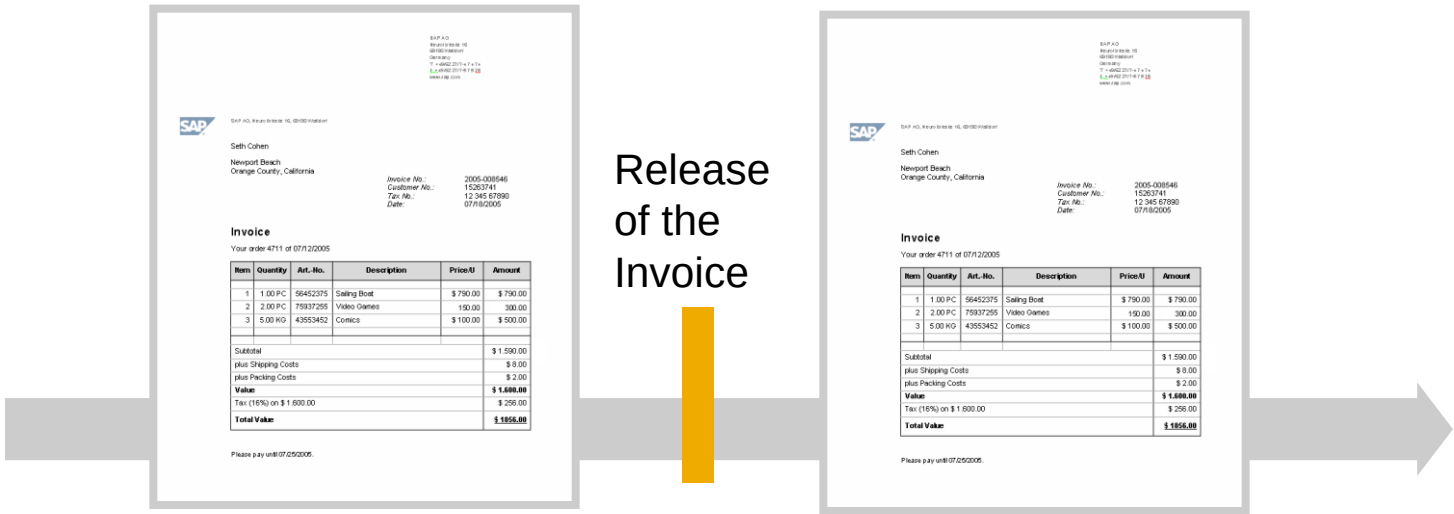
Property Checks

Question & Answers



Introduction

Introduction

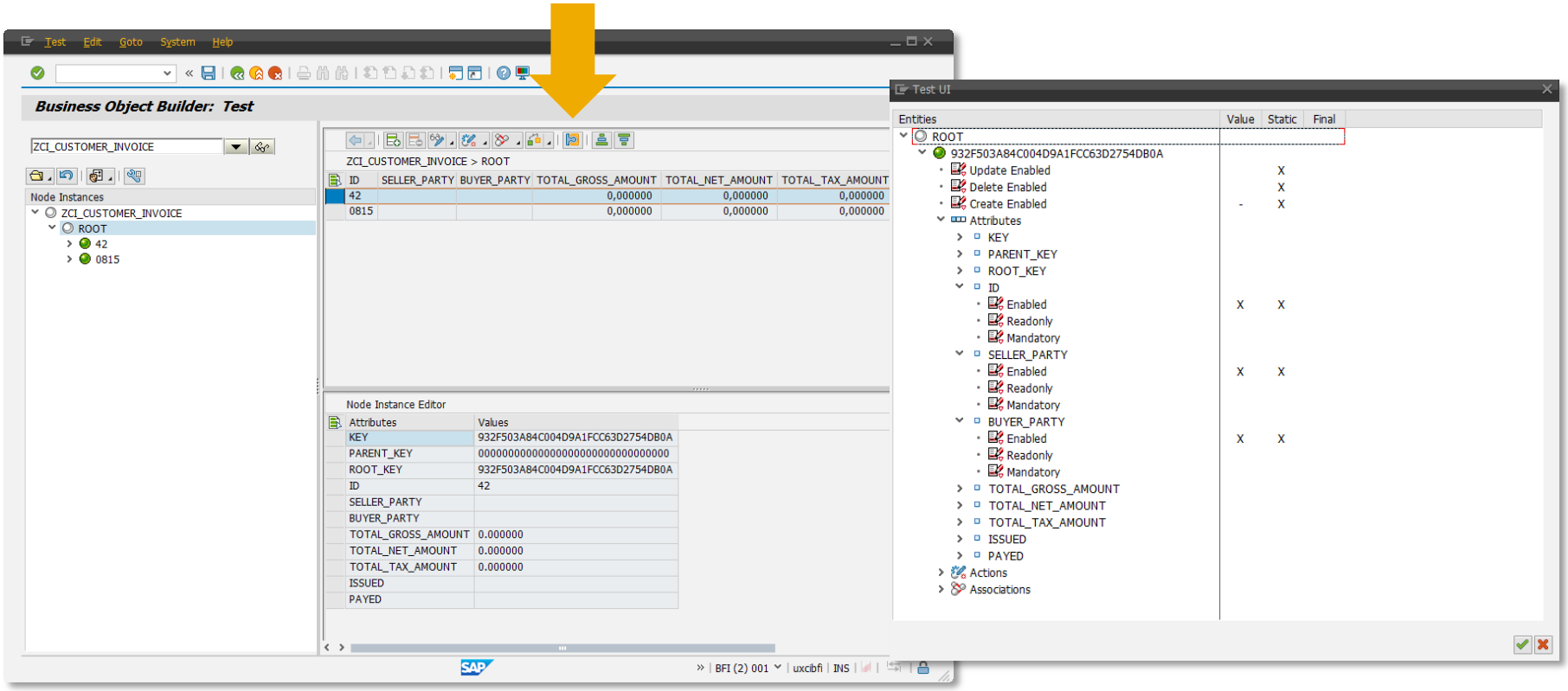


The sales agent must be able to edit the customer invoice until the invoice is released. After the invoice is released, all invoice attributes must be displayed in the UI as being read-only.

Properties can be used, for instance, to control the modifiability of node instance attributes.

Introduction

Example in Test UI



The test UI (`/BOBF/TEST_UI`) shows all properties of the selected instances in a dialog box.

For testing purposes, the properties can be ignored (“ Obey Properties” button).

Disabled attributes are marked as disabled in “Node Instance Editor”.

Example: Some attributes of this `ROOT` node instance are currently displayed as “ readonly”.

Introduction
Names

The following properties can be set:

- **Enabled**: Entity can be used everywhere (e.g. dialog box for action parameter input hidden).
- **Read-Only**: Entity cannot be changed.
- **Mandatory**: Entity must be given a value. Two different semantics:
  Must be filled in the initial create/update modification call.
  Must be filled, at the latest, by the finalize phase.
- **Create-Enabled**: New instances of the root node can be created.
  Other nodes can be created using an association.
- **Update-Enabled**: Existing instances of the node can be changed.
- **Delete-Enabled**: Existing instances of the node can be deleted.

Introduction

Availability

Entity ▼ / Possible Property ▶	ENABLED	READ_ONLY	MANDATORY	CREATE_ENABLED	UPDATE_ENABLED	DELETE_ENABLED
Node				X	X	X
Node attribute	X	X	X			
Association	X			X		
Association parameter	X					
Action	X					

Introduction
Dynamic vs. Static

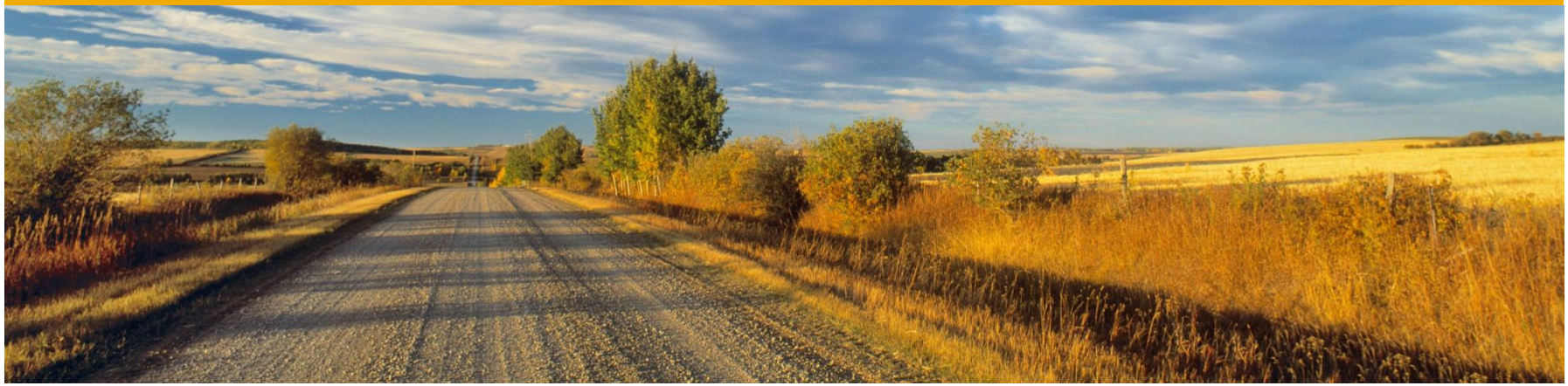
Properties can either be set as static or dynamic.

Static properties (configured only at design time in `/BOBF/CONF_UI`)

- Static property (can be overridden by dynamic properties)
- Final static property (cannot be overridden by any dynamic property)

Dynamic properties (can be changed at runtime)

- Lock-dependent
  (e.g. attributes are set implicitly to read-only if the consumer has no lock)
- S&AM properties
  Properties for actions dependent on the status schema and the current state
- Application-specific dynamic properties (including subtree properties)
  Set by property determinations (e.g. to display attributes as read-only for released customer invoices)



Static Properties

Static Properties
Overview

- Configured at design time in the BOPF configuration UI (`/BOBF/CONF_UI`).
- Static properties are configured separately for each node category.
- Static attribute properties not marked as final can be overridden by dynamic properties at runtime.
- Static attribute properties marked as final cannot be overridden by dynamic properties at runtime.

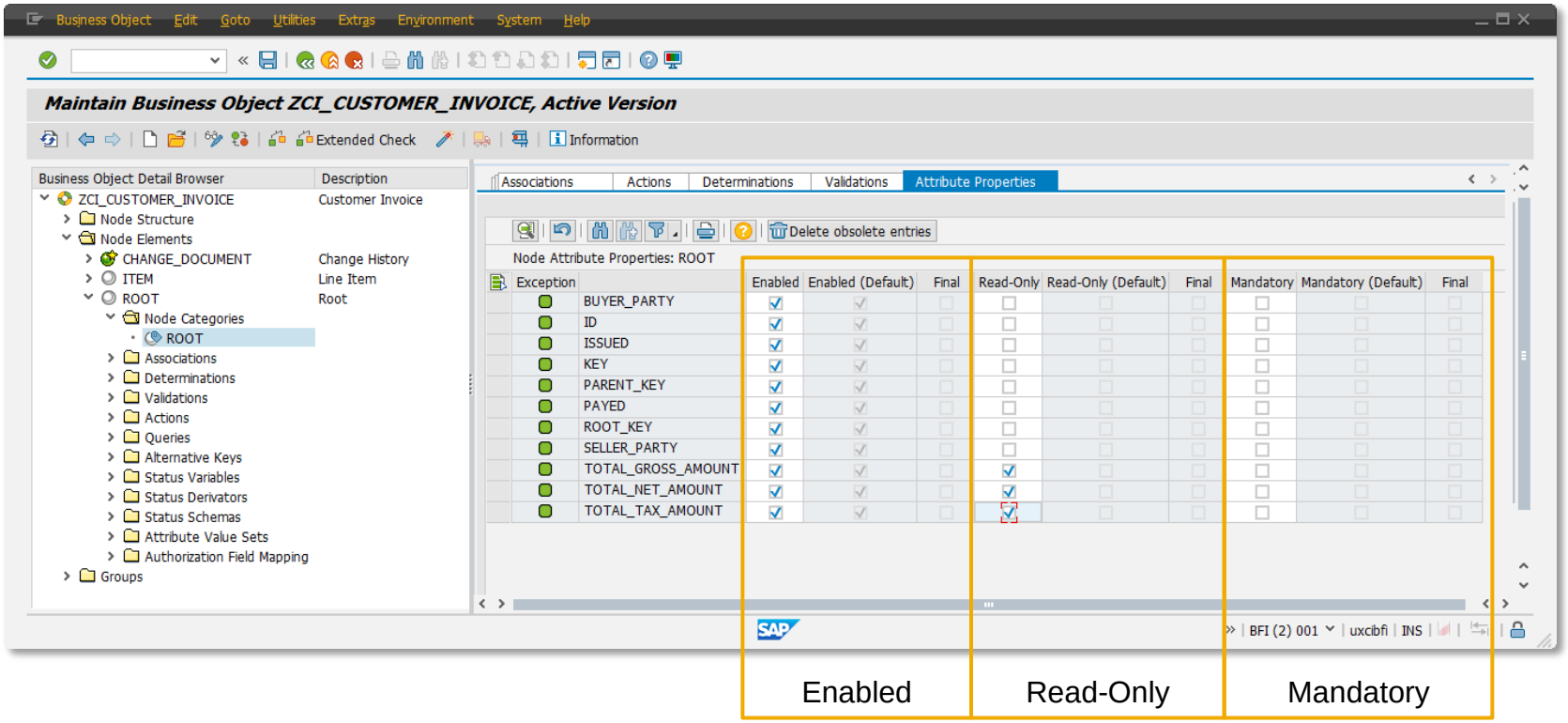
Static Properties

Availability

Entity / Possible Property	ENABLED	READ_ONLY	MANDATORY	CREATE_ENABLED	UPDATE_ENABLED	DELETE_ENABLED
Node				X	X	X
Node attribute	X	X	X			
Association	X					
Association parameter						
Action	X					

Static Properties

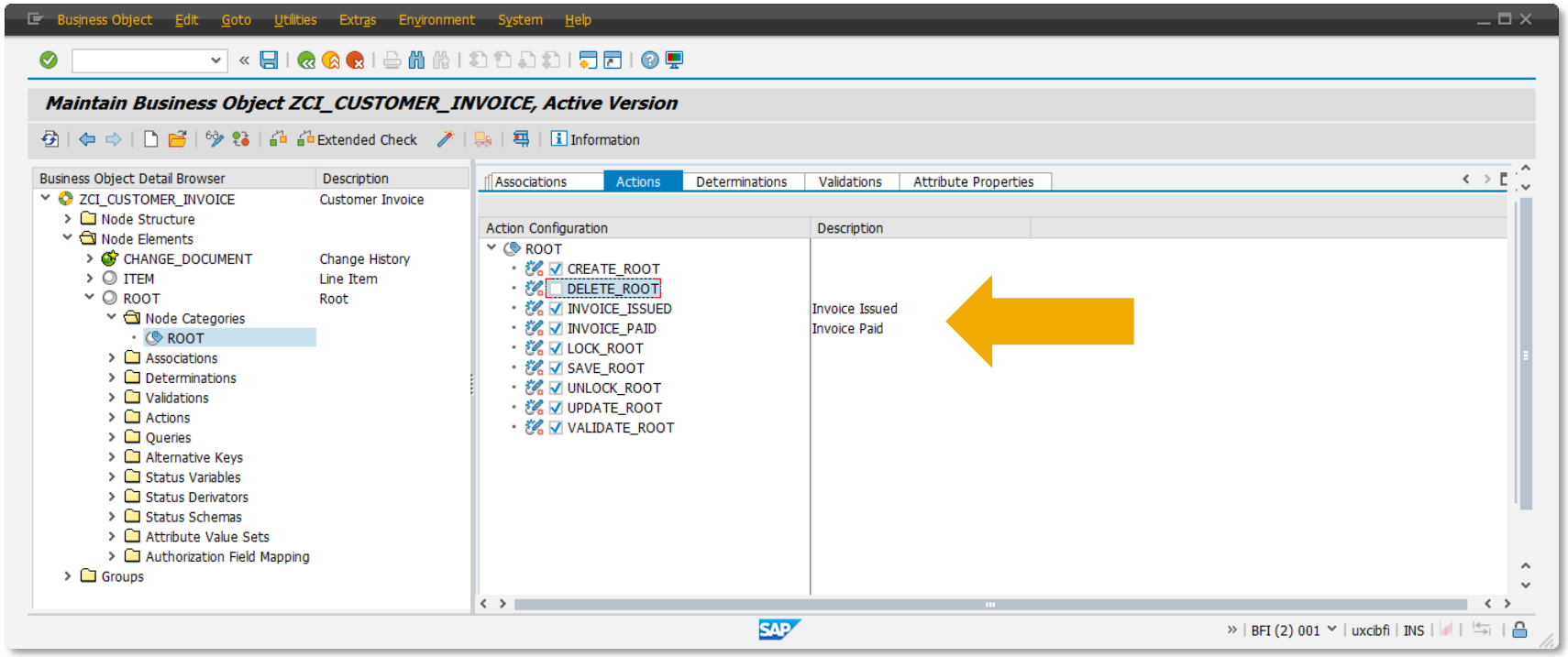
For Node Attributes



Define the “(Final) Enabled”, “(Final) Read-Only”, and “(Final) Mandatory” property for each attribute of each node category.

Static Properties

For Nodes and Actions

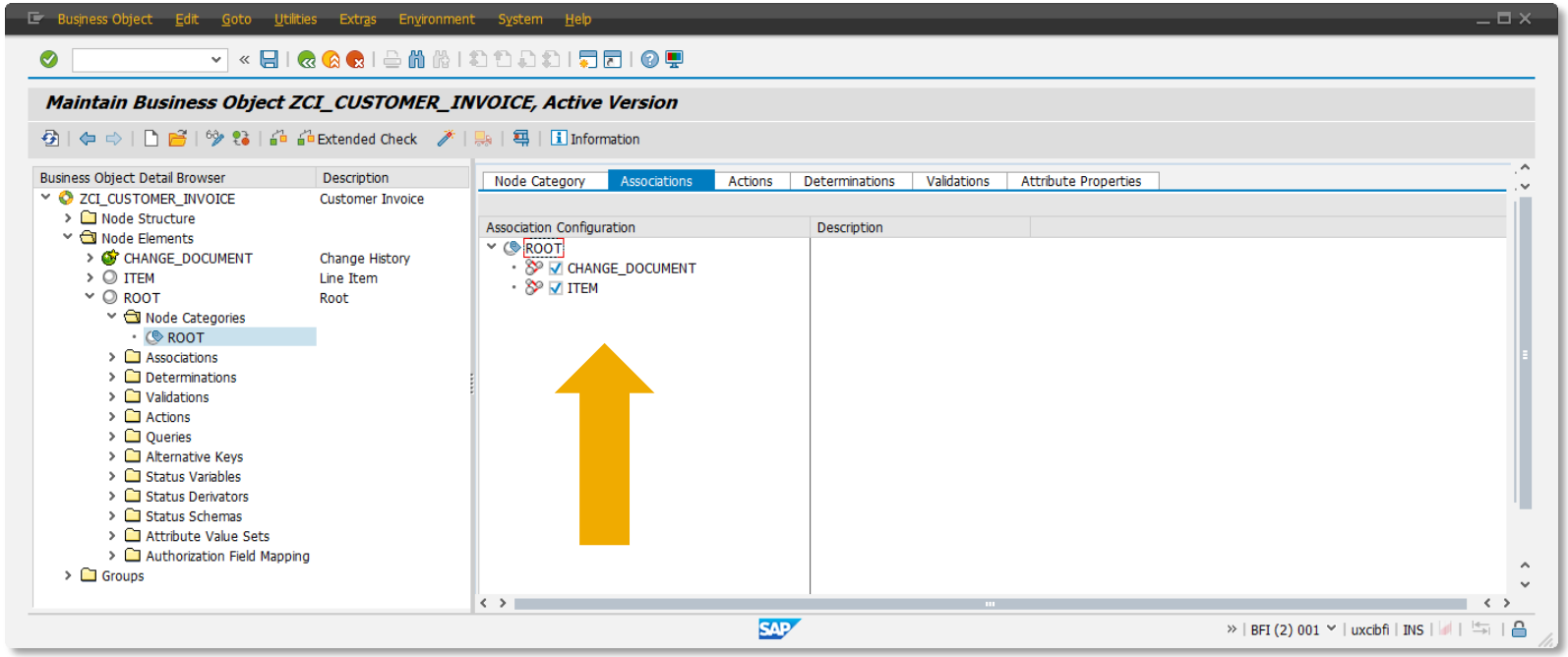


Choose “ Create Enabled” or “Final Create Disabled”, “ Delete Enabled” or “Final Delete Disabled”, and “ Update Enabled” or “Final Update Disabled” property of a node (affects all node instances) by selecting or deselecting the corresponding framework action.

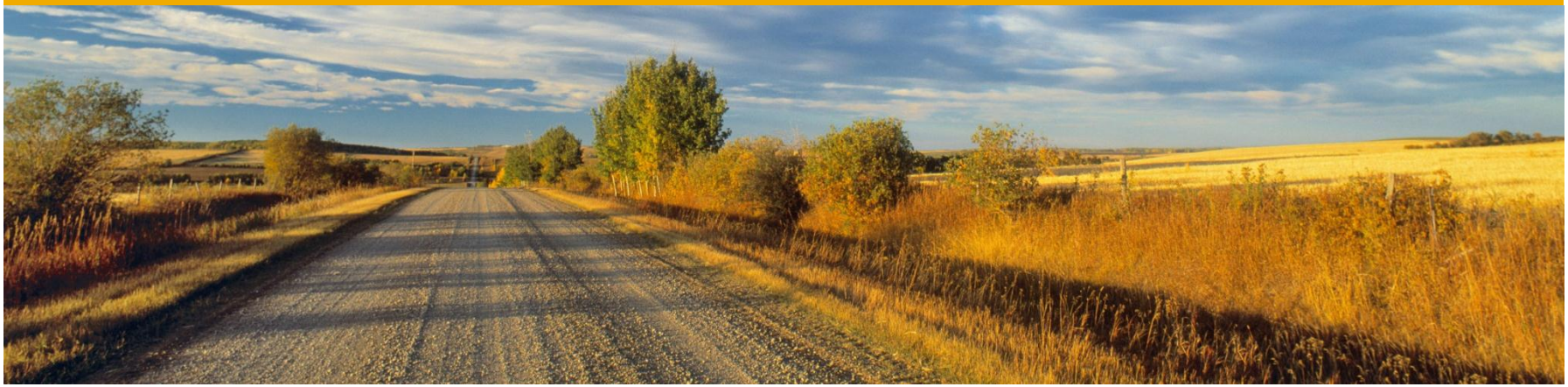
Choose “ Enabled” or “Final Disabled” for an action.

Static Properties

For Associations



Select “ Enabled” or “Final Disabled” property for an association.



Dynamic Properties

Dynamic Properties
Overview

- Can be changed at runtime.
- Are not persistent.
- Cannot be deleted, but can be overridden.
- Are always instance-related.

Dynamic Properties
Availability

Entity ▼ / Possible Property ▶	ENABLED	READ_ONLY	MANDATORY	CREATE_ENABLED	UPDATE_ENABLED	DELETE_ENABLED
Node					X	X
Node attribute	X	X	X			
Association	X			X		
Association parameter	X					
Action	X					

Dynamic Properties
Lock Dependency

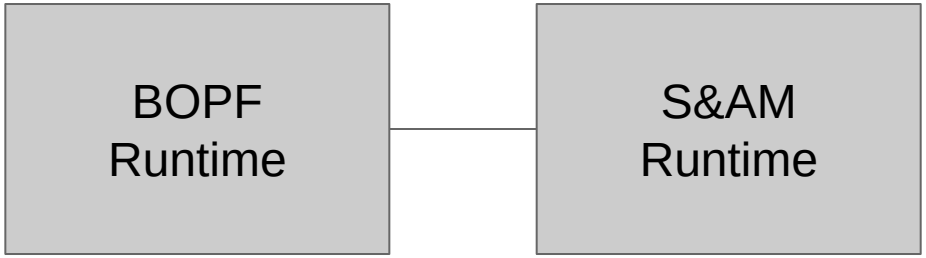
If a certain node instance is not locked optimistically or exclusively, BOPF sets the following properties automatically:

Associations: “Create Disabled”

Nodes: “Update Disabled” and “Delete Disabled”

Dynamic Properties

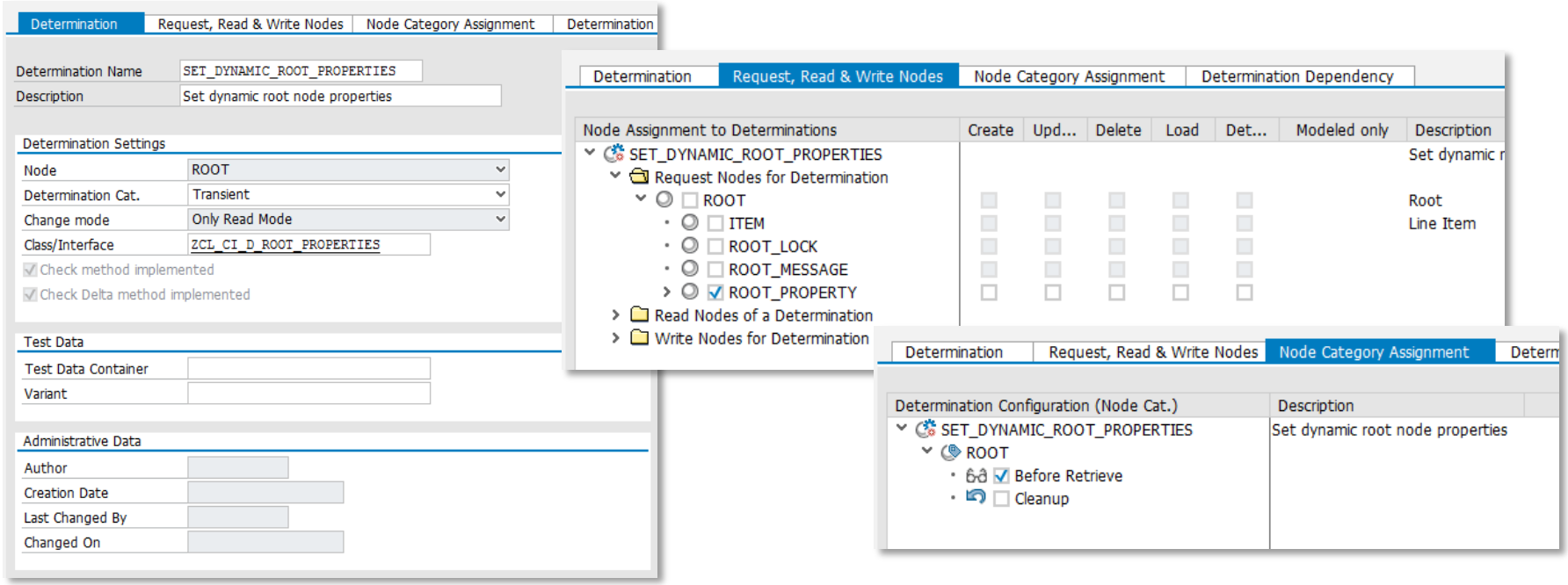
Status and Action Management



BOPF is able to request and inherit properties provided by S&AM runtime automatically (e.g. if a certain action is currently enabled or disabled).

This is done as soon as the `retrieve_property()` core service is called.

Dynamic Properties
Application-Specific



Application-specific properties must be created using **property determinations**.

- Create a determination on the node to which those entities are assigned that need dynamic properties.
- Use the property node as a request node and **do not** select any trigger conditions.
- Choose “ Before Retrieve” as the determination time.
- Implement the determination class.

Dynamic Properties

Application-Specific

```
* 1. Create an instance of the helper class
  DATA lo_set_property TYPE REF TO /bobf/cl_lib_h_set_property.
  CREATE OBJECT lo_set_property
    EXPORTING
      is_context        = is_ctx       " Determination context
      io_modify         = io_modify.   " Reference to modify object

* 2. Disable an action
  LO_SET_PROPERTY->SET_ACTION_ENABLED(
    EXPORTING
      iv_key            = ls_root-key
      iv_action_key     = /bobf/if_tst_cust_invoice_c=>sc_action-root-invoice_issued
      iv_value          = abap_false ).
```

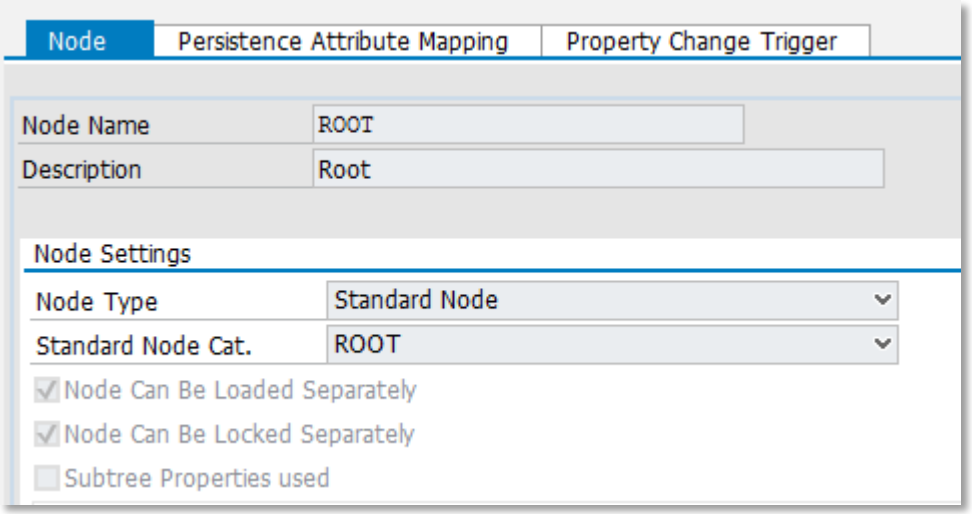
The `/BOBF/CL_LIB_H_SET_PROPERTY` helper class makes it easy to set the properties of different objects like actions, attributes, associations, and subtrees in determinations.

Use this class instead of modifying the property node.

The following methods are available:

- `SET_ASSOCIATION_CREATE_ENABLED()`
- `SET_ASSOCIATION_ENABLED()`
- `SET_ATTRIBUTE_ENABLED()`
- `SET_ATTRIBUTE_MANDATORY()`
- `SET_ATTRIBUTE_READ_ONLY()`
- `SET_ACTION_ENABLED()`
- `SET_NODE_DELETE_ENABLED()`
- `SET_NODE_UPDATE_ENABLED()`

Subtree Properties
Overview



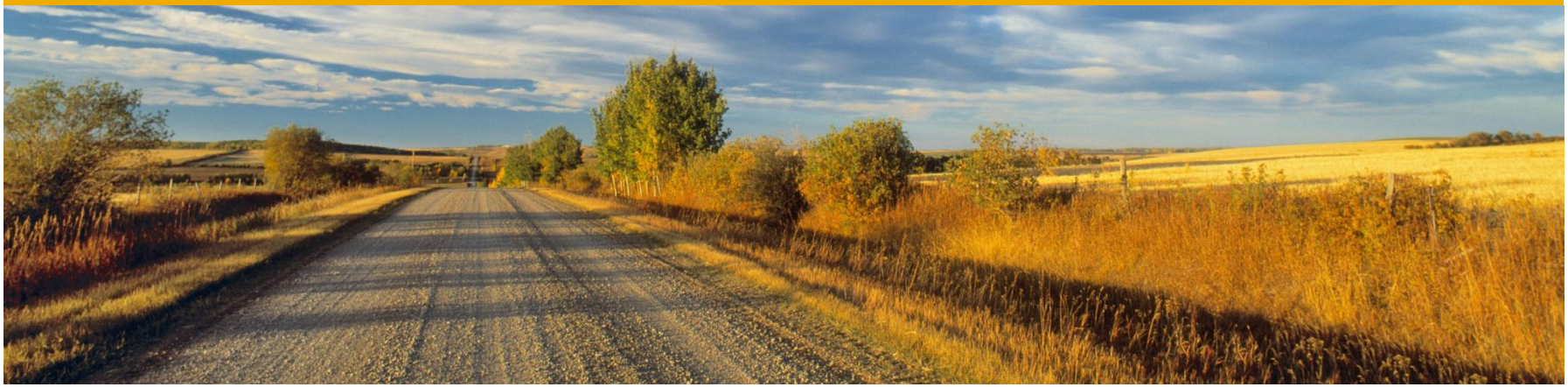
Subtree properties can be used to set the same property (`CREATE_ENABLED`, `UPDATE_ENABLED`, `DELETE_ENABLED`) for the whole subtree of a node instance.

The flag “Subtree Properties” used must be selected for the topmost node of the relevant subtree. At runtime, the subtree properties can be applied on this node.

Subtree Properties

Details

- Subtree properties are useful to restrict a subtree of node instances in a business object
- Subtree properties can also be set by the `/BOPF/CL_LIB_H_SET_PROPERTY` helper class (but do not forget the subtree flag in the configuration) .
- If a subtree property is deactivated, the other application-specific dynamic properties that were valid before this subtree property were set are valid again. This means a subtree property only “ covers” application-specific dynamic properties.
- Nested subtree properties are accumulated .
- A dependent object contained in the subtree scope is also affected.
- Use carefully (due to the impact on performance).
- Subtree properties cannot be overridden by other dynamic properties (e.g. “all fields should be read-only except for one field” will not work).



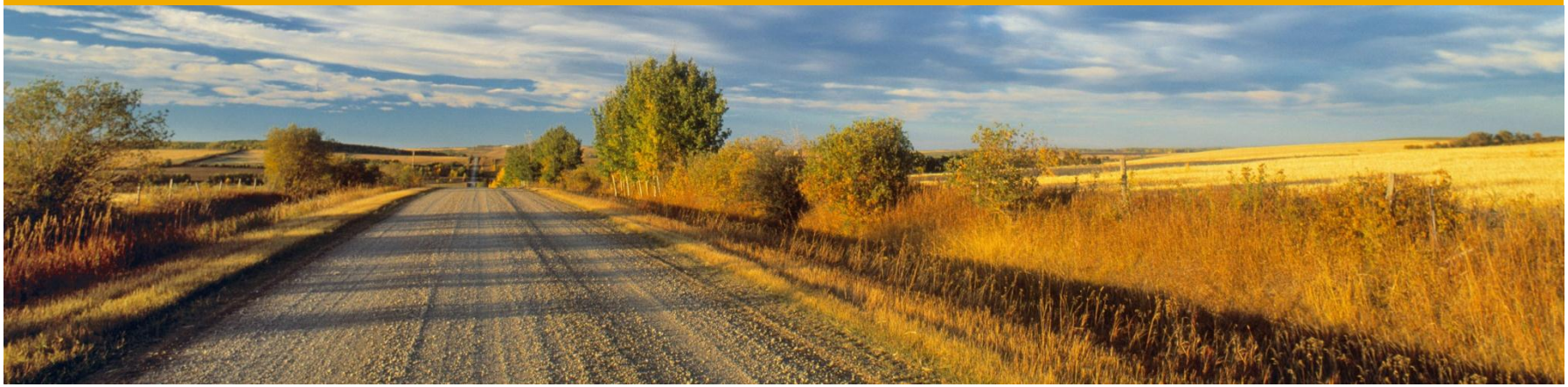
Priority of Properties

Priority of Properties

1. **Final static properties**
   (Configured in the node category configuration of `/BOBF/CONF_UI`.)
2. **Authority-dependent properties**
   (Automatically set by BOPF.)
3. **Lock-dependent properties**
   (Automatically set by BOPF.)
4. **Status and action management**
   (Automatically received by BOPF and configured at S&AM design time.)
5. **Application-specific dynamic poperties: Subtree properties**
   (Set using dynamic property determination and the helper class.)
6. **Other application-specific dynamic properties**
   (Set using dynamic property determination and the helper class.)
7. **Non-final static properties**
   (Configured in the node category configuration of `/BOBF/CONF_UI`.)

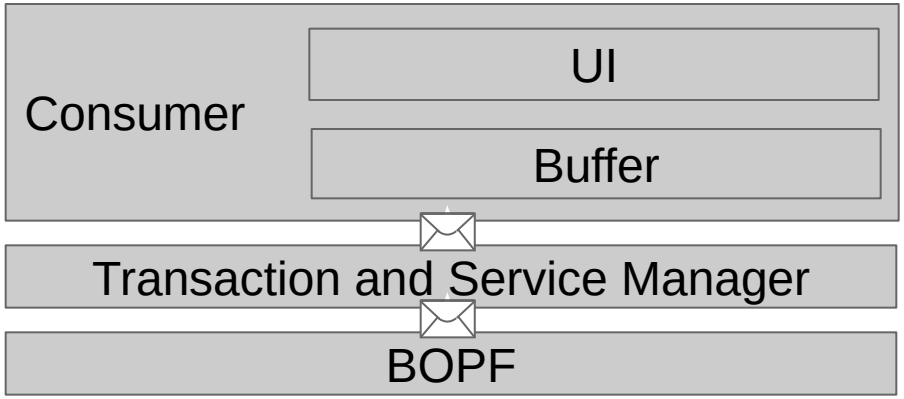
Examples

- S&AM properties override application-specific properties, but are overridden by lock-dependent properties.
- Properties for which the final flag was set can never be overruled by any other properties. Set the final flag with care.
- Non-final static properties are the weakest; use them as a kind of default property.



Property Change Notifications

Property Change Notifications



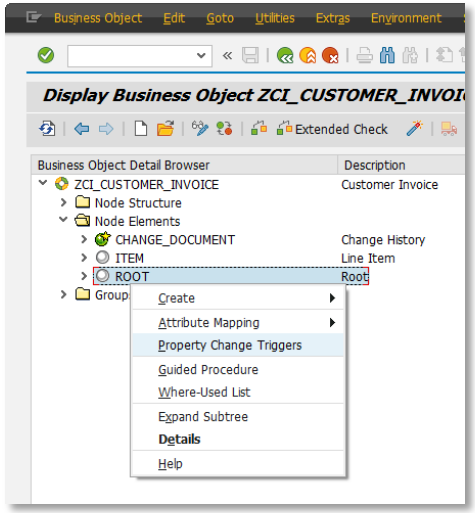
Example

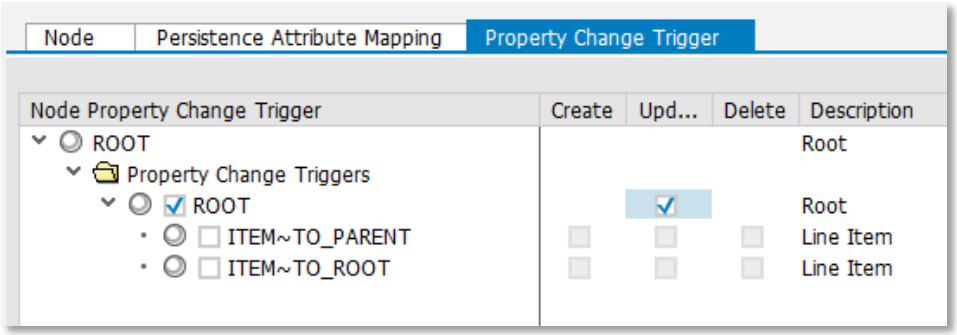
The consumer buffers all information received from BOPF in the buffer (also includes the properties returned by the `retrieve_property()` core service) → When the invoice is released (attribute " released" of the `ROOT` node is set to " `X`") → Amount attributes on the item must be set to read-only in the UI.
BUT: Consumer might not be aware of the fact that changing a `ROOT` node instance might modify the properties of the `ITEM` node instances → This means the consumer would not refresh its buffered (and now outdated) item properties.

To solve this: Consumer must be informed about the fact that changing the flag on the `ROOT` node might change the properties of the item attributes → Done using a property change notification (delivered in the `eo_message` object of the `ROOT`'s update modification call).

After receiving a property change notification, the consumer can refresh the buffered properties of the `ITEM` node by using `retrieve_properties()` again .

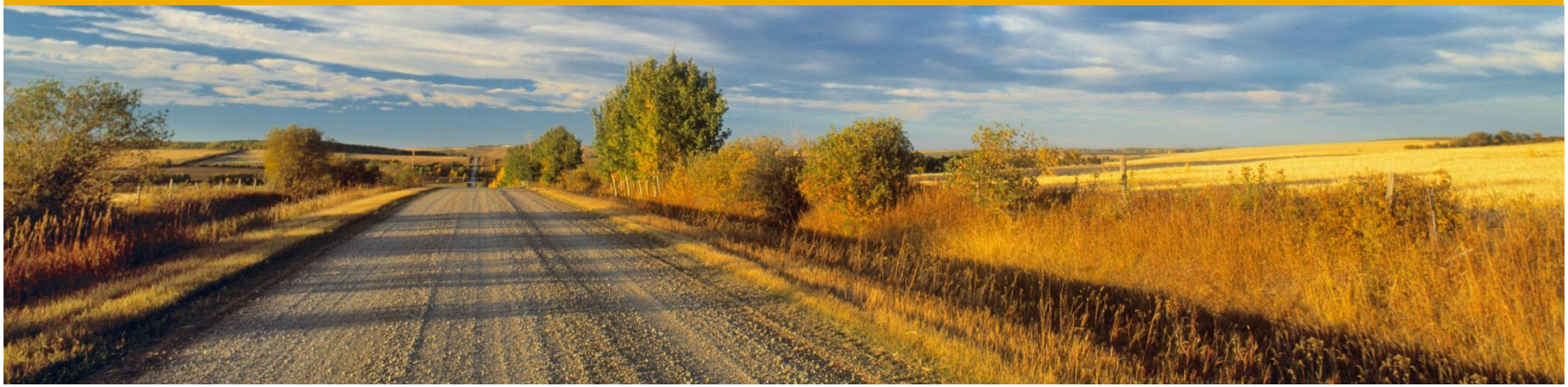
Property Change Notifications





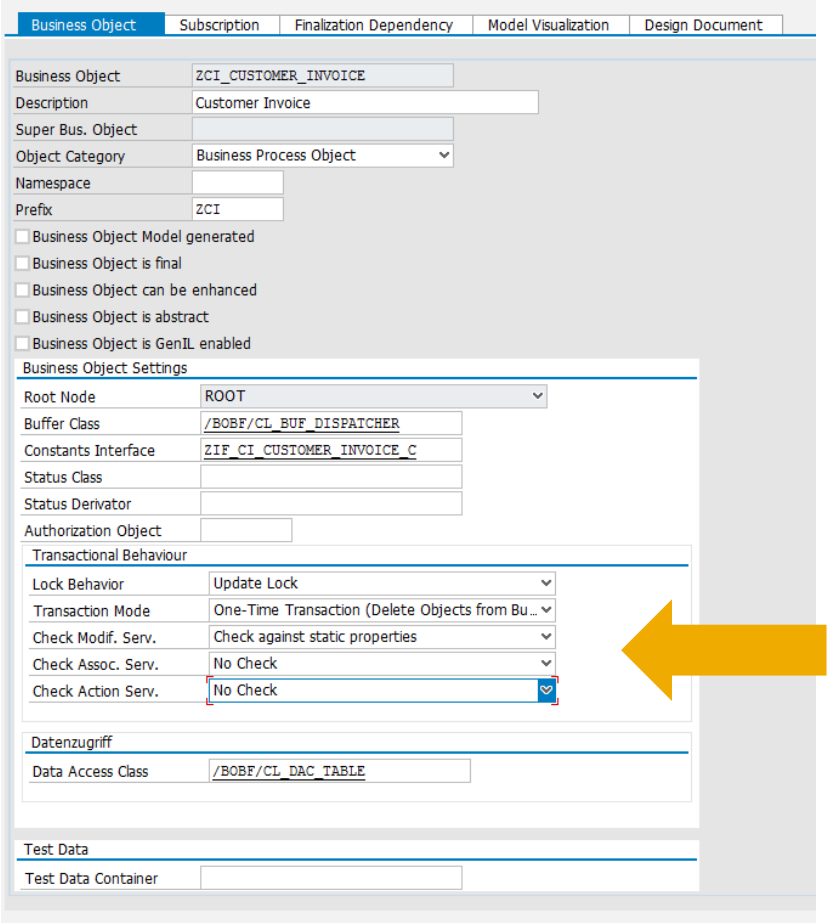
To create property change notifications automatically, property change triggers can be defined in the BOPF Configuration UI:

- Choose the request node and the trigger condition that trigger the sending of a property change notification (belonging to the node to which this trigger is assigned).
- Property change trigger can also be defined on other entities (e.g. actions), which is necessary if the action's properties depend on a create/update/delete of a related node instance
- Change notifications are only sent if the consumer called `retrieve_property()` on the node at least once before.
- If modeling is not sufficient, property change notifications can also be created from application code using the method `IO_MODIFY->NOTIFY_PROPERTY_CHANGE()`.

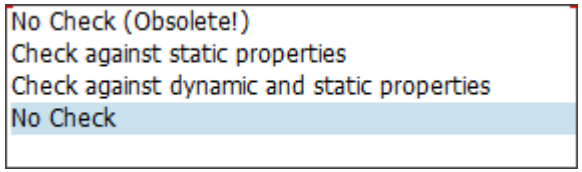


Property Checks

Automatic Check Properties



- BOPF is able to check whether properties are obeyed automatically and is, for instance, able to reject invalid modification calls.
- You can configure whether the modification, association, and action core service are checked automatically. If dynamic properties are not used , deactivate the dynamic property checks to improve performance. If static and dynamic properties are not used , you can disable all property checks.



Automatic Check Properties

Modifications

All external modification core service calls can be checked:

“Modify Create”:

- Association `CREATE_ENABLED` and `ENABLED`
- Node `CREATE_ENABLED`
- Attributes `READONLY` and `ENABLED` (`changed_fields`)

“Modify Update”:

- Node `UPDATE_ENABLED`
- Attributes `READONLY` and `ENABLED`

“Modify Delete”:

- Node `DELETE_ENABLED`

Use library determination `/BOBF/CL_LIB_V_MANDATORY_ATTR` and configure it either as a consistency validation (usually the trigger condition “check”) or as an action validation at “ save” (executed during `check_before_save`) for checking the mandatory property.

Hint: Internal calls (e.g. modifications using `io_modify` within an action implementation) are never checked against properties.

Automatic Check Properties
Associations

- Automatic association property checks ensure that only associations that are enabled can be used by a `retrieve_by_association` call.
- `to_root` and the `to_parent` framework associations are never disabled and are never checked.
- If `retrieve_by_association` is executed to access the “before” image, only the static properties are checked (because properties are by definition only related to the current state).

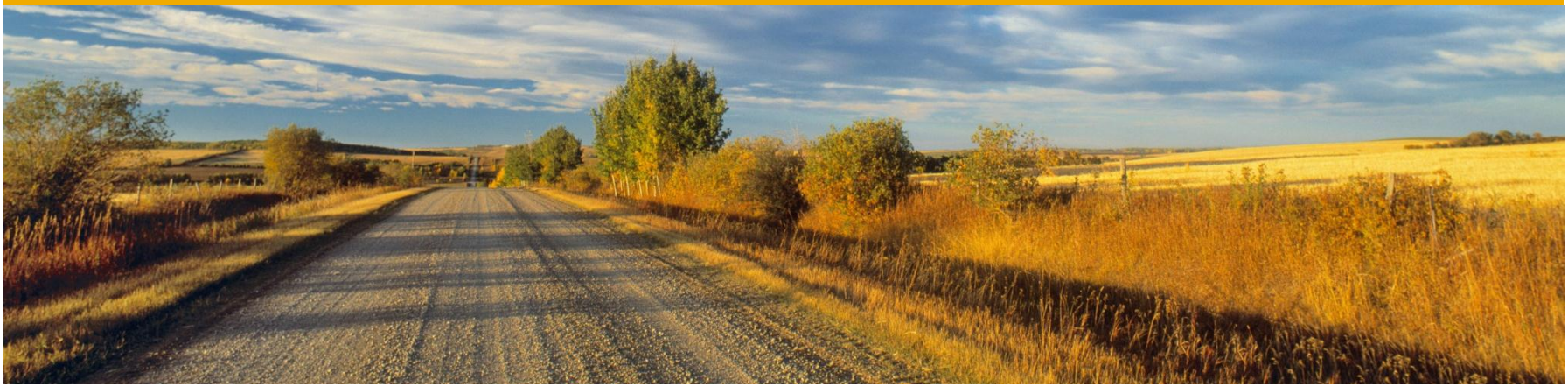
Hint: Internal calls (e.g. `retrieve_by_association` within an action implementation) are never checked against properties.

Automatic Check Properties

Actions

Checks whether a certain action is disabled by properties and rejects the execution if this is the case.

Hint: Internal calls (e.g. calling an action out of an action implementation) are never checked against properties.



Questions & Answers

Questions

- I have configured a node attribute as read-only. Does this also influence the creation of a node instance with respect to modification property checks?
- Can I set dynamic properties of node A using a property determination assigned to node B (with no triggering condition to A)?
- Is it possible to modify property nodes directly?
- What happens if I nest subtree properties?
- How do I set the properties of node (instances) of dependent objects?

Answers

I have configured a node attribute as read-only. Does this also influence the creation of a node instance with respect to modification property checks?
Yes, both the creation and the update are affected.

May I set dynamic properties of node A using a property determination assigned to node B (with no triggering condition to A)?
No, the consumer might retrieve only properties of node A and thus the property determinations assigned to B are not executed at all.

Is it possible to modify property nodes directly?
Yes, but do not do this. Always use the helper method to set properties.

What happens if I nest subtree properties?
The restrictions are accumulated.

How do I set the properties of node (instances) of dependent objects?
The properties set on the delegated node are also valid for all entities of the DO itself.

Thank you

© 2012 SAP AG. All rights reserved.