

BOPF Properties

SAP AG, 2012

The SAP logo is located in the bottom left corner of the image. It consists of the letters 'SAP' in a bold, white, sans-serif font, set against a blue rectangular background that has a slight gradient and a small white triangle at the bottom left corner.

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

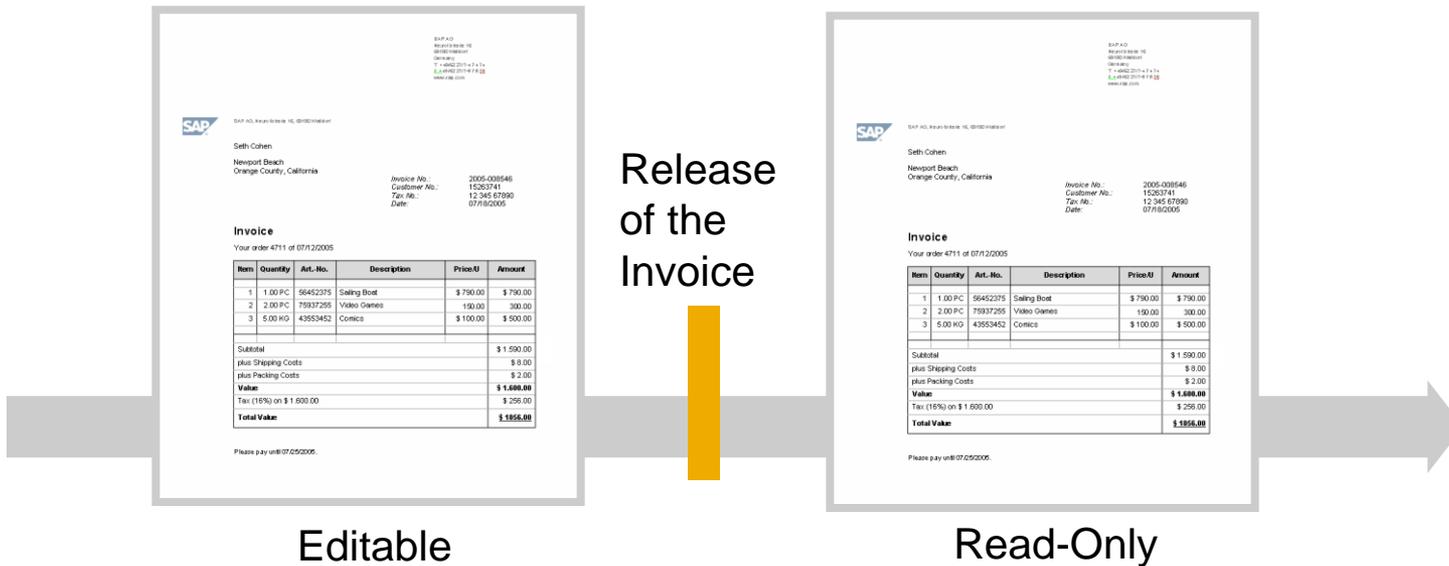
Property Checks

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 screenshot displays the SAP Business Object Builder Test UI. A yellow arrow points to the 'Test UI' window. The main window shows a table of test instances for 'ZCI_CUSTOMER_INVOICE'. The 'Node Instance Editor' shows the properties of the selected instance (ID 42). The 'Test UI' dialog box is open, showing a tree view of the entity structure and a table of properties.

ID	SELLER_PARTY	BUYER_PARTY	TOTAL_GROSS_AMOUNT	TOTAL_NET_AMOUNT	TOTAL_TAX_AMOUNT
42			0,000000	0,000000	0,000000
0815			0,000000	0,000000	0,000000

Attributes	Values
KEY	932F503A84C004D9A1FCC63D2754DB0A
PARENT_KEY	00000000000000000000000000000000
ROOT_KEY	932F503A84C004D9A1FCC63D2754DB0A
ID	42
SELLER_PARTY	
BUYER_PARTY	
TOTAL_GROSS_AMOUNT	0.000000
TOTAL_NET_AMOUNT	0.000000
TOTAL_TAX_AMOUNT	0.000000
ISSUED	
PAYED	

Entities	Value	Static	Final
ROOT			
932F503A84C004D9A1FCC63D2754DB0A			
Update Enabled		X	
Delete Enabled			X
Create Enabled	-	X	
Attributes			
KEY			
PARENT_KEY			
ROOT_KEY			
ID			
Enabled	X	X	
ReadOnly			
Mandatory			
SELLER_PARTY			
Enabled	X	X	
ReadOnly			
Mandatory			
BUYER_PARTY			
Enabled	X	X	
ReadOnly			
Mandatory			
TOTAL_GROSS_AMOUNT			
TOTAL_NET_AMOUNT			
TOTAL_TAX_AMOUNT			
ISSUED			
PAYED			
Actions			
Associations			

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

Maintain Business Object ZCI_CUSTOMER_INVOICE, Active Version

Business Object Detail Browser

Node Attribute Properties: ROOT

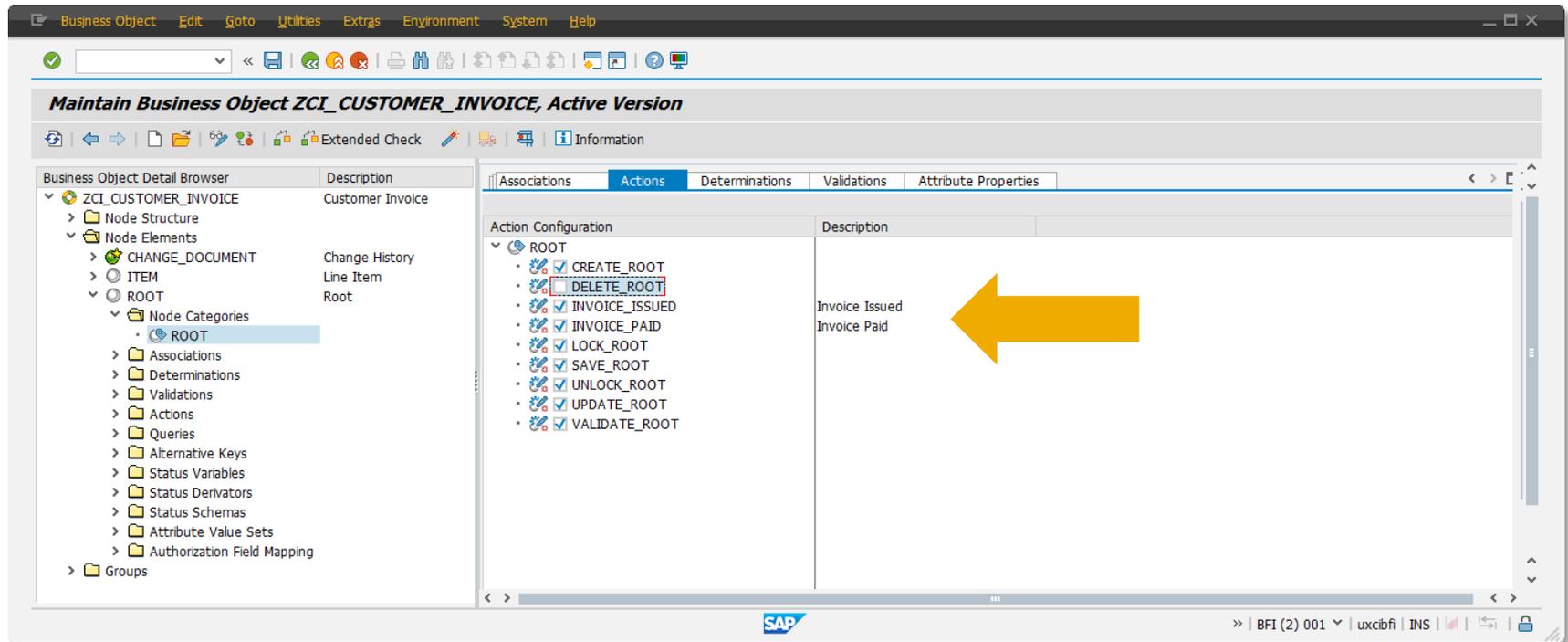
Exception	Enabled	Enabled (Default)	Final	Read-Only	Read-Only (Default)	Final	Mandatory	Mandatory (Default)	Final
BUYER_PARTY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ISSUED	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
KEY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PARENT_KEY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PAYED	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ROOT_KEY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SELLER_PARTY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TOTAL_GROSS_AMOUNT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
TOTAL_NET_AMOUNT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
TOTAL_TAX_AMOUNT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				

Enabled Read-Only Mandatory

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

Static Properties

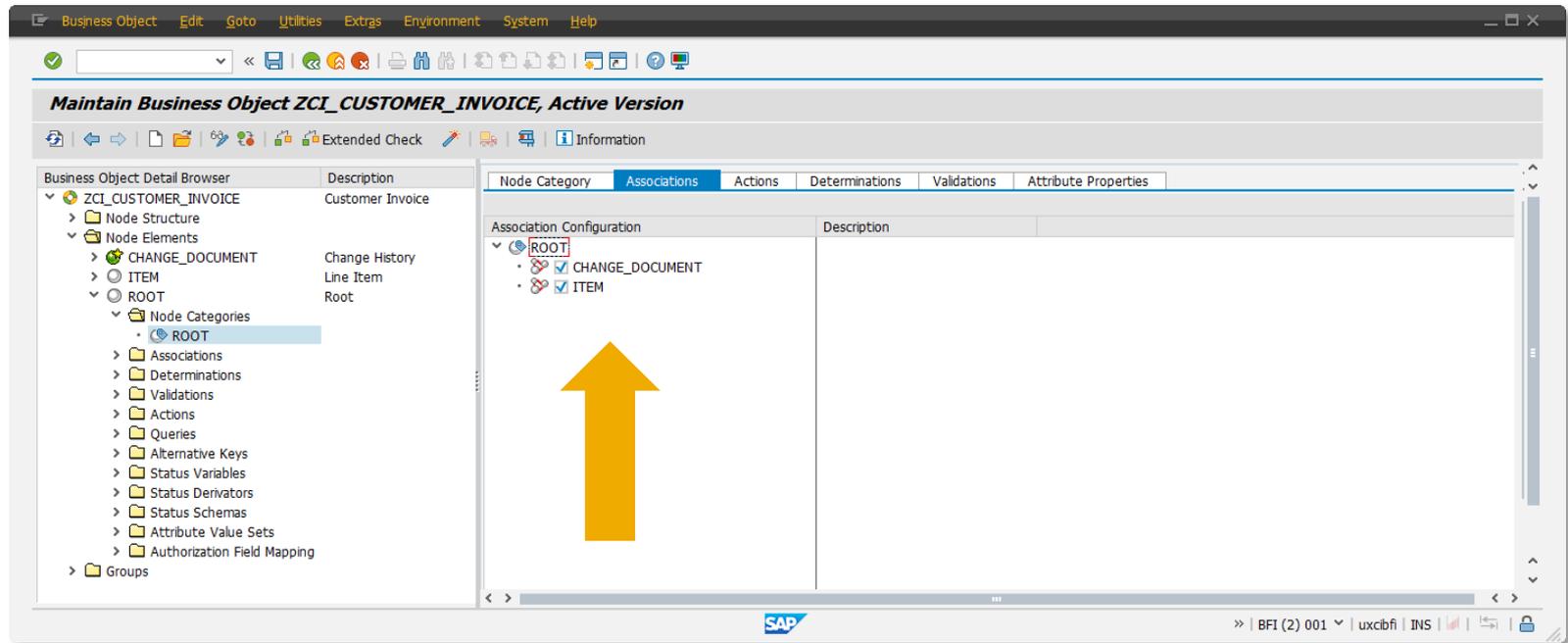
For Nodes and Actions



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

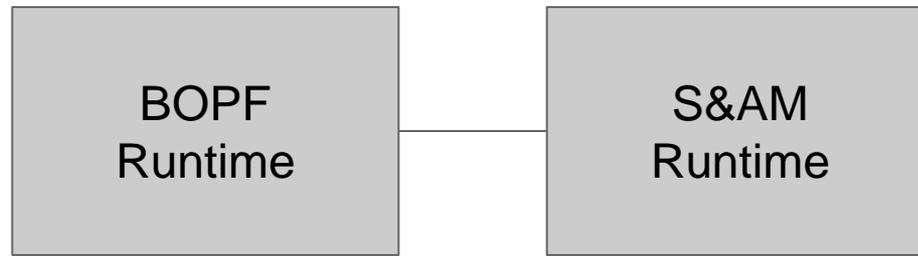
If a certain node instance is not locked optimistically or exclusively, BOPF sets the following properties automatically:

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

The image displays three overlapping screenshots from the SAP software interface, illustrating the configuration of a dynamic property determination.

- Top Left Screenshot:** Shows the 'Determination' configuration for 'SET_DYNAMIC_ROOT_PROPERTIES'. The description is 'Set dynamic root node properties'. Under 'Determination Settings', the 'Node' is set to 'ROOT', 'Determination Cat.' is 'Transient', 'Change mode' is 'Only Read Mode', and the 'Class/Interface' is 'ZCL_CI_D_ROOT_PROPERTIES'. There are checkboxes for 'Check method implemented' and 'Check Delta method implemented', both of which are checked.
- Top Right Screenshot:** Shows the 'Node Assignment to Determinations' table. The determination is 'SET_DYNAMIC_ROOT_PROPERTIES'. Under 'Request Nodes for Determination', the 'ROOT' node is selected with a radio button. Other nodes listed include 'ITEM', 'ROOT_LOCK', 'ROOT_MESSAGE', and 'ROOT_PROPERTY' (which is also selected with a radio button). The table has columns for 'Create', 'Upd...', 'Delete', 'Load', 'Det...', 'Modeled only', and 'Description'.
- Bottom Right Screenshot:** Shows the 'Determination Configuration (Node Cat.)' for 'SET_DYNAMIC_ROOT_PROPERTIES'. Under the 'ROOT' node, the 'Before Retrieve' trigger condition is selected with a radio button, and the 'Cleanup' checkbox is unchecked.

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.

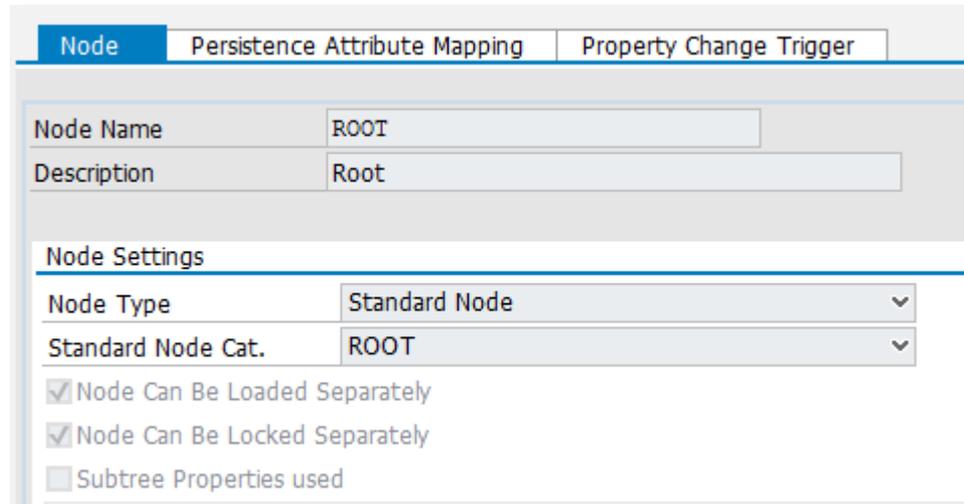
Use this class instead of modifying the property node.

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



The screenshot shows a configuration dialog for a node with three tabs: "Node", "Persistence Attribute Mapping", and "Property Change Trigger". The "Node" tab is active and contains the following fields:

Node Name	ROOT
Description	Root
Node Settings	
Node Type	Standard Node
Standard Node Cat.	ROOT
<input checked="" type="checkbox"/> Node Can Be Loaded Separately	
<input checked="" type="checkbox"/> Node Can Be Locked Separately	
<input type="checkbox"/> Subtree Properties used	

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 properties: 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.)

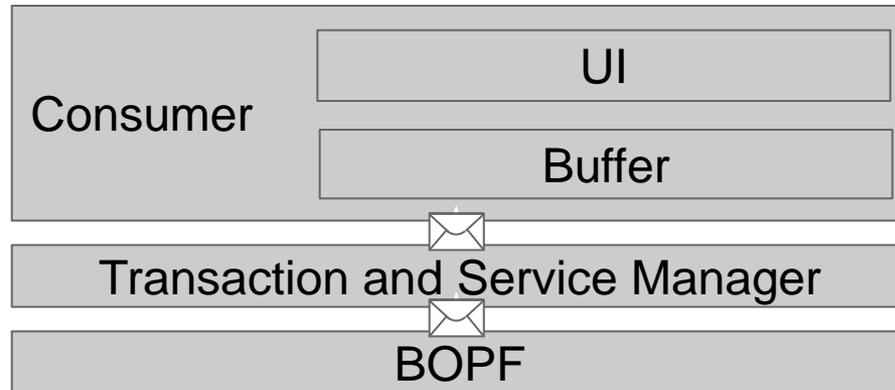
Examples

- 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

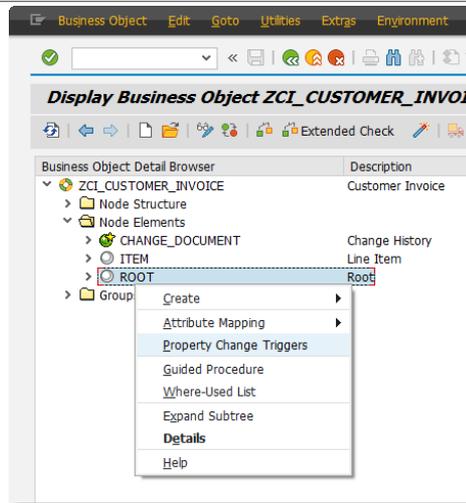
The consumer buffers all information received from BOPF in the buffer (also includes the properties returned by the `retrieve_property()` core service) → When the invoice is released (attribute “released” of the `ROOT` node is set to “X”) → Amount attributes on the item must be set to read-only in the 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



Node	Persistence Attribute Mapping	Property Change Trigger	Create	Upd...	Delete	Description
ROOT		Property Change Triggers				Root
ROOT		Property Change Triggers		<input checked="" type="checkbox"/>		Root
ITEM~TO_PARENT		ITEM~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Line Item
ITEM~TO_ROOT		ITEM~TO_ROOT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Line Item

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

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 is final
 Business Object can be enhanced
 Business Object is abstract
 Business Object is GenIL enabled

Business Object Settings

Root Node: ROOT
Buffer Class: /BOBF/CL_BUF_DISPATCHER
Constants Interface: ZIF_CI_CUSTOMER_INVOICE_C
Status Class:
Status Derivator:
Authorization Object:
Transactional Behaviour

Lock Behavior	Update Lock
Transaction Mode	One-Time Transaction (Delete Objects from Bu...
Check Modif. Serv.	Check against static properties
Check Assoc. Serv.	No Check
Check Action Serv.	No Check

Datenzugriff

Data Access Class: /BOBF/CL_DAC_TABLE

Test Data

Test Data Container:

- 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.

No Check (Obsolete!)
Check against static properties
Check against dynamic and static properties
No Check

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

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, PowerPoint, Silverlight, and Visual Studio are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, z10, z/VM, z/OS, OS/390, zEnterprise, PowerVM, Power Architecture, Power Systems, POWER7, POWER6+, POWER6, POWER, PowerHA, pureScale, PowerPC, BladeCenter, System Storage, Storwize, XIV, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, AIX, Intelligent Miner, WebSphere, Tivoli, Informix, and Smarter Planet are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the United States and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are trademarks or registered trademarks of Adobe Systems Incorporated in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and its affiliates.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems Inc.

HTML, XML, XHTML, and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Apple, App Store, iBooks, iPad, iPhone, iPhoto, iPod, iTunes, Multi-Touch, Objective-C, Retina, Safari, Siri, and Xcode are trademarks or registered trademarks of Apple Inc.

IOS is a registered trademark of Cisco Systems Inc.

RIM, BlackBerry, BBM, BlackBerry Curve, BlackBerry Bold, BlackBerry Pearl, BlackBerry Torch, BlackBerry Storm, BlackBerry Storm2, BlackBerry PlayBook, and BlackBerry App World are trademarks or registered trademarks of Research in Motion Limited.

Google App Engine, Google Apps, Google Checkout, Google Data API, Google Maps, Google Mobile Ads, Google Mobile Updater, Google Mobile, Google Store, Google Sync, Google Updater, Google Voice, Google Mail, Gmail, YouTube, Dalvik and Android are trademarks or registered trademarks of Google Inc.

INTERMEC is a registered trademark of Intermec Technologies Corporation.

Wi-Fi is a registered trademark of Wi-Fi Alliance.

Bluetooth is a registered trademark of Bluetooth SIG Inc.

Motorola is a registered trademark of Motorola Trademark Holdings LLC.

Computop is a registered trademark of Computop Wirtschaftsinformatik GmbH.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, SAP HANA, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.

Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase Inc. Sybase is an SAP company.

Crossgate, m@gic EDDY, B2B 360°, and B2B 360° Services are registered trademarks of Crossgate AG in Germany and other countries. Crossgate is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

The information in this document is proprietary to SAP. No part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP AG.