

BOPF Delegation Concept

SAP AG, 2012

The SAP logo is located in the bottom left corner of the slide. 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 white border on the right side.

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.

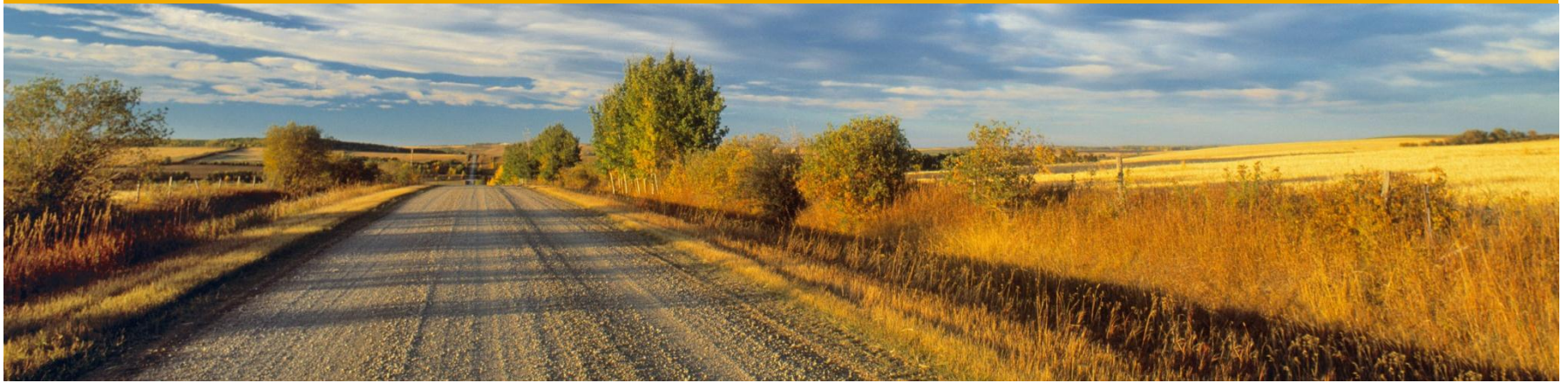
Agenda

Introduction

BOPF Delegation Concept

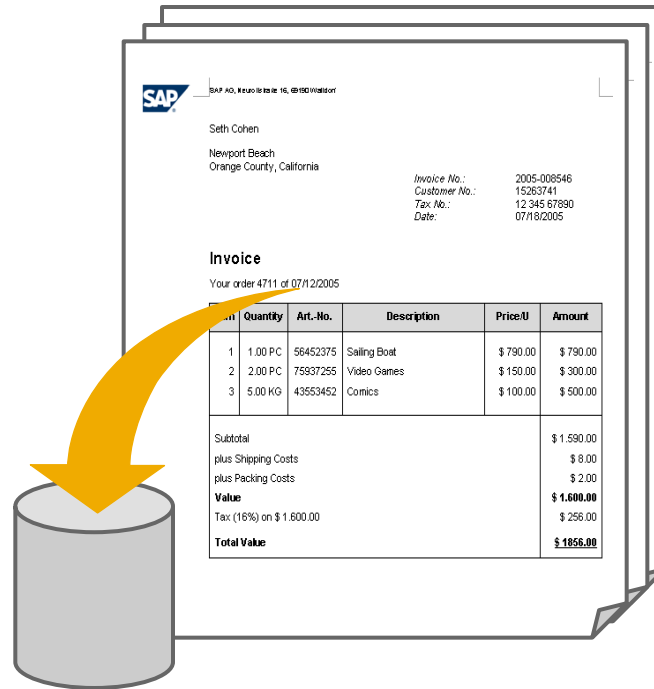
Use Case: Dependent Objects

Accessing Dependent Objects



Introduction

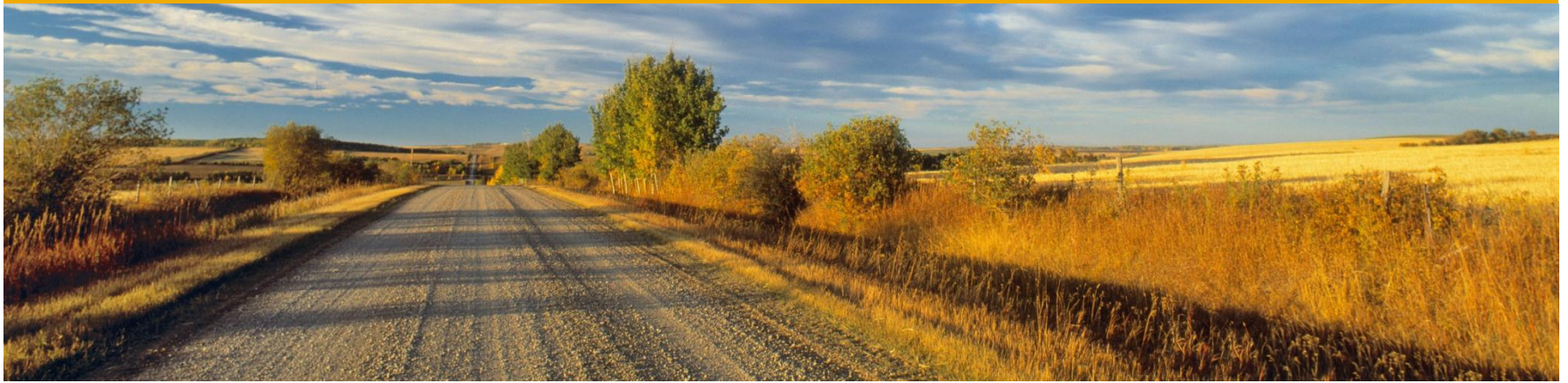
Introduction



The business object “customer_invoice” is to save all documents generated in addition to the invoice.

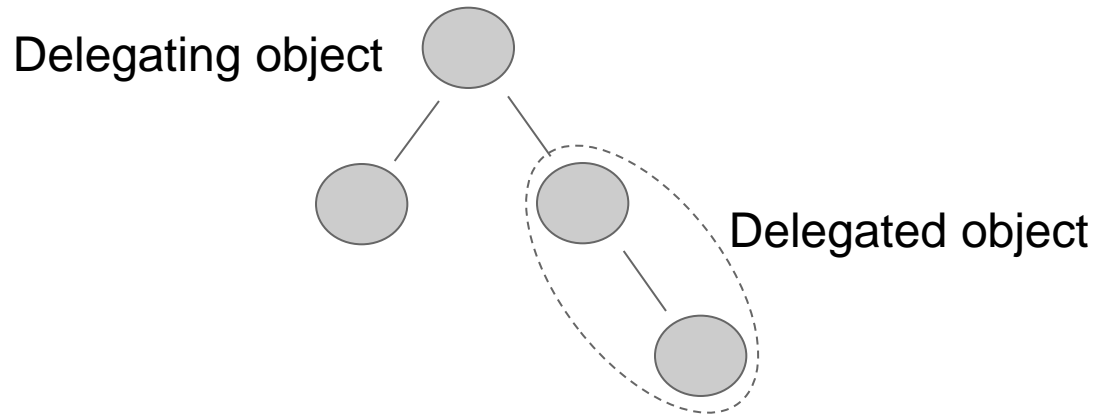
Many other business objects need this function too.

If parts of the logic of many business objects are redundant, it is advisable to extract this functional and to reuse it.



BOPF Delegation Concept

Overview



Principle: Parts of the model and functions are delegated to a separate object (delegated object).

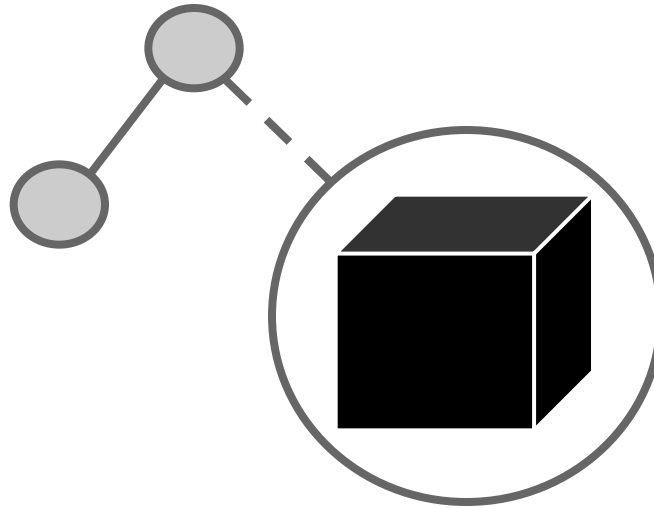
The delegating object can define additional constraints on the delegated part, e.g.:

- Overrule dynamic properties
- Perform additional validations
- Restrict dynamic value sets

This delegated object contains

- Its own business logic (e.g. actions, determinations, validations)
- Its own persistency, buffer handling, and delegations to delegated objects

Black Box Approach

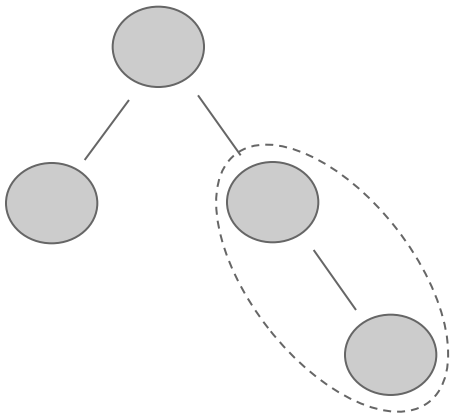


BOPF implements a Black Box Approach

- At design time, only a representation node of the delegated model exists.
- At runtime, the complete model is assembled and used.
- **Pro:** Model changes of the delegated object do not invalidate the model of the delegating object.
- **Con:** Modeling of additional logic in delegated parts not always possible.

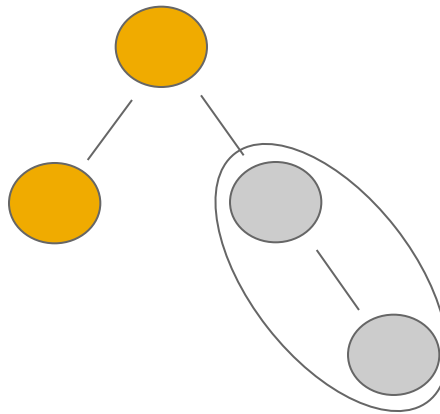
Viewpoints of the Delegation Concept

Service Consumer Model View



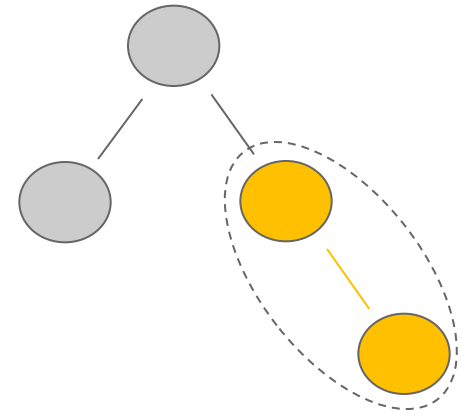
The delegation is not visible for the service consumer.

Delegating Object Model View



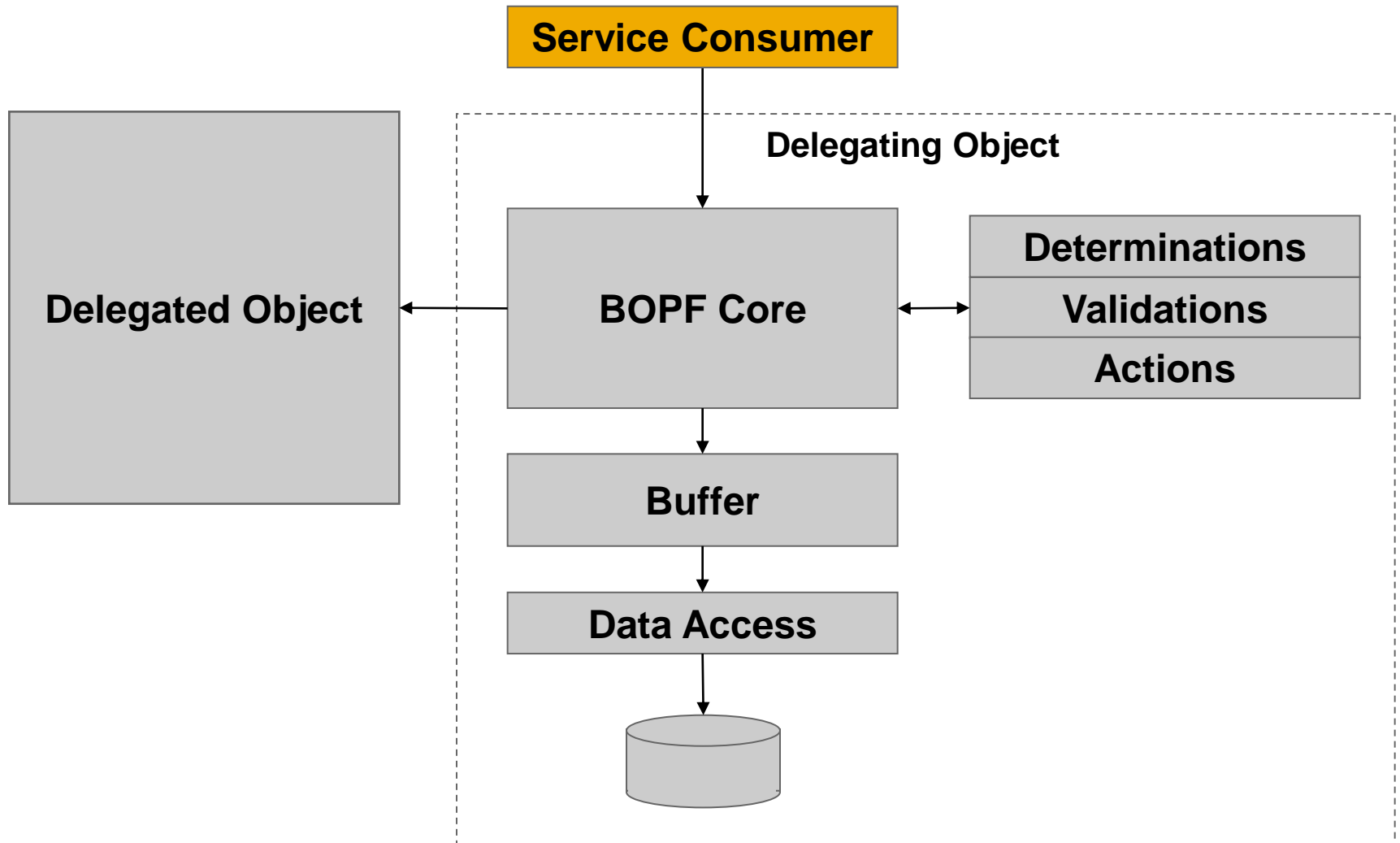
Only the delegation node is visible.

Delegated Object Model View

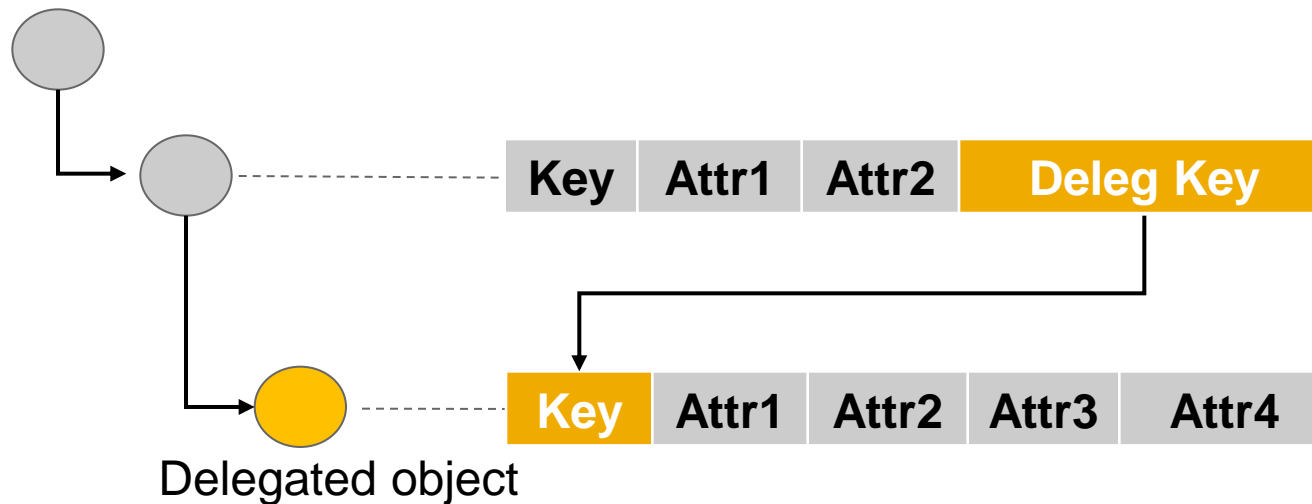


Only the internal structure of the delegated object is visible.

Architecture



Connection



Usually, the delegating object persists a foreign key of the delegated object:

- The key is created by the delegated object and has to be unique.
- The key is usually an attribute of the hosting node (e.g. a GUID or ID) but it can also be the technical key of the hosting node (e.g. `KEY`).



Use Case: Dependent Object

Overview

The BOPF delegation concept is used to include dependent objects.

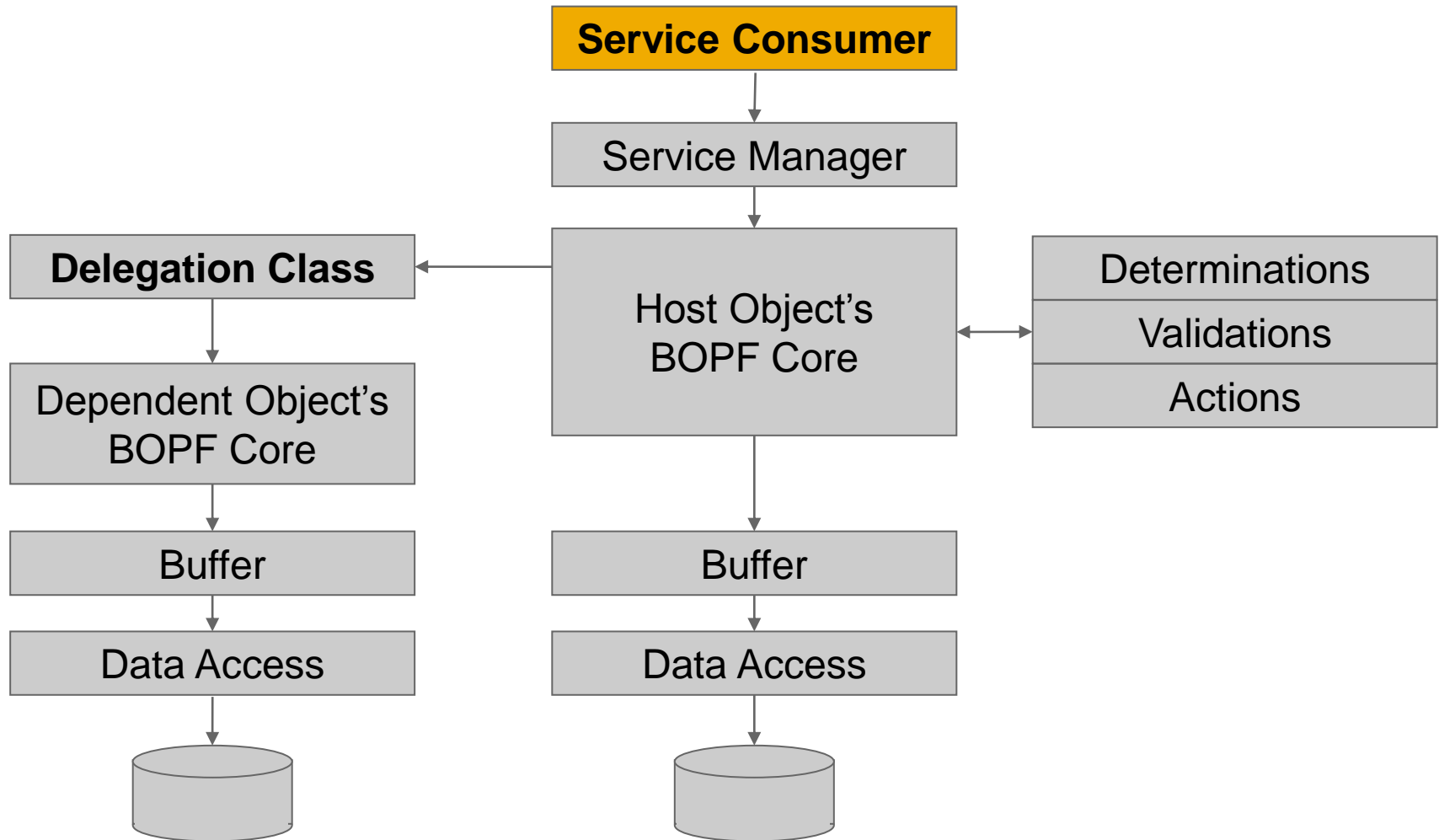
Naming

- Delegating object is called “Host Object” (HO)
- Delegated object is called “Dependent Object” (DO)

Dependent objects rules

- Only host objects are allowed to access dependent objects.
- A dependent object can also be a host object containing another dependent object.
- Each dependent object instance belongs to exactly one host object.
- The dependent object cannot call its own host object.
- All model entities of the dependent object are visible for service consumer of the host object, except for queries
- All nodes of the Dependent Object are locked always by the Host Object

Architecture



Delegated Nodes

The diagram illustrates a node structure for a 'Demo Customer' application. A tree view on the left shows a hierarchy starting with 'ROOT' under '/BOFU/DEMO_CUSTOMER'. A yellow oval highlights a subtree where 'ATTACHMENT_FOLDER' is a child of 'ROOT', and 'ADDRESS' is a child of 'ATTACHMENT_FOLDER'. A yellow arrow points to the 'ATTACHMENT_FOLDER' node in the tree. To the right, a configuration window for a 'Node' shows the following settings:

Node	
Node Name	ATTACHMENT_FOLDER
Node Prefix	ATF
Description	ATTACHMENT FOLDER
Node Settings	
Node Type	Delegated Node
Data Model	
<input type="checkbox"/> Key is equal to parent key	
Buffer	
Delegation Class	/BOBF/CL_LIB_DELEGATION_BOBF
Authorization Checks	
Active	<input type="checkbox"/>
Check Class	
Cross Business Object Relationship	
Ref. Business Object	/BOBF/ATTACHMENT_FOLDER

Delegated nodes are placeholders for dependent objects.

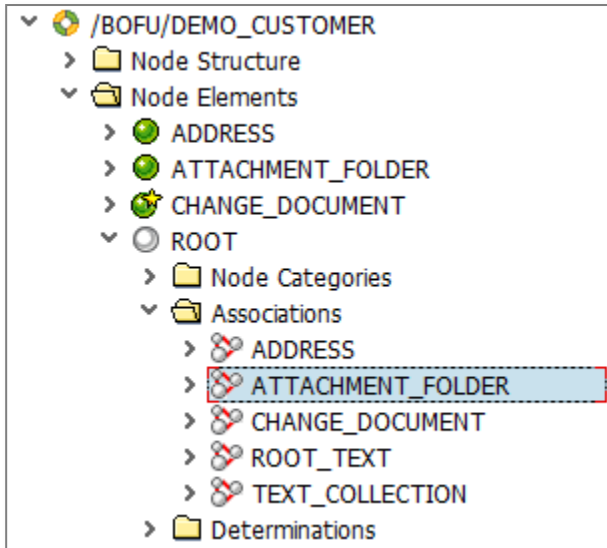
These nodes are modeled and can be used within the configuration like normal nodes.

During runtime, these nodes are replaced by the complete structure of the dependent object.

Delegated nodes can also be assigned to a consistency group.

Only internal calls can perform queries on delegated nodes. These calls are delegated to the dependent object.

Connection Between Delegated Node and Delegation Object



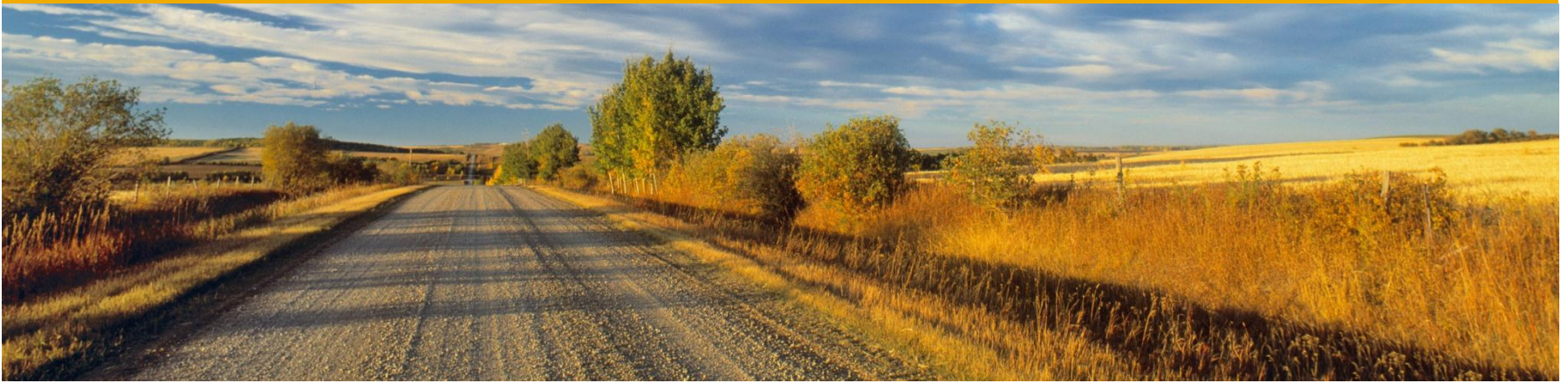
A screenshot of the 'Association' configuration dialog box. The dialog has three tabs: 'Association', 'Association Binding', and 'Property Change Trigger'. The 'Association' tab is active. The 'Association Name' is 'ATTACHMENT_FOLDER'. The 'Description' field is empty. The 'Association Settings' section contains the following fields:

Association Settings	
Association Type	Composite
Association Category	Association to delegated object
Cardinality	Cardinality 1:0...1
Source Node	ROOT
Source Attribute	ATF_KEY
Associated Node	ATTACHMENT_FOLDER
Resolving Node	Resolve by Source Node
Assoc. Change	written by Buffer

The 'Implementation' section contains the following fields:

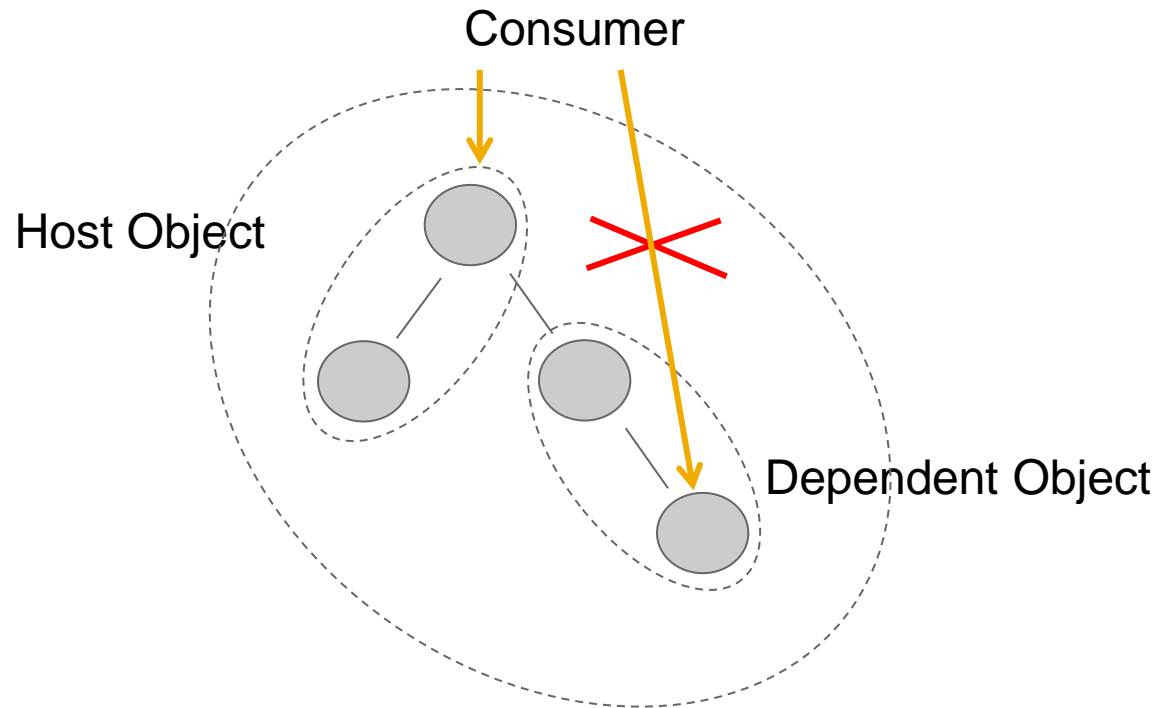
Implementation	
Association Class	/BOBF/CL_LIB_C_DO_ATTACHMENT
Parameter Structure	

All associations from and to the delegated object are only known by the delegating object. Since the buffer layer cannot provide these associations, they have to be implemented. These associations are usually foreign key associations.



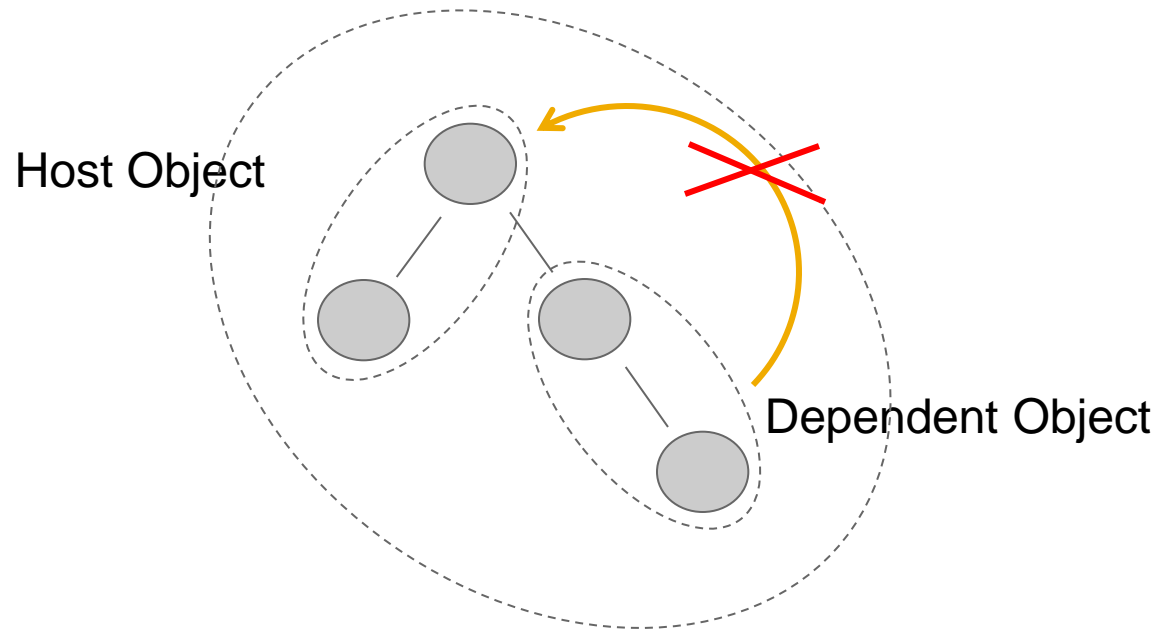
Accessing Dependent Objects

Using Dependent Object Entities as a Consumer



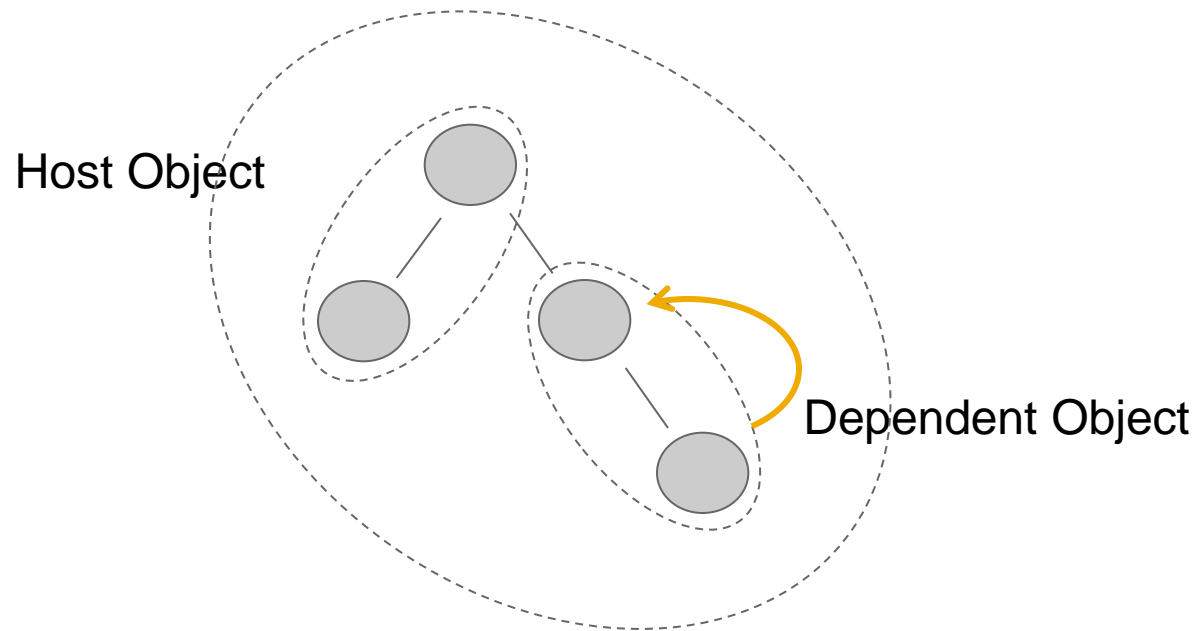
The consumer must only access the host object via the service manager. This service manager offers access to all entities (e.g. actions) including those from the dependent object.

Using Host Object Entities as a Dependent Object Entity



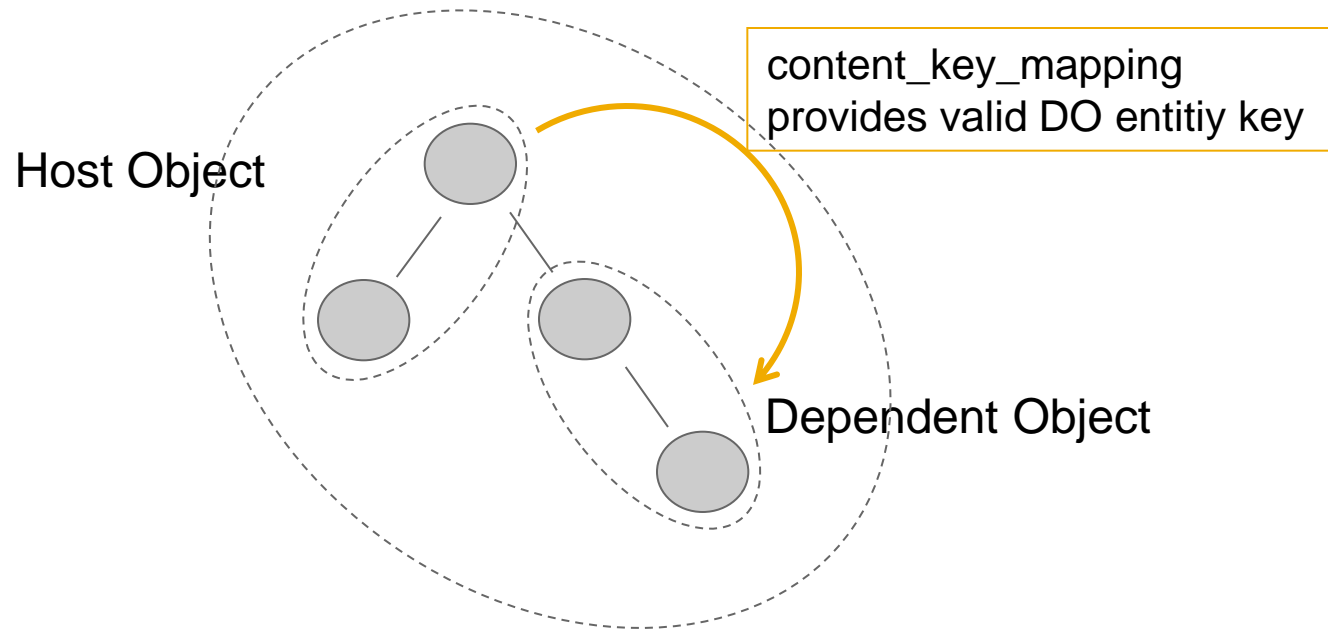
Entities of the dependent object must not use entities from the host object. This is because each host object could have a different set of entities. For instance, invoking a certain action on the host object root node only works for a certain host object.

Using Dependent Object Entities as a Dependent Object Entity



The dependent object entities may access entities from the dependent object by the help of the dependent object's constant interface, just by using `io_read` and `io_modify`.

Using Dependent Object Entities as a Host Entity



In order to access an entity of the dependent object out of an entity implementation (e.g. action) of the host object, its key must be first mapped.

This is because the same dependent object could be included multiple times in the same host object. However each node of a business object must be unique. Thus at runtime, a mapping is needed.

Content Key Mapping

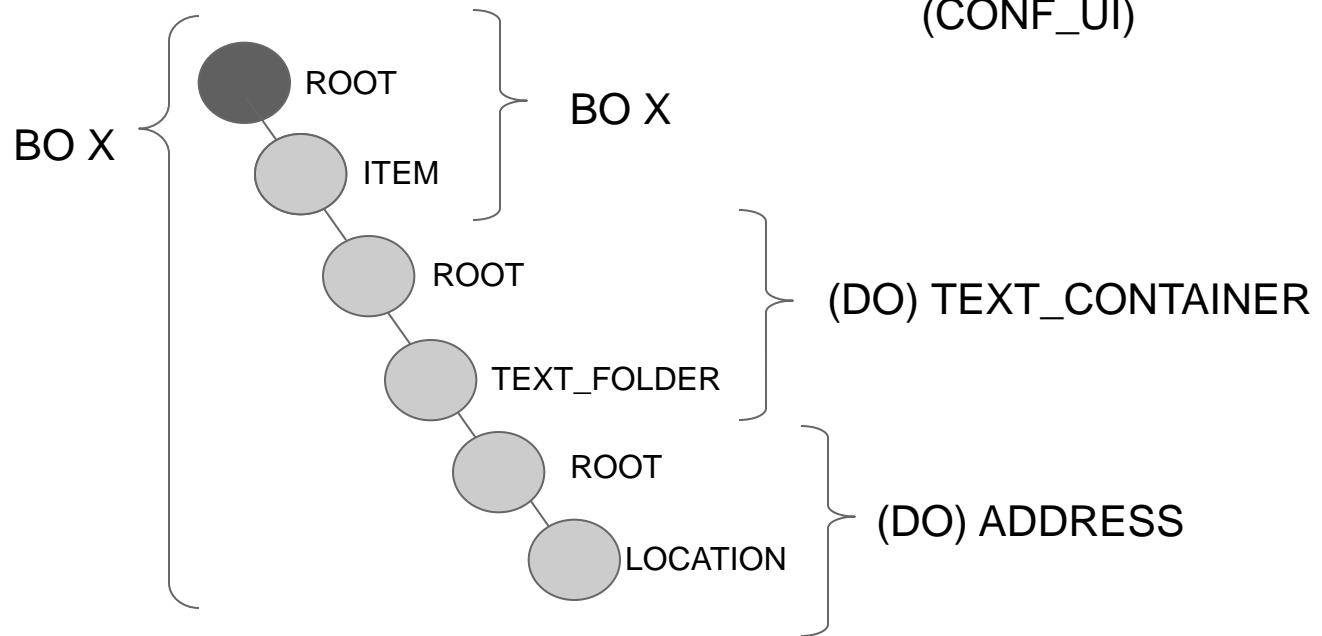
Use the method `GET_CONTENT_KEY_MAPPING` of the configuration of the host object in order to receive the valid key of an entity of the dependent object:

- **IV_CONTENT_CAT**
(E.g. `/BOBF/IF_CONF_C=>SC_CONTENT_ACT` for an action
`/BOBF/IF_CONF_C=>SC_CONTENT_ASS` for an association
`/BOBF/IF_CONF_C=>SC_CONTENT_NOD` for a node)
- **IV_DO_CONTENT_KEY**
Key of the entity out of the DO's constant interface
- **IV_DO_ROOT_NODE_KEY**
Key of the representation node of the Host Object's constant interface
- **EV_CONTENT_KEY**
Key of the entity, which can be used to address this entity by the help of core services like `retrieve`, `do_action`, `retrieve_by_association`...

Example Use Case: Usage of two Dependent Objects

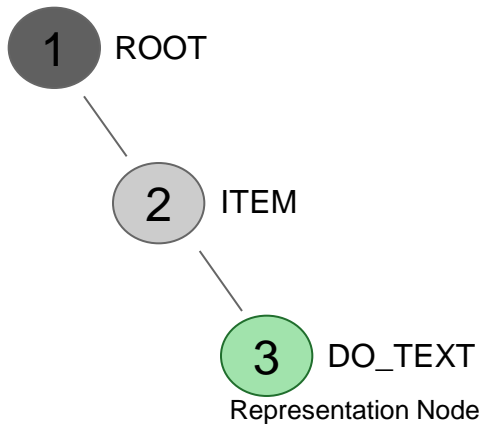
Runtime Viewpoint

Designtime Viewpoint
(CONF_UI)



Designtime Viewpoints of the a Business Object and its Dependent Objects

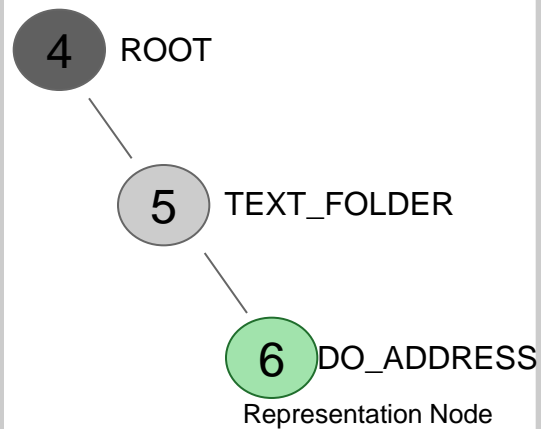
BO Customer Invoice



IF_CONST_CUST_C

ROOT = „1“
ITEM = „2“
DO_TEXT = „3“

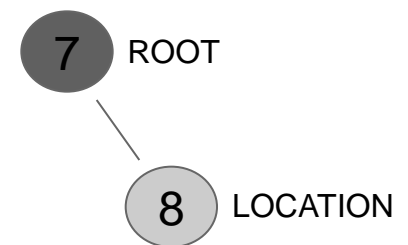
DO Text Container



IF_CONST_TEXT_C

ROOT = „4“
TEXT_FOLDER = „5“
DO_ADDRESS = „6“

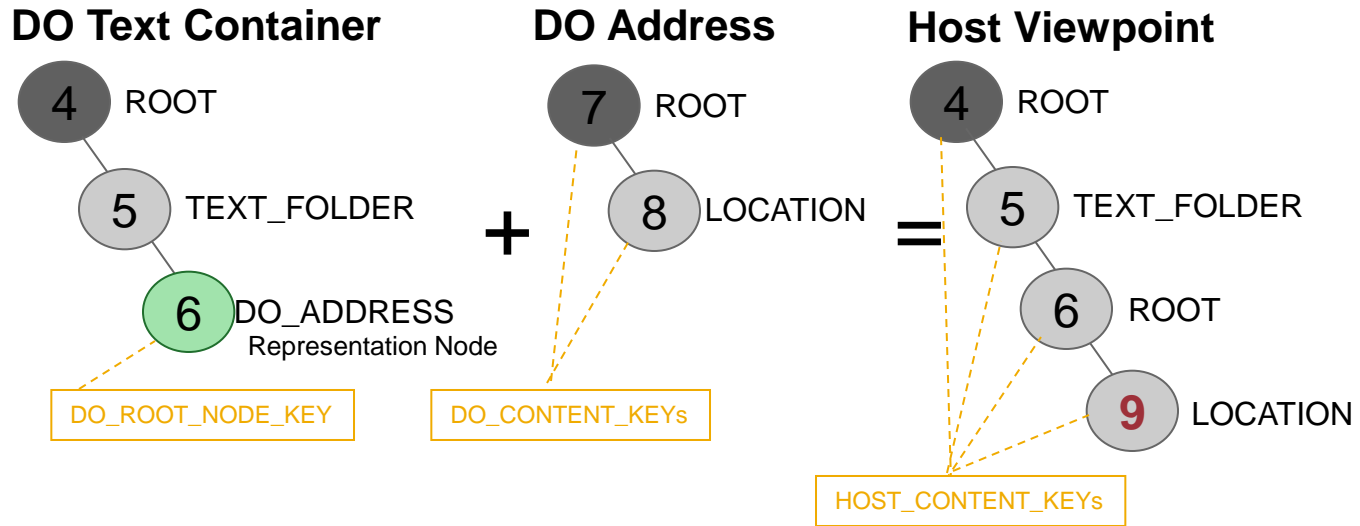
DO Address



IF_CONST_ADDRESS_C

ROOT = „7“
LOCATION = „8“

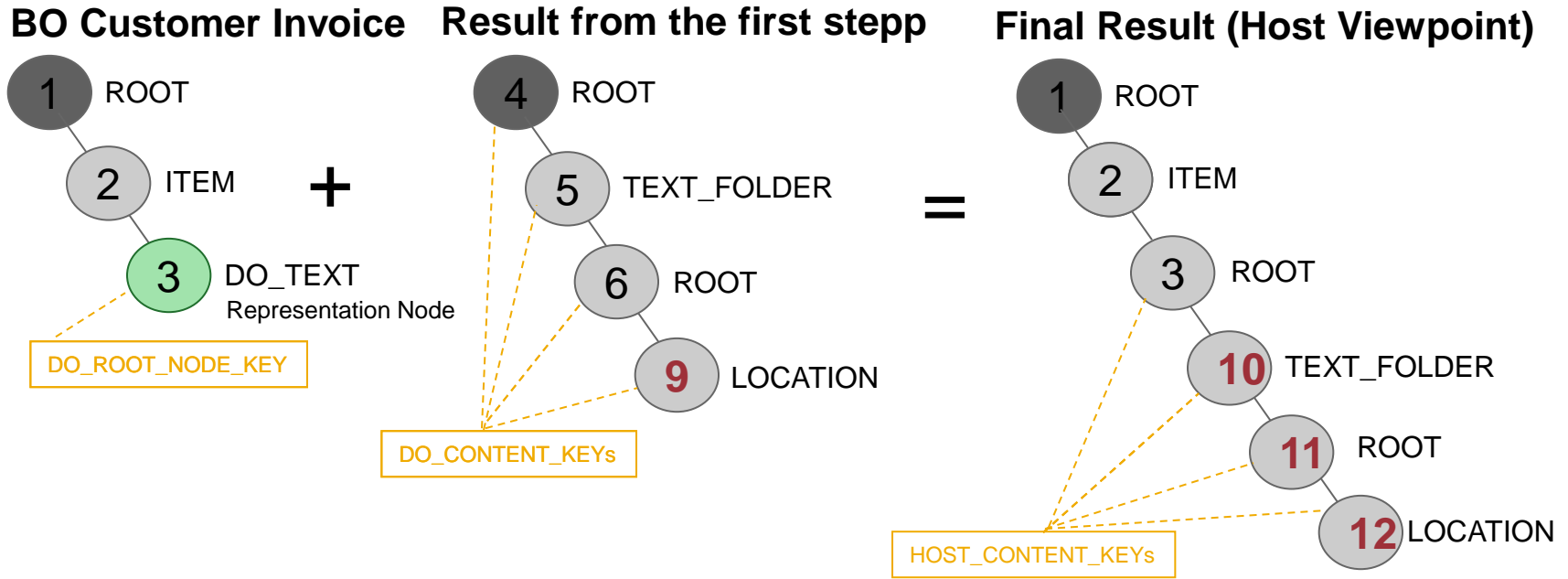
Creating the Content Key Mapping Table (1)



Table, used by get_content_key_mapping of the configuration object of DO Text Container:

7 ROOT:	
CONTENT_CAT	Node
HOST_CONTENT_KEY	6 DO_ADDRESS
DO_CONTENT_KEY	7 ROOT
DO_ROOT_NODE_KEY	6 DO_ADDRESS
8 LOCATION:	
CONTENT_CAT	Node
HOST_CONTENT_KEY	9 (new)
DO_CONTENT_KEY	8 LOCATION
DO_ROOT_NODE_KEY	6 DO_ADDRESS

Creating the Content Key Mapping Table (2)

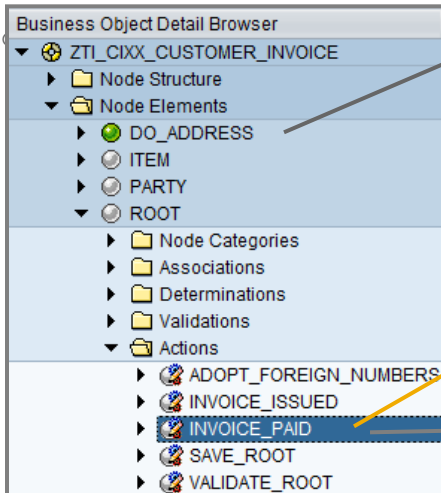


Table, used by get_content_key_mapping of the configuration object of BO customer invoice

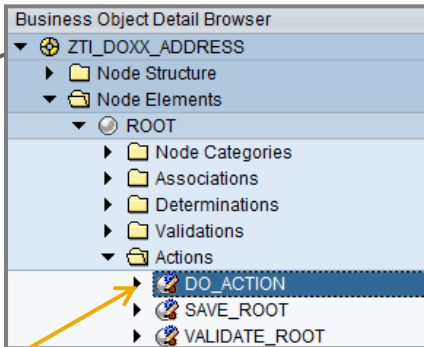
4 ROOT:		6 ROOT:	
CONTENT_CAT	Node	CONTENT_CAT	Node
HOST_CONTENT_KEY	3 DO_TEXT	HOST_CONTENT_KEY	11 (new)
DO_CONTENT_KEY	4 ROOT	DO_CONTENT_KEY	6 ROOT
DO_ROOT_NODE_KEY	3 DO_TEXT	DO_ROOT_NODE_KEY	3 DO_TEXT
5 TEXT_FOLDER:		9 Location	
CONTENT_CAT	Node	CONTENT_CAT	Node
HOST_CONTENT_KEY	10 (new)	HOST_CONTENT_KEY	12 (new)
DO_CONTENT_KEY	5 TEXT_FOLDER	DO_CONTENT_KEY	9 LOCATION
DO_ROOT_NODE_KEY	3 DO_TEXT	DO_ROOT_NODE_KEY	3 DO_TEXT

Example of Content Key Mapping

Host Object



Dependent Object



Scenario:

Action INVOICE_PAID of the HO calls action DO_ACTION of the DO

There to the DO's action key must received by the help of content key mapping first

Finally the DO's action can be called

```
data:
  lo_conf      type ref to /bobf/if_frw_configuration,
  lv_do_action_key type /bobf/act_key,
  lt_key       type /bobf/t_frw_key.

" 1. get configuration
try.
  lo_conf = /bobf/cl_frw_factory=>get_configuration( iv_bo_key = zti_cixx_customer_invoice_c=>sc_bo_key ).
catch /bobf/cx_frw.
endtry.

" 2. get key of the do's action by the help of content key mapping
lo_conf->get_content_key_mapping(
  exporting
    iv_content_cat      = /bobf/if_conf_c=>sc_content_act
    iv_do_content_key   = zti_doxx_address_c=>sc_action-root-do_action
    iv_do_root_node_key = zti_cixx_customer_invoice_c=>sc_node-do_address
  receiving
    ev_content_key      = lv_do_action_key ).

" 3. call the do's action
io_modify->do_action(
  exporting
    iv_act_key   = lv_do_action_key
    it_key       = lt_key ).
```



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.