

Release 4.0, SP04
November 2003
English

Individual Objects in the Interaction Center « Boat Example »

Cookbook

SAP AG
Neurottstr. 16
69190 Walldorf
Germany

Copyright

© Copyright 2003 SAP AG. All rights reserved.

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®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation.

INFORMIX®-OnLine for SAP and INFORMIX® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

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

Citrix®, the Citrix logo, ICA®, Program Neighborhood®, MetaFrame®, WinFrame®, VideoFrame®, MultiWin® and other Citrix product names referenced herein are trademarks of Citrix Systems, Inc.

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

JAVA® is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPHIRE, Management Cockpit, mySAP, mySAP.com, 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 in several other countries all over the world. MarketSet and Enterprise Buyer are jointly owned trademarks of SAP Markets and Commerce One. All other product and service names mentioned are the trademarks of their respective owners.

Contents

Copyright	2
Cookbook: Individual Objects in the Interaction Center	4
1 Overview	4
1.1 Purpose of this Document	4
1.2 Overall Scenario	4
1.3 Required Steps	5
2 Modeling of the Individual Object "Boat"	5
2.1 Object Family	6
2.2 Creating Attributes	6
2.3 Creating Set Types	7
2.4 Creation of categories	9
2.5 Maintain Number Schemas for Categories	12
3 Create Screens Specific to the Individual Object (Boat)	12
3.1 Get_ui_data	14
3.2 Modify_Screen	15
3.3 Set_ui_Data	15
3.4 Top Include	16
3.5 Extend UI structure	16
4 IC Customizing	17
4.1 Maintain Application Area of Individual Objects	17
4.2 Create IC Organization and Assign IC Profile	18
5. BAdI Implementation	20
6. Defining an Alternative ID Type for Your Individual Object	21
6.1 Define ID Types	21
6.2 Define ID Profiles	22
6.3 Assign ID Profile to Object Family	23
6.4 Using the Alternative ID in the IC	23
7. Customizing for Event History	24
7.1 History Overview	24
7.2 Event History	24
7.3 Integration With Business Transactions	25
7.4 Link Interpretation and Extract Handling	28
7.5 External Links	32
7.6 Building Extracts: Example of Business Transaction Integration	32

Cookbook: Individual Objects in the Interaction Center

1 Overview

1.1 Purpose of this Document

This document uses a short example to outline the most important steps that you need to complete before you can create and maintain your individual objects in the Interaction Center (IC SAP GUI).

1.2 Overall Scenario

The Company Boats International Inc. wants to use the Interaction Center (IC) to improve customer support for their sailing boats and motor boats.

Therefore they need to be able to:

- Quickly create boats in the IC when a customer calls
- Find customers boats by boat ID and partner function
- Maintain relationships between business partners and boats
- View the boat history
 - o Business transaction documents (such as activities and service orders)
 - o Attributes
- Maintain a simple configuration with the IBASE
- Manage documents for each boat
- Create business transactions with reference to a certain boat (activities and complaints, for example)

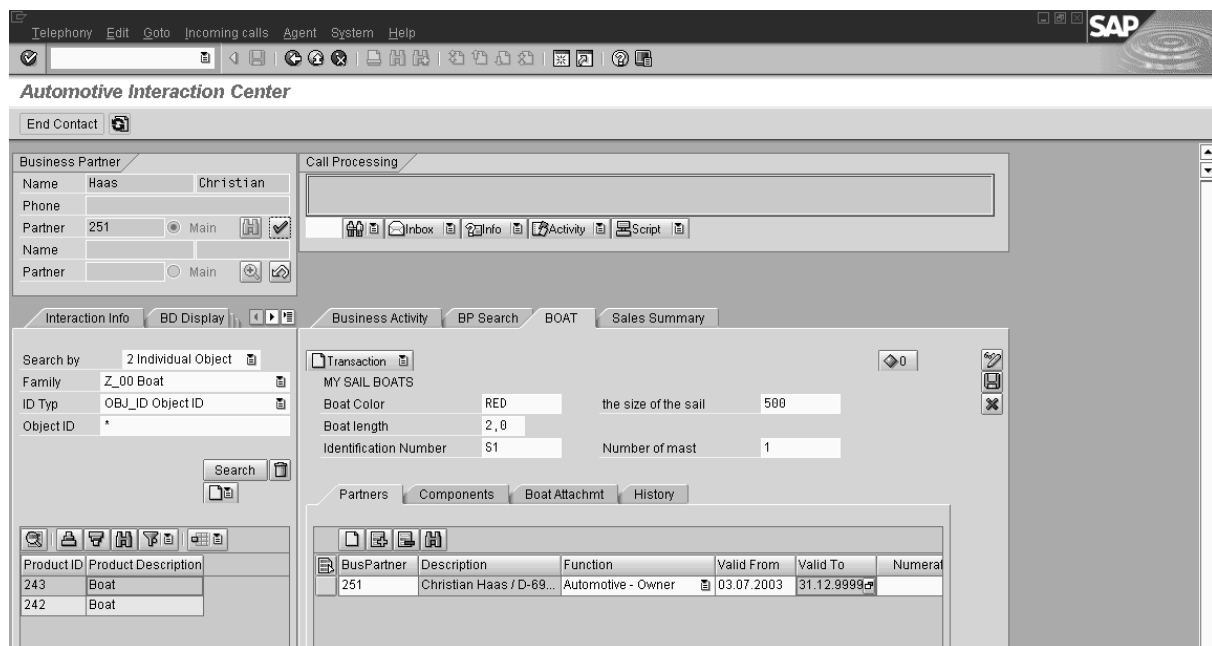


Figure 1 – Preview of what the "boat" IC will look like

1.3 Required Steps

1. Modelling of boats:
 - a. Create a new object family for boats
 - b. Create attributes
 - c. Create set type, assign attributes to set types
 - d. Create a boat-specific category
2. Create screens specific to your attributes
3. IC Customizing:
 - a. IC profile
 - b. Individual object workspace
 - c. BAdI implementation
 - d. IC organization structure

2 Modeling of the Individual Object “Boat”

Boats International Inc. has motor boats and sailing boats. Some of the attributes that describe a certain boat are specific to its type (motor vs. sailing boat), and other attributes are general. Therefore the attributes are grouped into set types:

The generic set type for boats contains the attributes that are valid for all types of boat:

- Color
- Length
- Boat identifier

The sailing boat specific set type contains the attributes that are specific to sailing boats:

- Number of mast
- Size of the sail

The motor boat specific set type contains the attributes that are specific to motor boats:

- Engine performance
- Fuel consumption

2.1 Object Family

You need to define a new object family for boats.

In the Implementation Guide, choose *Cross-Application Components -> SAP Product -> Individual Objects -> Define Object Families*.

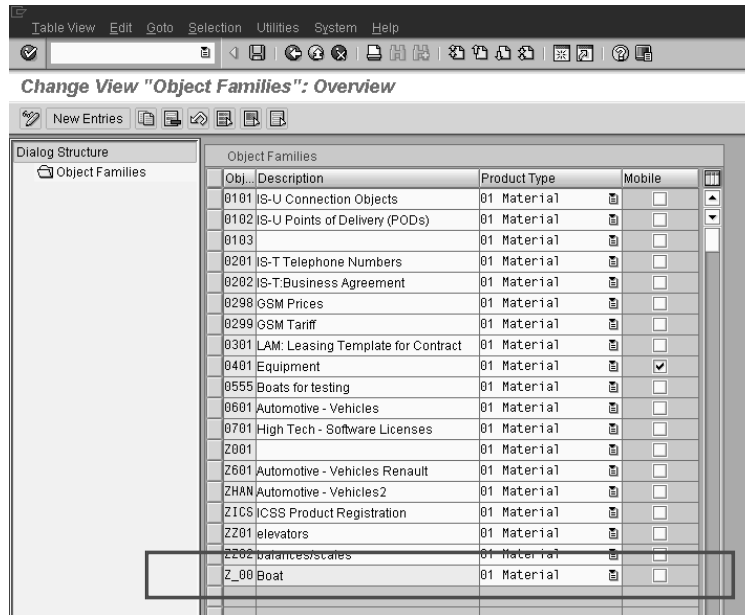


Figure 2 - Define Object Families

2.2 Creating Attributes

In the SAP Easy Access menu, choose *Master Data -> Products -> Maintain Set Types and Attributes* (transaction code: COMM_ATTRSET)

The screenshots show you how to set up the attribute ZBOAT_COLOR.

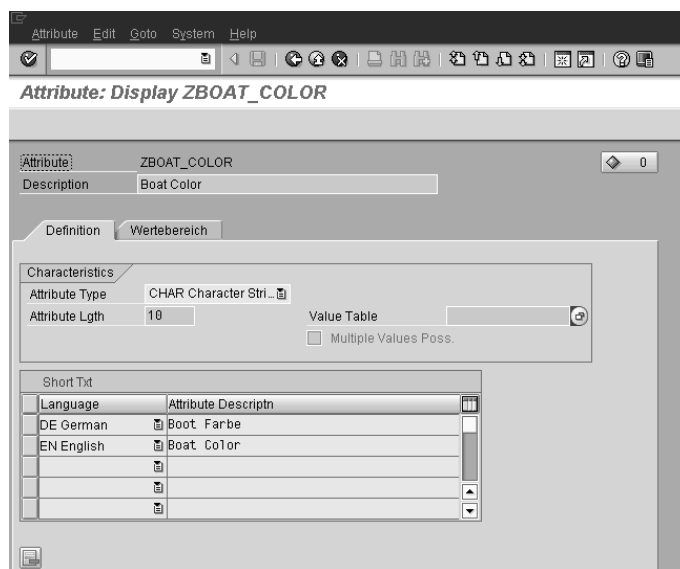


Figure 3 - Creating attributes (trx: COMM_ATTRSET)

Attribute: Display ZBOAT_COLOR

Attribute: ZBOAT_COLOR
Description: Boat Color

Definition Wertebereich

Fix. Values Allowed

Single Value ...	Upper Limit	Fixed Value Des.
BLACK		black
BLUE		blue
RED		red
WHITE		white

Sprachabhängige Festwertkurztexte

Single Value / ...	Upper Limit	Language	Fixed Value Description
BLACK		DE German	
BLACK		EN English	black
BLUE		DE German	
BLUE		EN English	blue
RED		DE German	

Entry 1 - 5 Fr. 8

Figure 4 - Creating attributes (trx: COMM_ATTRSET)

2.3 Creating Set Types

1. In the SAP Easy Access menu, choose *Master Data -> Products -> Maintain Set Types and Attributes* (transaction code: COMM_ATTRSET).

The screenshots show you how the attributes that are valid for all boats are assigned to the set type ZBOAT_GENERAL.

Set Type: Display ZBOAT_GENERAL

Set Type: ZBOAT_GENERAL
Description: General Boat Set Typ

Definition Assigned Attributes

Product Type Sele

Product Type ☒ Material ☐ Service ☐ Financing ☐ Intellectual Property

Characteristics

Org. Dependency 00 Independent of Organization

Multiple Usage ☐

Object Family

CDB Table

Short Txt

Language	Set Type Descr.
EN English	General Boat Set Typ

Figure 5 - Creating set types (COMM_ATTRSET)

2. Select the product type as material.
3. Leave the *Object Family* field empty.

- Do not select *Multiple Usage* if you want to use the set type for individual objects in the Interaction Center (IC) WinClient. **Multiple usage is currently not supported by the IC. Do not set the *Multiple Usage* indicator.**

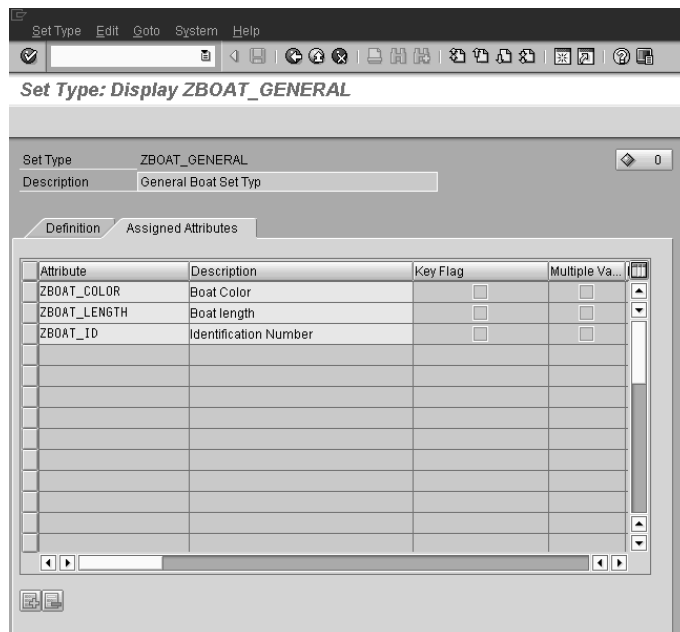
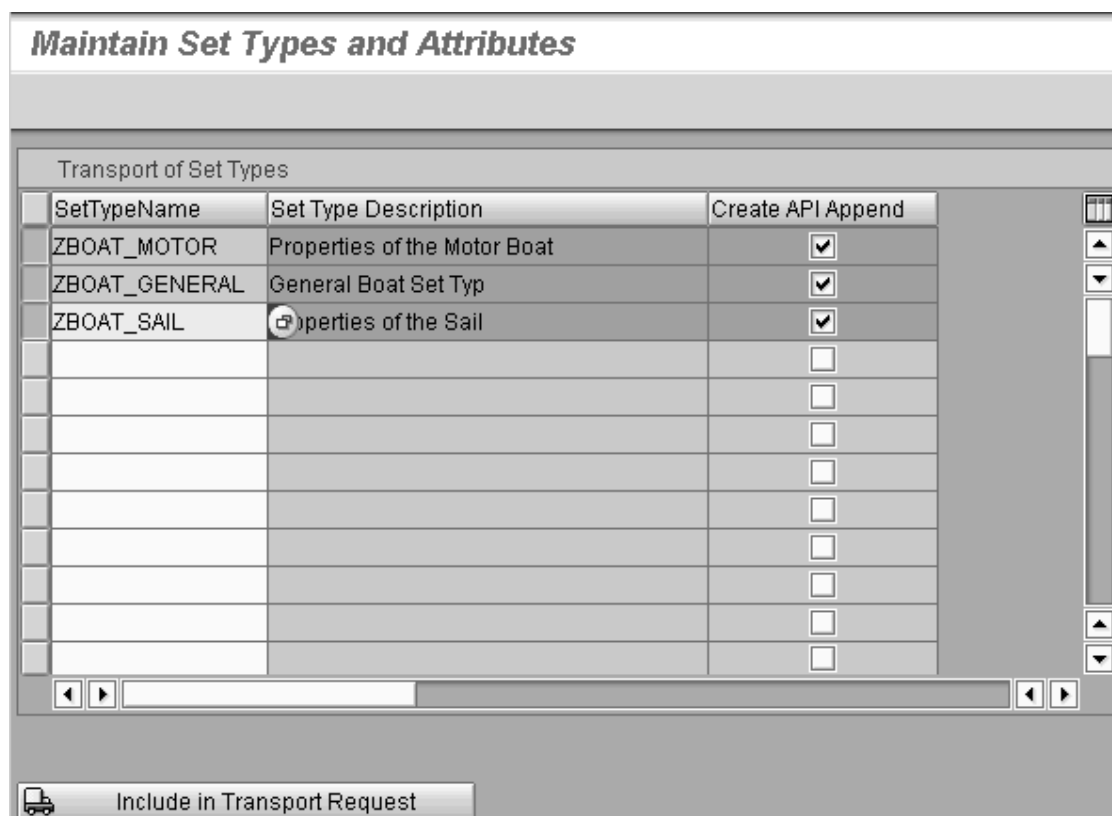


Figure 6 - Creating set types (COMM_ATTRSET)

Certain programs, screens, and data dictionary objects are created when you save or regenerate the set types.

You also need to transport the set types so that they get appended to the product API structure:

- In the menu, choose *Set Types/Attributes / Transport/ Set Type*.



- You can use transaction SE11 to check if your set types were successfully appended to the structure COMT_PROD_MAT_MAINTAIN_API:

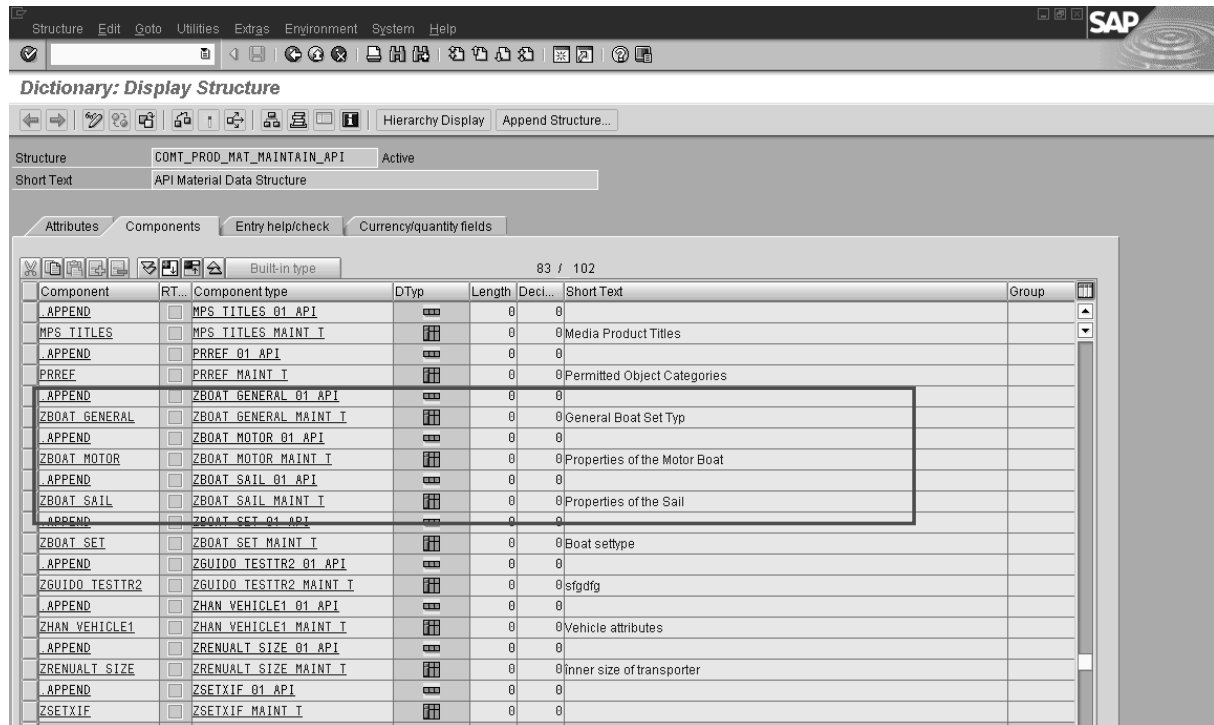


Figure 7 – Transaction SE11: Structure COMT_PROD_MAT_MAINTAIN_API

2.4 Creation of categories

In the SAP Easy menu, choose *Master Data -> Products -> Maintain Categories and Hierarchies* (transaction code: COMM_HIERARCHY).

Three categories are created:

- ZBOAT
- ZBOAT_MOTOR
- ZBOAT_SAIL

They are linked as follows:

- ZBOAT
 - o ZBOAT_MOTOR
 - o ZBOAT_SAIL

And they are set up as follows:

Parameter		Category		
		ZBOAT	ZBOAT_MOTOR	ZBOAT_SAIL
Category	Product Type	Material	Inherited	Inherited
	Object Family	Z_00	Inherited	Inherited
	Documents Maintainable	X	Inherited	Inherited
	Conditions Maintainable	X	Inherited	Inherited
	Partner Determination Procedure	-	00000035	00000035
	Alternative ID Type	-	-	-

Set types	Set types	ZBOAT_GENERAL (if you want other set types like COMM_PR_*)	Inherited ZBOAT_MOTOR	Inherited ZBOAT_SAIL
Relship types		-	-	-

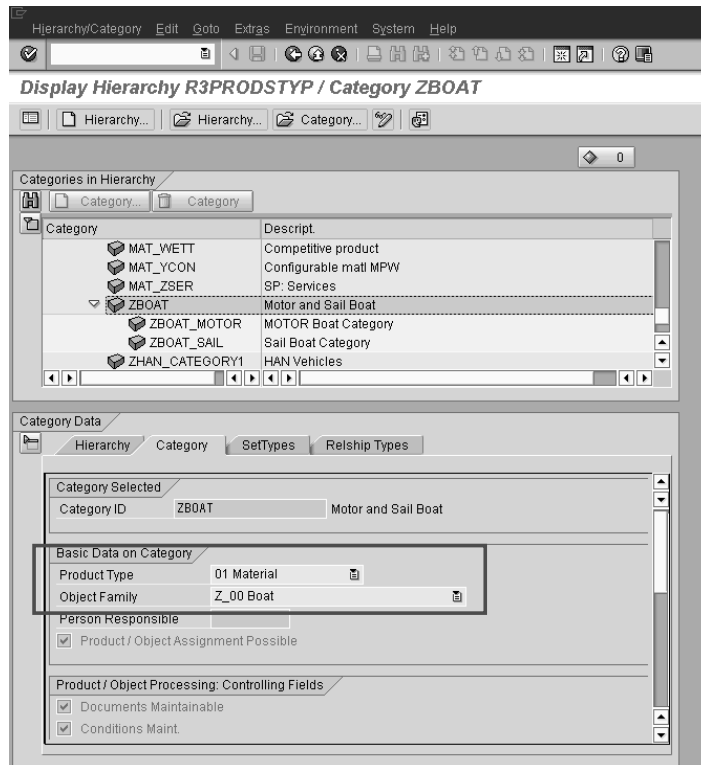


Figure 8 - Category "ZBOAT" in hierarchy R3PRODSTYP [trx: comm_hierarchy]

Go to the hierarchy for which you want to enter your new category. Choose the hierarchy according to which set types you want to inherit.

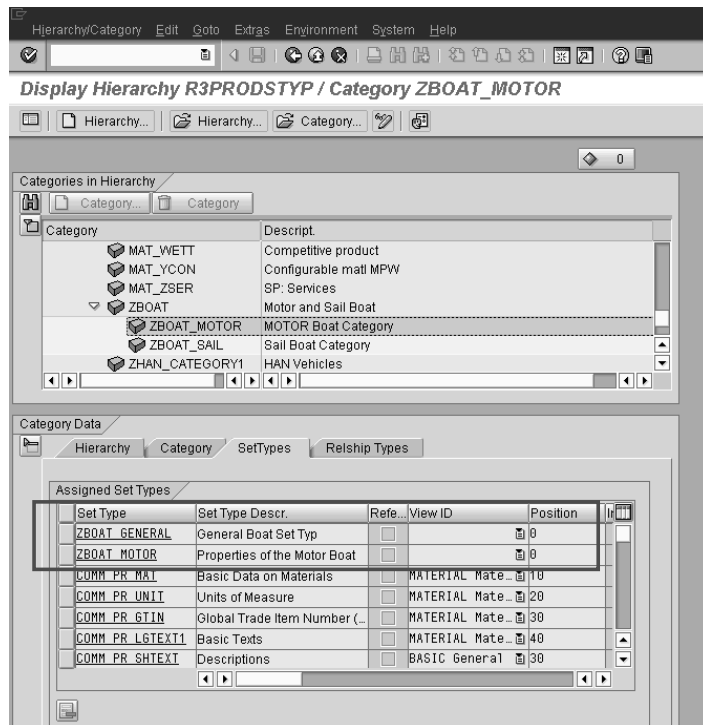


Figure 9 - Category "ZBOAT_MOTOR"

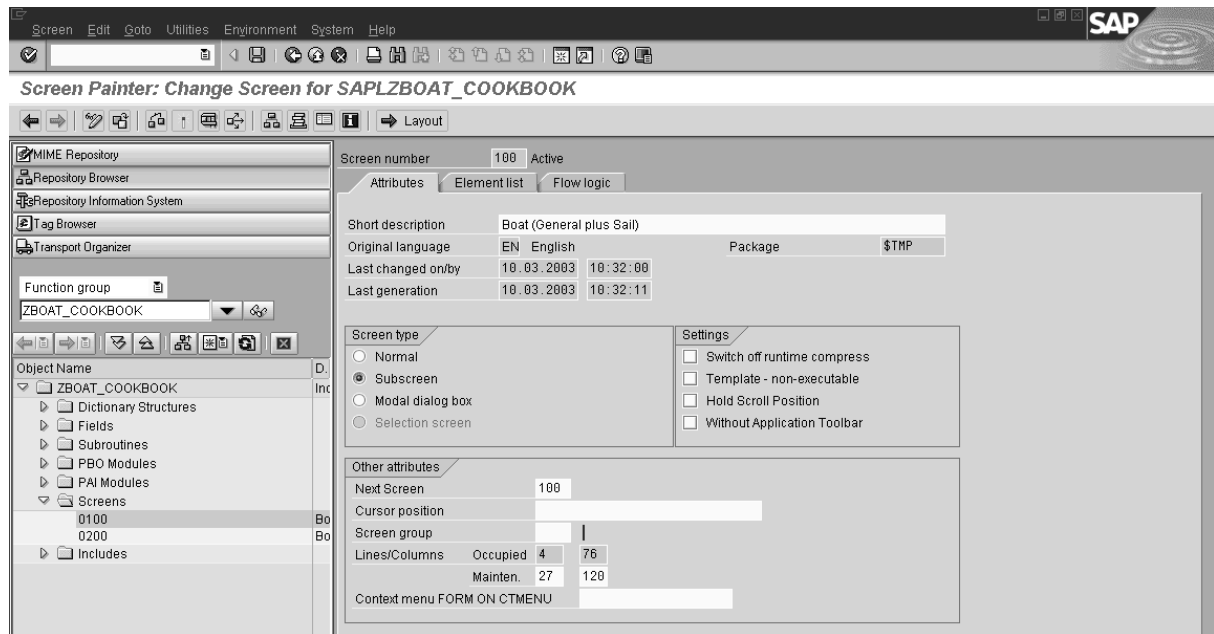


Figure 11 - Screen for sailing boats

- Use the data elements from the table <Your Set Type Name>_DYNP. In the example this is ZBOAT_GENERAL_DYNP. Note that <Your Set Type Name>_DYNP is created by the system when you save <Your Set Type Name>.

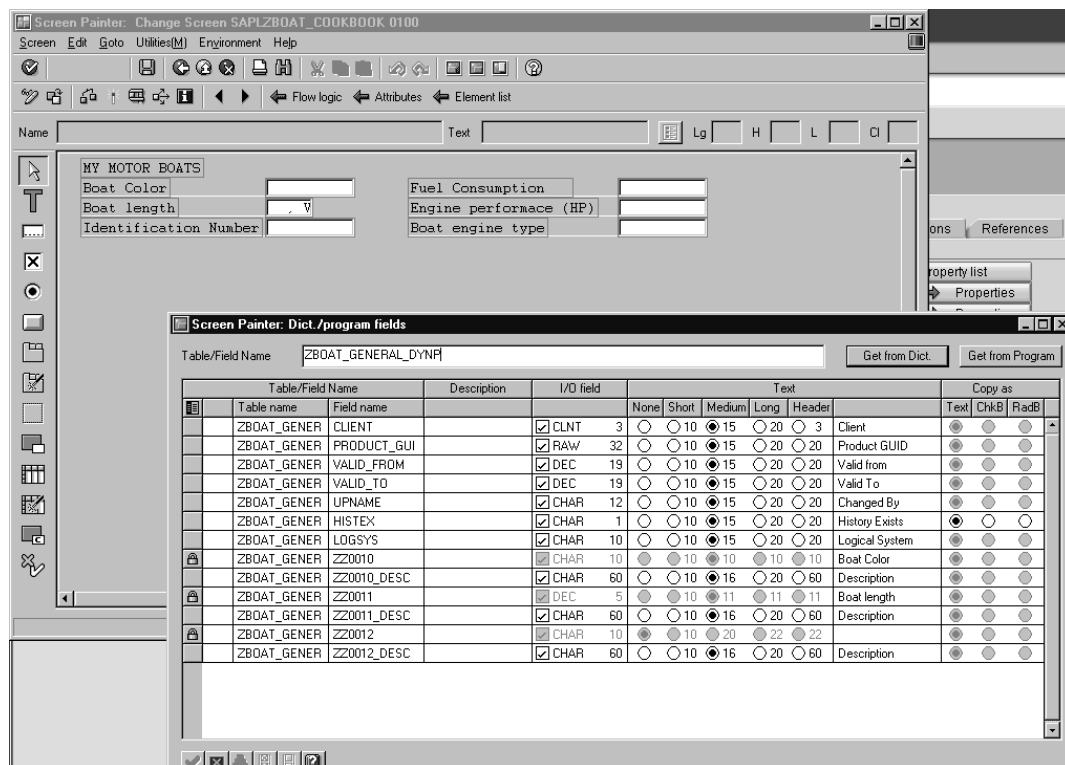


Figure 12 - Screen Painter

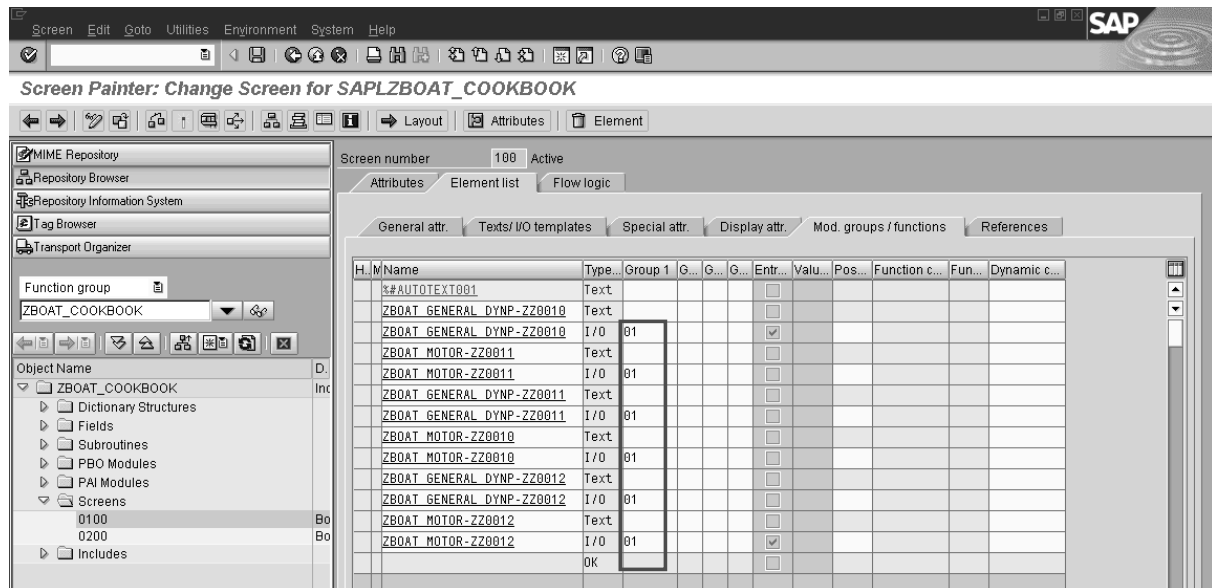


Figure 13 - Group Functions

4. Create the program code for the PBO and PAI:

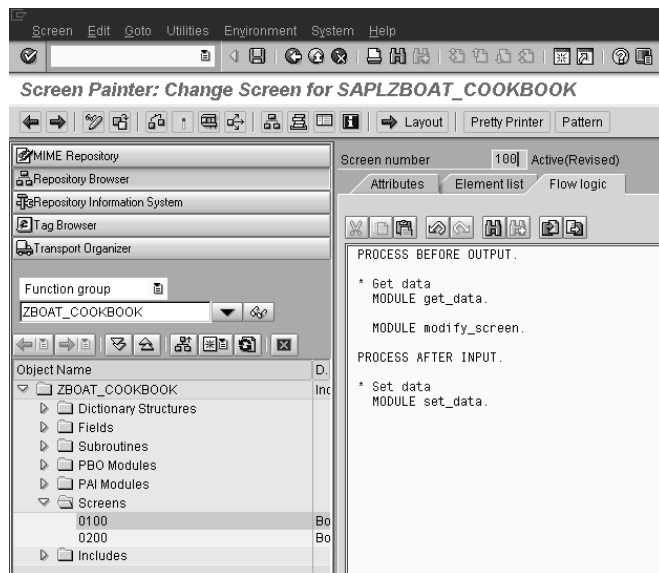


Figure 14 - Flow Logic

3.1 Get_ui_data

!! As of SAP CRM 4.0 Support Package 02, use the same PBO and PAI logic as in screen 201 of function module CRM_IOBJECT_CIC_IOBA. Cut and paste it from there. You only need to exchange the ISAM set types with your set types in the forms GET_UI_DATA and SET_UI_DATA.

FORM get_data .

* UI structures (View context)

CALL FUNCTION 'CRM_IOBJECT_CIC_IOBA_UI_GET'
IMPORTING

ET_IOBJ_CIC_UI = gv_iobj_ui. "data used locally in the
"function group of the screens

```

* Move data from controller context to view context
ZBOAT_MOTOR_DYNP = GV_IOBJ_UI-ZBOAT_Motor.
ZBOAT_SAIL_DYNP = GV_IOBJ_UI-ZBOAT_Sail.
ZBOAT_GENERAL_DYNP = GV_IOBJ_UI-ZBOAT_General.

```

```
ENDFORM.                " get_data
```

3.2 Modify_Screen

```
FORM modify_screen.
```

```

* ----- *
* Get process mode
DATA: lv_process_mode TYPE crmt_mode VALUE IS INITIAL.
CALL FUNCTION 'CRM_IOBJECT_CIC_PROC_MODE_GET'
  IMPORTING
    ev_process_mode = lv_process_mode.
* ----- *
* Set UI data in create/change/display mode as required
IF LV_PROCESS_MODE = GC_MODE-CREATE OR
  LV_PROCESS_MODE = GC_MODE-CHANGE .
* process mode is create or change
  LOOP AT SCREEN.
    IF SCREEN-GROUP1 = '01'.
      SCREEN-INPUT = '1'.
    ENDIF.
    MODIFY SCREEN.
  ENDLOOP.
ELSE.
* process mode is display
  LOOP AT SCREEN.
    IF SCREEN-GROUP1 = '01'.
      SCREEN-INPUT = '0'.
    ENDIF.
    MODIFY SCREEN.
  ENDLOOP.
ENDIF.

```

```
ENDFORM.                " modify_screen
```

3.3 Set_ui_Data

```
FORM set_data .
```

```

* Move data from controller context to view context
GV_IOBJ_UI-ZBOAT_Motor = ZBOAT_MOTOR_DYNP .
GV_IOBJ_UI-ZBOAT_Sail = ZBOAT_SAIL_DYNP .
GV_IOBJ_UI-ZBOAT_General = ZBOAT_GENERAL_DYNP .

* Set Individual Object data
CALL FUNCTION 'CRM_IOBJECT_CIC_IOBA_SET_DATA'
  EXPORTING
    it_iobj_cic_ui = gv_iobj_ui.

```

```
ENDFORM.                " set_data
```

3.4 Top Include

FUNCTION-POOL ZBOAT_COOKBOOK.

```
data:      GV_IOBJ_UI      TYPE CRMT_IOBJ_CIC_UI_2.
data:      GV_IOBJ_UI_BACKUP TYPE CRMT_IOBJ_CIC_UI_2.
```

```
tables: ZBOAT_MOTOR_DYNP,
        ZBOAT_SAIL_DYNP,
        ZBOAT_GENERAL_DYNP.
```

* Ind.Object process mode

CONSTANTS :

```
BEGIN OF gc_mode,
  create TYPE crmt_mode VALUE 'A',
  change TYPE crmt_mode VALUE 'B',
  display TYPE crmt_mode VALUE 'C',
END OF gc_mode.
```

3.5 Extend UI structure

Use transaction SE11 to append your set types to the structure CRMT_IOBJ_CIC_UI_2.

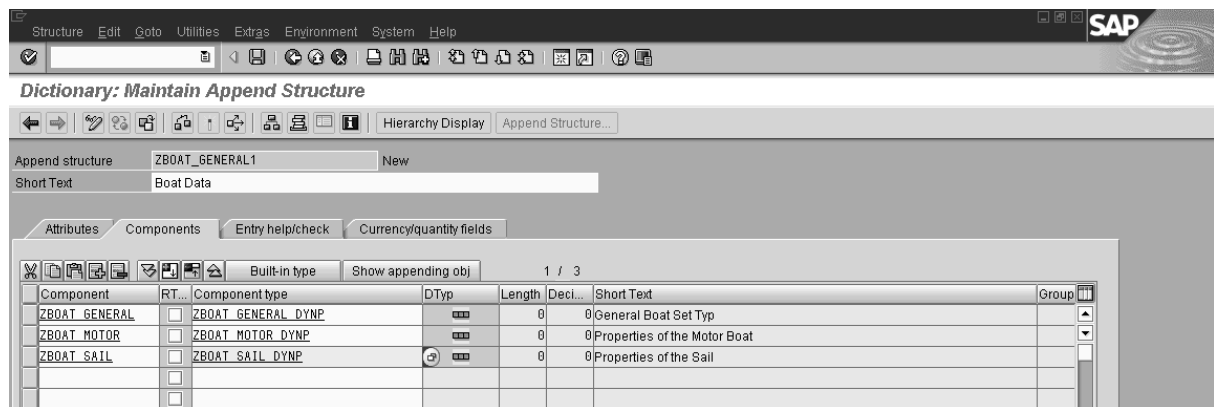


Figure 15 - Maintain Append Structure

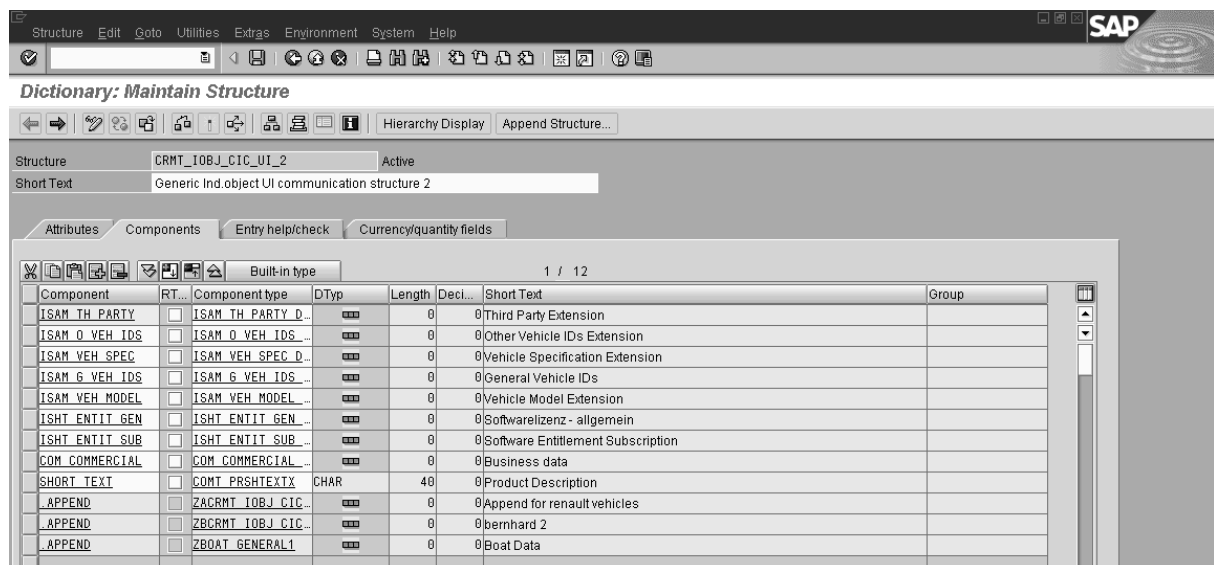


Figure 16 - Append to CRMT_IOBJ_CIC_UI_2

4 IC Customizing

1. In the Implementation Guide (IMG), choose *Interaction Center -> Component Configuration -> Define Profile for Automatically Created Workspaces*.
2. Create a new profile by copying an existing one (for example, ISAM_WP01).

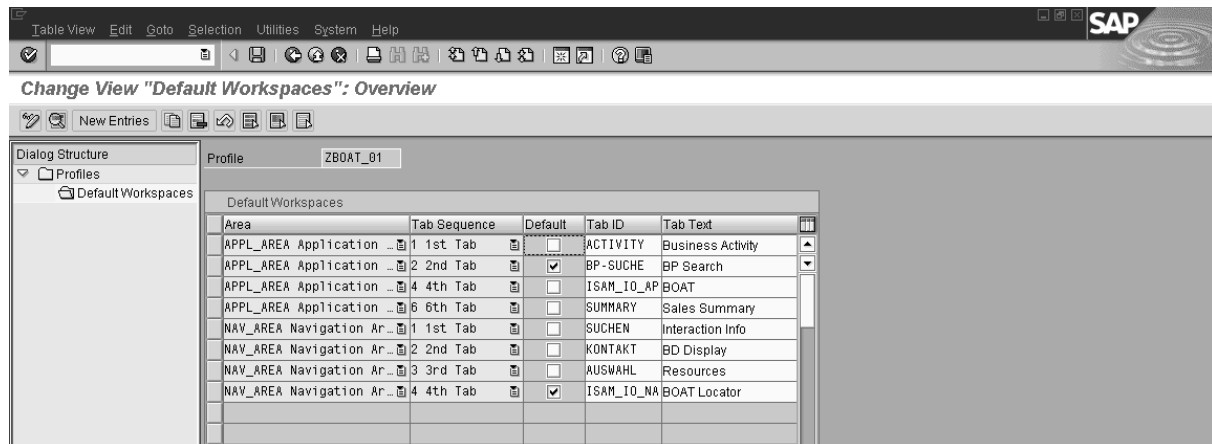


Figure 17 - Define Profile for Automatically Created Workspace

Note:

The workspace ISAM_IOBJECT_APP is responsible for displaying and maintaining the individual object (for example, boat).

The workspace ISAM_IOBJECT_NAV is responsible for finding the individual object.

3. In the IMG, choose *Interaction Center -> Define IC Profile*.
4. Create a new IC profile by copying an existing one (for example, ISAM_P01).

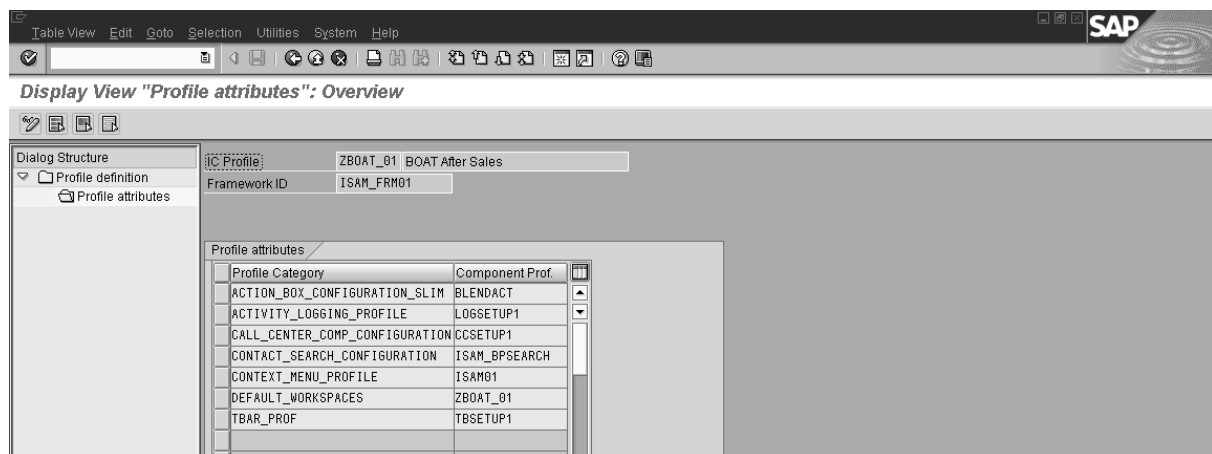


Figure 18 - Define IC Profile

4.1 Maintain Application Area of Individual Objects

You need to maintain the application area for each category and Interaction Center profile:

1. In the Implementation Guide (IMG), choose *Customer Relationship Management -> Interaction Center -> Component Configuration -> Visible Components -> Maintain Application Area of Individual Object*.
2. Copy one of the existing IC profile/category combinations (for example, ISAM_P01/ISAM_OWN_MANUALLY).

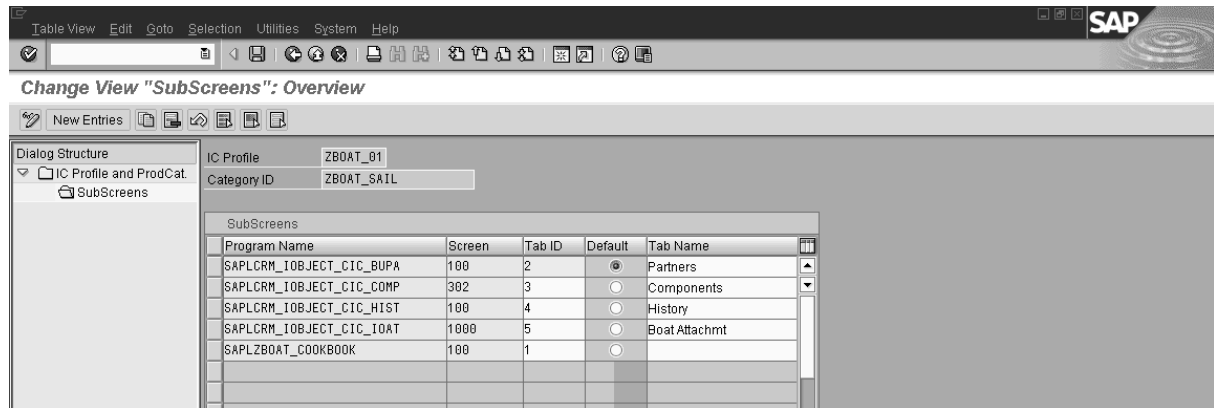


Figure 19 - Maintain Application Area of Individual Object

3. Enter the name of the function group and the screen that you created in chapter 3.
4. Add SAPL before the function group name.

The Tab ID 1 reflects the header screen in the individual object workspace:

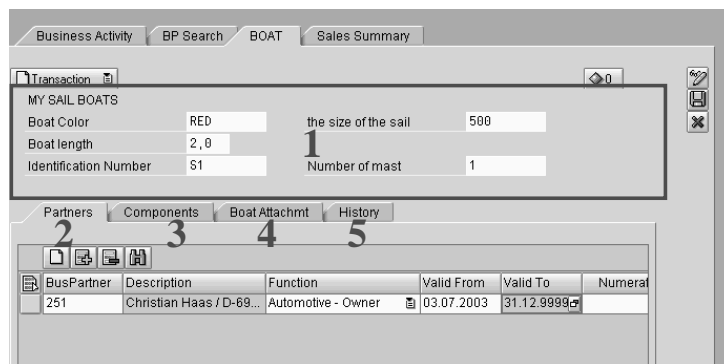
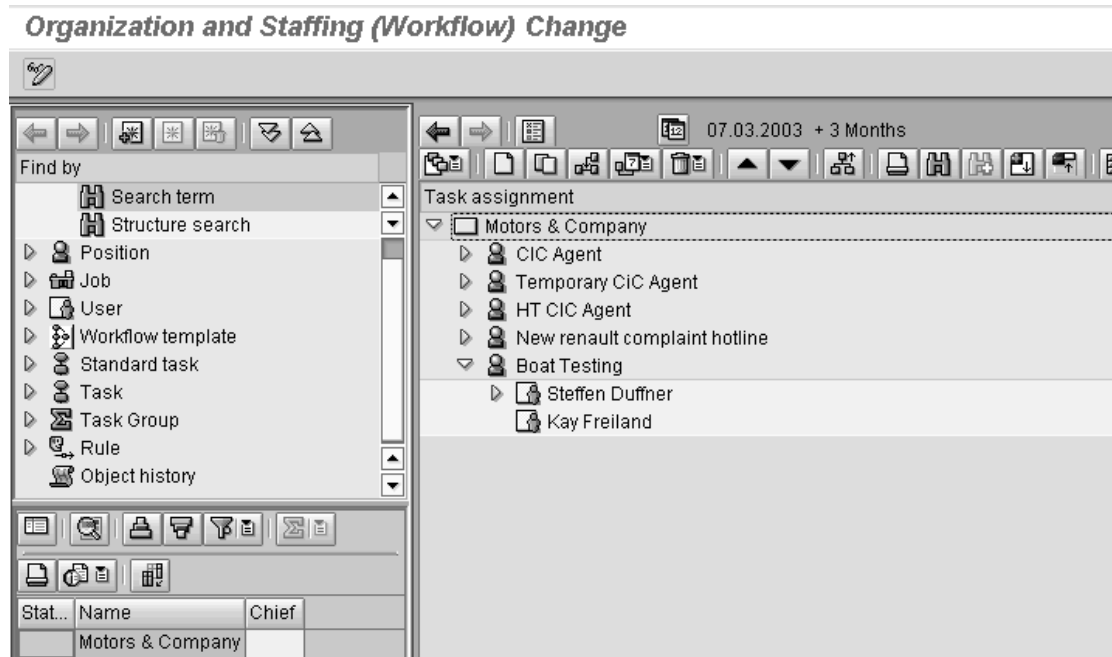


Figure 20 - Effect of Tab ID

4.2 Create IC Organization and Assign IC Profile

1. In the SAP Easy Access menu, choose *Interaction Center -> Supporting Processes -> IC Structure -> Create/Change Organization and Staffing*.
2. Create a new organization or use an existing one.



3. To create a new position (for example, *Boat Testing*), select the organization with a right mouse click.
4. To assign employees or users to the position, select the position with a right mouse click.
5. To assign the IC profile to the position of the IC users, double-click the organization you want to assign to your new profile.
6. Choose *Goto -> Detail object -> Enhanced object description*.
7. Select the field *CIC Profile* and choose *Create*.
8. In the following screen, enter the name of the new profile in the *IC Profile* field.

Maintain object

Plan version 01 Current plan
Object type S Position
Object ID 50000526 Boat Testing
Object Abbr. BOAT Agent

Active Planned Submitted Approved Rejected

Infotype Name
Object
Relationships
Description
Standard Profiles
PD Profiles
Address
Mail Address
SAP Organizational Object
General Attribute Maint.
CIC Profile

Time period
Period
From 07.03.2003 to 31.12.9999
Today
All
From curr.date
To current date
Current week
Current month
Last week
Last month
Current Year

Select.

Figure 21 - Assign IC profile (1)

Infotype Edit Goto Extras Utilities System Help

Change CIC Profile

Infotype documentation

Position ght Agent Boat Testing
Planning Status Active
Start date 07.03.2003 To 31.12.9999

IC Profile ZBOAT_01

Figure 22 - Assign IC profile (2)

5. BAdI Implementation

To be able to create individual objects for your categories (see chapter 2.4) in the IC, you need to implement the BAdI CRM_ISAM_COMMON.

Create a new implementation and enter the following program code for the method DETERMINE_DEFAULT_SETUP. Exchange the category IDs with your specific category IDs.

```
method IF_EX_CRM_ISAM_COMMON~DETERMINE_DEFAULT_SETUP .
  CONSTANTS:
    lc_motors TYPE comt_category_id VALUE
```

```

'ZBOAT_MOTOR', "use your category id
lc_sail          TYPE comt_category_id VALUE
'ZBOAT_SAIL'.    "use your category id

DATA:
  ls_categories          TYPE crmt_isam_categories,
  ls_req_settypes        TYPE COMT_FRGTYPE_ID,
  lt_req_settypes        TYPE COMT_FRGTYPE_ID_TAB.

* settings for category ZBOAT_MOTOR
CLEAR ls_categories.
MOVE lc_motors TO ls_categories-category_id. "use your category constant
* it should be possible to create ind. objects of this category
* in the interaction center (SAP GUI)
MOVE 'X'          TO ls_categories-valid_for_create.
MOVE 'X'          TO ls_categories-valid_for_partner_fct.

CALL FUNCTION 'CRM_IOBJECT_GET_CATEGORY_TEXT'
  EXPORTING
    IV_CATEGORY_ID   = ls_categories-category_id
  IMPORTING
    EV_CATEGORY_TEXT = ls_categories-category_text.

APPEND ls_categories TO et_categories.

* settings for category ZBOAT_SAIL
CLEAR ls_categories.
MOVE lc_sail TO ls_categories-category_id. "use your category constant
MOVE 'X'          TO ls_categories-valid_for_create.
MOVE 'X'          TO ls_categories-valid_for_partner_fct.

CALL FUNCTION 'CRM_IOBJECT_GET_CATEGORY_TEXT'
  EXPORTING
    IV_CATEGORY_ID   = ls_categories-category_id
  IMPORTING
    EV_CATEGORY_TEXT = ls_categories-category_text.

APPEND ls_categories TO et_categories.

* set default object family
ev_object_family = 'Z_00'. "Z_00 = Boat

endmethod.

```

6. Defining an Alternative ID Type for Your Individual Object

You defined the attribute ZBOAT_ID in chapter **Error! Reference source not found.** In this chapter you make the necessary settings so that this attribute can be used as an alternative identifier for your boats.

6.1 Define ID Types

1. In the Implementation Guide (IMG), choose *Cross-Application Components -> SAP Product -> Alternative Product IDs -> Define ID Types.*

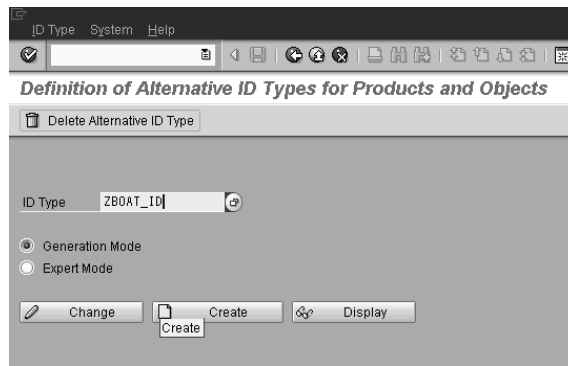


Figure 23 - Define ID Types (1)

2. Choose *Create*.

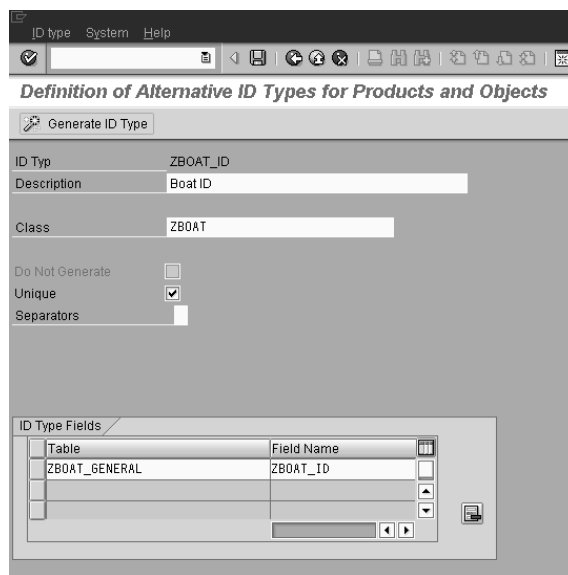


Figure 24 - Define ID Types (2)

3. Enter a description and a class name, and set the *Unique* indicator if you want the ID to be unique.
4. Specify the table and the attribute name. The table name is the same as the set type in which the attribute occurs (chapter 2.3).

6.2 Define ID Profiles

In the IMG, choose *Cross-Application Components -> SAP Product -> Alternative Product IDs -> Define Profiles*.

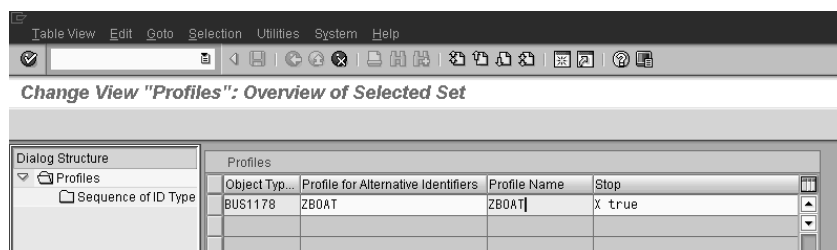


Figure 25 - Define ID Profiles (1)

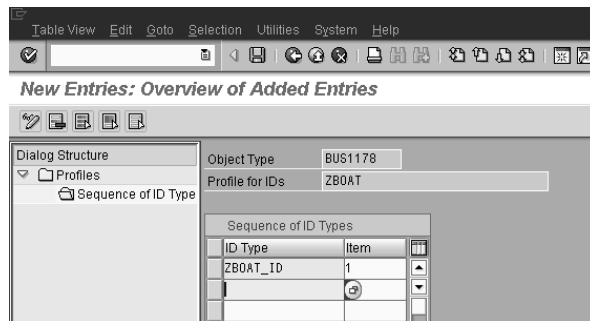


Figure 26 - Define ID Profiles (2)

6.3 Assign ID Profile to Object Family

In the IMG, choose *Cross-Application Components -> SAP Product -> Individual Object -> Assign ID Profile to Object Family*.

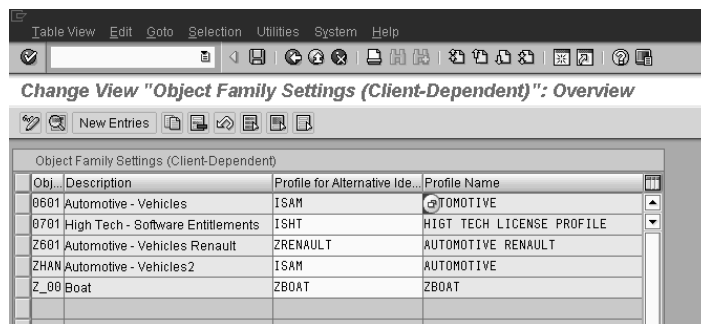


Figure 27 - Assign ID Profile to Object Family

6.4 Using the Alternative ID in the IC

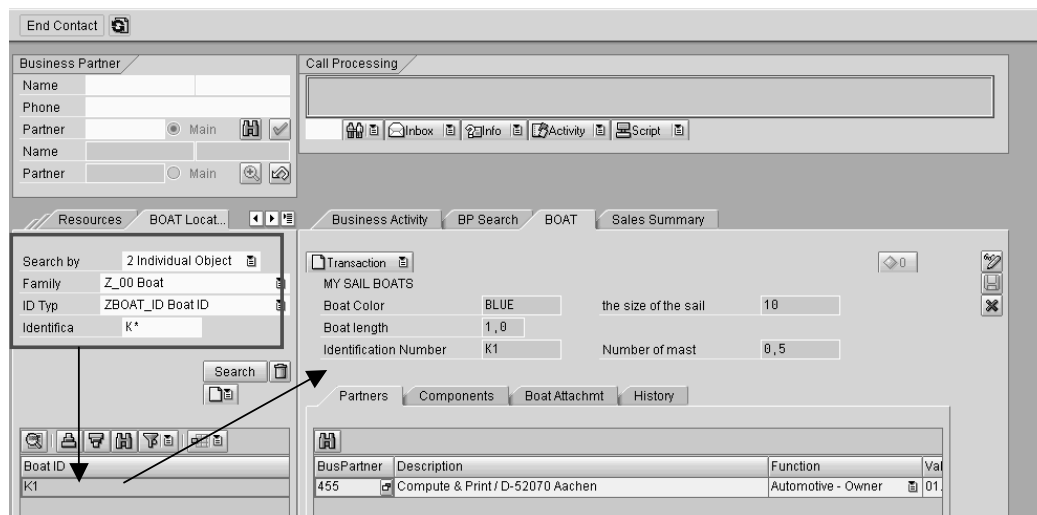


Figure 28 - Search for boats with alternative ID

7. Customizing for Event History

7.1 History Overview

The history overview is already set up for the Interaction Center by default. The same structure of histories is displayed for all object families, therefore the individual object “Boat” has the histories listed by default.

7.2 Event History

To add a history object type to the event history, proceed as follows:

In the Implementation Guide (IMG), choose *Cross-Application Components -> SAP Product -> Individual Object -> History -> Event History -> Define Object Types*.

For example, add the history object type AZ_00 derived from the Z_00 object family:

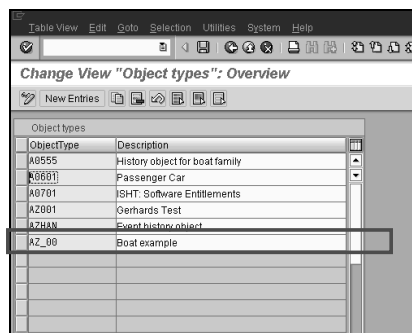


Figure 29 - Adding History Object Type AZ_00

Note: The letter “A” before the object family forms the history object type.

7.2.1 Manual maintenance

If you also want to maintain events manually in the event history, you must define a manually maintainable event:

In the IMG, choose *Cross-Application Components -> SAP Product -> Individual Object -> History -> Event History -> Define Event Types*.

For example, you can add event ZBOAT_MANUAL_NOTE as a manually maintainable event as below:

VMS_0000	Rework: Cancel Order Receipt	<input type="checkbox"/>
VMS_U000	Start Rework (PO and Reposting)	<input type="checkbox"/>
VMS_V001	Delete Vehicles with no Documents	<input type="checkbox"/>
VMS_WY00	Assign Master Warranty to Vehicle	<input type="checkbox"/>
ZBOAT_MANUAL_NOTE	Manual Note Regarding the Boat	<input checked="" type="checkbox"/>
ZHAN_EVENT1	Assign Business Documents	<input type="checkbox"/>
ZHAN_EVENT2	Assign End-Customer	<input type="checkbox"/>
ZTEST_MANUAL_1	Washed	<input checked="" type="checkbox"/>

Figure 30 - Adding a manually maintainable event

- Note 1: Set the *Manual Maintainable* indicator.
- Note 2: See the section *Link Interpretation and Extract Handling* for setting up the events as relevant to a history object type.
- Note 3: If you want to store extracts in a special format, see the section *Link Interpretation and Extract Handling* to set up the required functions. In the pre-delivered functionality, the extracts are edited in text mode and displayed as text, HTML, or XML based on the syntax used (if it starts with <HTML> it is displayed as HTML, if it starts with <XML> it is displayed as XML). The extract is displayed in an HTML container.
- Note 4: If you want the extract as an XML document, the extract must be a well-formed XML.

7.3 Integration With Business Transactions

Prerequisite 1: Make sure your business transactions are defined properly and contain the profile for alternative identifier data (defined for your object family) in their details, and an object reference profile for the individual object.

In the Implementation Guide (IMG), choose *Customer Relationship Management -> Transactions -> Basic Settings -> Define Transaction Types*.

See screenshot:

Figure 31 - Definition of transaction type

Prerequisite 2: Since business transactions are integrated through the Individual Object Integration Framework (IOITF), all the defined events (including the events that you define) must also exist in the IOITF Customizing and other corresponding Customizing. To set this up, proceed as follows:

In the IMG, choose *Cross-Application Components -> SAP Product -> Individual Object -> Individual Object Integration Framework (IOITF) -> Events -> Define/Register Event Types*.

See screenshot:

Event Name	Application Context	Individual Object Context	EHIO Rel
IOBJ_T0_10_ITEM_ENTER	CL_CRM_10_10BJECT_IOITF_EDC_A	CL_CRM_10_10BJECT_IOITF_EDC_0	<input type="checkbox"/>
IOBJ_T0_10_ITEM_REMOVE	CL_CRM_10_10BJECT_IOITF_EDC_A	CL_CRM_10_10BJECT_IOITF_EDC_0	<input type="checkbox"/>
IOBJ_IMPORT_CHANGE	CL_CRM_I_EDC_IMPORT_10BJECT	CL_CRM_E_EDC_IMPORT_10BJECT	<input type="checkbox"/>
IOBJ_IMPORT_CREATE	CL_CRM_I_EDC_IMPORT_10BJECT	CL_CRM_E_EDC_IMPORT_10BJECT	<input type="checkbox"/>
IOBJ_T0_10_ACTIVITY_ADDED			<input checked="" type="checkbox"/>
IOBJ_T0_10_ACTIVITY_REMOVED			<input checked="" type="checkbox"/>
IOBJ_T0_10_COMPL_ADDED			<input checked="" type="checkbox"/>
IOBJ_T0_10_COMPL_REMOVED			<input checked="" type="checkbox"/>
IOBJ_T0_10_CONFI_ADDED			<input checked="" type="checkbox"/>
IOBJ_T0_10_CONFI_REMOVED			<input checked="" type="checkbox"/>
IOBJ_T0_10_OPPOR_ADDED			<input checked="" type="checkbox"/>
IOBJ_T0_10_OPPOR_REMOVED			<input checked="" type="checkbox"/>
IOBJ_T0_10_SALES_ADDED			<input checked="" type="checkbox"/>
IOBJ_T0_10_SALES_REMOVED			<input checked="" type="checkbox"/>
IOBJ_T0_10_SERVI_ADDED			<input checked="" type="checkbox"/>
IOBJ_T0_10_SERVI_REMOVED			<input checked="" type="checkbox"/>
IOBJ_T0_10_SRVCONTR_ADDED			<input checked="" type="checkbox"/>
IOBJ_T0_10_SRVCONTR_REMOVED			<input checked="" type="checkbox"/>
ISHT_DOWNLOAD	CL_ISHT_REFMOD_EDC_A	CL_ISHT_REFMOD_EDC_0	<input checked="" type="checkbox"/>
ISHT_EXTEND_MAINTENANCE	CL_ISHT_REFMOD_EDC_A	CL_ISHT_REFMOD_EDC_0	<input checked="" type="checkbox"/>
ISHT_MAINTENANCE_EXPIRED	CL_ISHT_REFMOD_EDC_A	CL_ISHT_REFMOD_EDC_0	<input checked="" type="checkbox"/>
ISHT_MAINTENANCE_PURCHASED	CL_ISHT_REFMOD_EDC_A	CL_ISHT_REFMOD_EDC_0	<input checked="" type="checkbox"/>
ISHT_PRODUCT_REGISTERED	CL_ISHT_REFMOD_EDC_A	CL_ISHT_REFMOD_EDC_0	<input checked="" type="checkbox"/>
ISHT_PRODUCT_RETURNED	CL_ISHT_REFMOD_EDC_A	CL_ISHT_REFMOD_EDC_0	<input type="checkbox"/>
ISHT_UPGRADE	CL_ISHT_REFMOD_EDC_A	CL_ISHT_REFMOD_EDC_0	<input type="checkbox"/>

Figure 32 - IOTF Event Customizing

To assign them to the Object Family, proceed as follows:

In the IMG, choose *Cross-Application Components -> SAP Product -> Individual Object -> Individual Object Integration Framework (IOITF) -> Events -> Assign Event Type to Object Family*.

See screenshot:

ObjFamily	EventName
0701	ISHT_DOWNLOAD
0701	ISHT_EXTEND_MAINTENANCE
0701	ISHT_MAINTENANCE_EXPIRED
0701	ISHT_MAINTENANCE_PURCHASED
0701	ISHT_PRODUCT_REGISTERED
0701	ISHT_PRODUCT_RETURNED
0701	ISHT_UPGRADE
2201	ORDERADM_I_CHECK
2201	ORDERADM_I_DELETE
Z_00	IOBJ_T0_10_ACTIVITY_ADDED
Z_00	IOBJ_T0_10_ACTIVITY_REMOVED
Z_00	IOBJ_T0_10_COMPL_ADDED
Z_00	IOBJ_T0_10_COMPL_REMOVED
Z_00	IOBJ_T0_10_CONFI_ADDED
Z_00	IOBJ_T0_10_CONFI_REMOVED
Z_00	IOBJ_T0_10_OPPOR_ADDED
Z_00	IOBJ_T0_10_OPPOR_REMOVED
Z_00	IOBJ_T0_10_SALES_ADDED
Z_00	IOBJ_T0_10_SALES_REMOVED
Z_00	IOBJ_T0_10_SERVI_ADDED
Z_00	IOBJ_T0_10_SERVI_REMOVED
Z_00	IOBJ_T0_10_SRVCONTR_ADDED
Z_00	IOBJ_T0_10_SRVCONTR_REMOVED

Figure 33 - Assign Event Type to Object Family

To define a behavior profile, proceed as follows:

In the IMG, choose *Cross-Application Components -> SAP Product -> Individual Object -> Individual Object Integration Framework (IOITF) -> Configuration of Status Manager -> Define Behavior Profile*.

See screenshots:

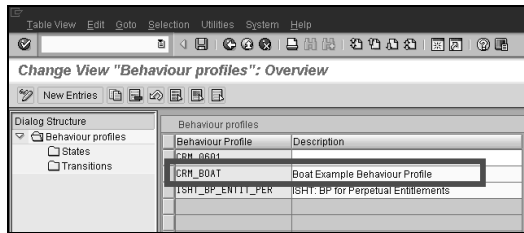


Figure 34 - Behavior Profile

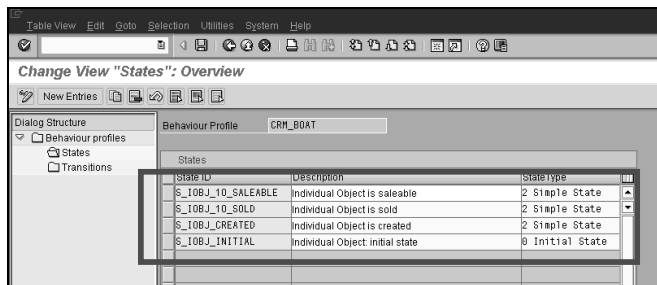


Figure 35 - States

Transitions (include all events):

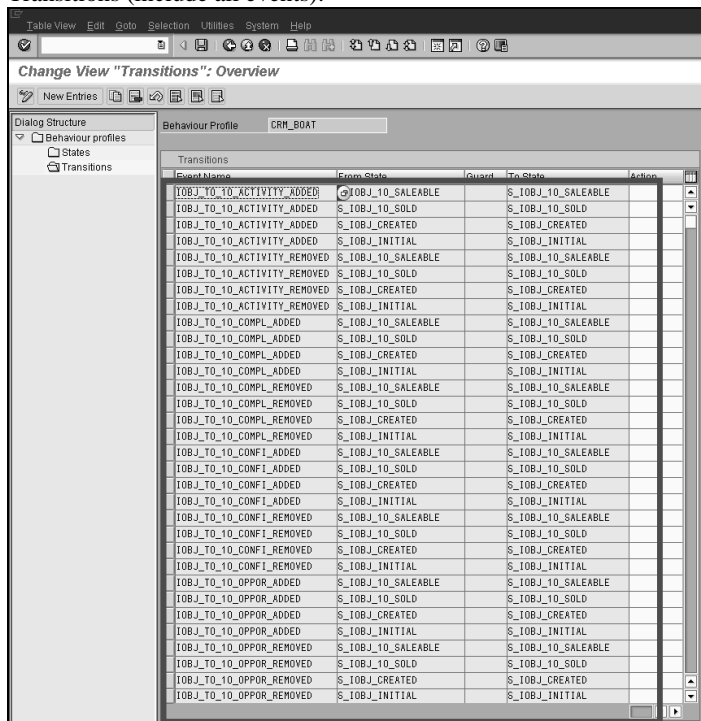


Figure 36 - Transitions

To assign a behavior profile to the Object family, proceed as follows:

In the IMG, choose *Cross-Application Components -> SAP Product -> Individual Object -> Individual Object Integration Framework (IOITF) -> Configuration of Status Manager -> Assign Behavior Profile to Object Family*.

See screenshot:

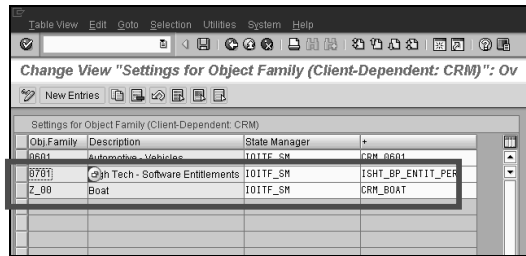


Figure 37 - Settings for the Object Family

Prerequisite 3: There must be a BAdI method implementation CRM_10_IOBJECT_EH-EH_DETERMINE_EVENT. There is a default implementation. If you need to rebuild it, follow the default implementation example.

Events for the integration with business transactions are already defined in the history:

In the IMG, choose *Cross-Application Components -> SAP Product -> Individual Object -> History -> Event History -> Define Event Types*.

See screenshot:

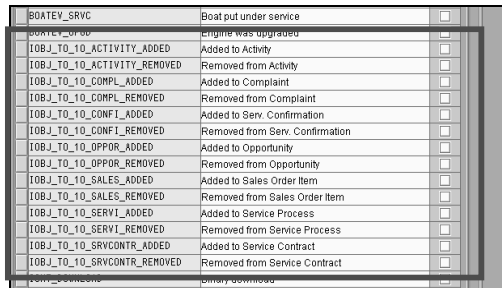


Figure 38 - Define Event Types

Note: See the section *Link Interpretation and Extract Handling* for setting up the events as relevant to a history object type.

7.4 Link Interpretation and Extract Handling

You must add a history object type to the Customizing for link interpretation.

Caution

This Customizing also sets the relevance of the history events within a history object type, therefore all events that should be recorded for an object type must be set up in this Customizing.

In the IMG, choose *Cross-Application Components -> SAP Product -> Individual Object -> History -> Event History -> Define Interpretation of Links and Extracts*.

For example, you can add the history object type AZ_00:

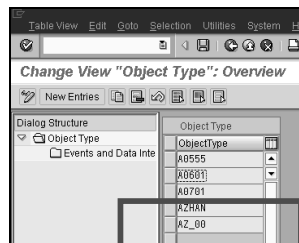


Figure 39 - Add Object Type

If you choose *Events and Data Interpretation* in the left frame, the system displays the relevant history events for the given history object type. It should include the events used for the business transaction integration and the events that can be maintained manually. External documents that are linked should also be listed here. For more details see the section *External Links*.

See screenshot:

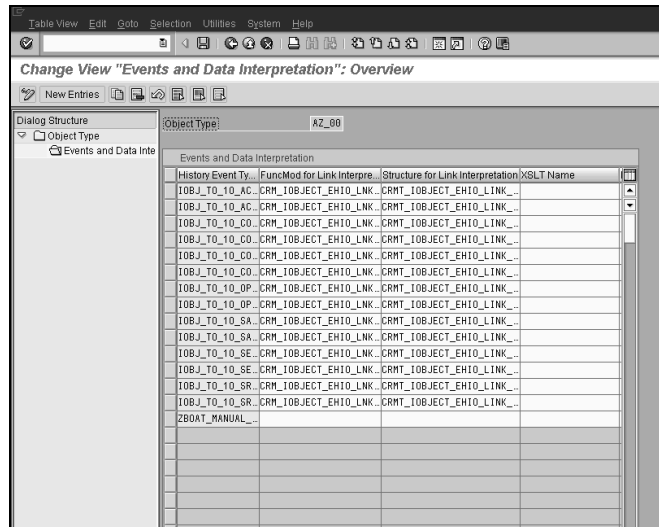


Figure 40 - Event and Data Interpretation

The following sections provide an explanation of the columns in this table.

7.4.1 History Event Type

This column contains the event types that are already defined and are required for the respective history object type. All events should go in here, including manually maintainable events, business transaction integration, and external events.

7.4.2 Function Module for Link Interpretation

Enter the name of a self-defined function module. The function module must be able to perform an action that jumps to the linked document. It has the following interface (example from the business transaction integration):

```
FUNCTION crm_iobject_ehio_lnkdspond_ui.
  IMPORTING
    REFERENCE(IV_LINK) TYPE COMT_EH_LINK
    REFERENCE(IV_STRUCTNAME) TYPE COMT_EH_LINKINTERPRETERSTRUC
    REFERENCE(IV_LOGSYS) TYPE COMT_EH_LOGSYS
    REFERENCE(IV_NOCALLTRANSACTION) TYPE COMT_EH_BOOLEAN OPTIONAL
  EXPORTING
    REFERENCE(ES_LINKDATA) TYPE COMT_EH_DISPLAY_LINK
  EXCEPTIONS
    LINKDISPLAY_FAILED
* -----
* Display functionality for one order documents triggered by Event
* History in classic UI. If the function module is called within CIC0,
* the one order workspace is opened. Otherwise, CRMD_ORDER is used for
* displaying the one order document.
* -----
```

All data is passed on to this function module when it is called.

7.4.3 Structure for Link Interpretation

This structure is defined as a type of “template” for the link string in the history record. The structure name is passed on to the interpretation function module, and the link is parsed into this structure. It is useful for easier interpretation of the link. The structure used for business transaction integration looks as follows:

Component	RT	Component type	Data Type	Length	Dec	Short Text
OBJECT_TYPE		CHAR18	CHAR	18	0	Character Field Length = 10
HEADER_GUID		CHAR32	CHAR	32	0	Character field, length 32
NUMBER_DOCU		CHAR10	CHAR	10	0	Character Field Length = 10
ITEM_GUID		CHAR32	CHAR	32	0	Character field, length 32
NUMBER_ITEM		NUMC10	NUMC	10	0	Numeric Character Field, Length 10

6.4.4 XSLT Name

XSLT is the technology used for transforming an extract before display. The extract can be modeled in XML format, for instance, and be displayed as HTML content. The transformation from XML to HTML can be made through the defined XSLT. If none is used in Customizing, the default can be used, and a 1:1 transformation is performed. The default XSLT that can be used as a starting point is shown below. The XSLT transformation **must be called** from within the extract function module display if transformation is required.

COM_EH_DEFAULTXSLT

```
<xsl:transform
xmlns:sapxsl="http://www.sap.com/sapxsl" version="1.0">
  <xsl:strip-space elements="*" />
  <xsl:template match="/">
    <xsl:copy-of select="." />
  </xsl:template>
</xsl:transform>
```

7.4.5 Function Module for Extract Display

The extract must be displayed using a function module. A default function module is part of the standard: COM_EH_DEFAULTXML_UI. This is used if no information exists in Customizing. The function module must have a fixed interface as in the default example below. At runtime, the XML extract is provided as the string together with the corresponding XSLT name.

FUNCTION com_eh_defaultxml_ui.

```

* "-----
* "Local interface:
* " IMPORTING
* "   VALUE(IV_XMLSTRING) TYPE  STRING OPTIONAL
* "   VALUE(IV_XSLTNAME) TYPE  COMT_EH_XSLTNAME OPTIONAL
* " EXCEPTIONS
* "   CNTL_ERROR
* "-----
* This function module interprets the extract of the event history record
* The record is a unlimited length string where the rows resulted at the
* edit process are separated with the value contained in the global
* constant gc_separator_ui.
* The extract can be plain text, HTML, or XML
* The result will be displayed into an HTML control, therefore some
* preparation is requires as follows:
* If the extract is plain text (does not begin with <html, <xml or
* <?xml), the special characters '<' and '>' need to be transformed
* to avoid the interpretation in the html control. Additionally,
* the separator will be replaced with <br>
* If the extract is an HTML, no special filtering is required, except
* for removing of the separator
* If the extract is an XML, it is possible to perform an XSLT transfor-
* mation using the XSLT name maintained in customizing - or in a
* personal manner.
* In this case, if no XSLT is maintained, no transformation is performed
* Due to transformations, the mime type must be checked again at the end
```

DATA: lv_xmlstring TYPE string.

DATA: lv_tmpstr TYPE string.

```

DATA: lv_str(4) TYPE c.
DATA: ls_line TYPE comt_eh_xml.
DATA: lt_html TYPE comt_eh_xml_tab.
DATA: lv_err TYPE c.
DATA: lv_splitstr TYPE string.

lv_xmlstring = iv_xmlstring.

* check if the separator is present in case extract is
* longer than 256 characters
* if not, insert it, for future splitting into table
IF strlen( lv_xmlstring ) GT 256.
  FIND gc_separator_ui IN lv_xmlstring.
  IF sy-subrc NE 0.
    lv_splitstr = lv_xmlstring.
    CLEAR lv_xmlstring.
    WHILE strlen( lv_splitstr ) > 256.
      IF lv_xmlstring IS INITIAL.
        lv_xmlstring = lv_splitstr(256).
      ELSE.
        CONCATENATE lv_xmlstring lv_splitstr(256) INTO lv_xmlstring
SEPARATED BY gc_separator_ui.
      ENDIF.
      REPLACE SECTION OFFSET 0 LENGTH 256 OF lv_splitstr WITH ''.
    ENDWHILE.
    IF NOT lv_splitstr IS INITIAL.
      CONCATENATE lv_xmlstring lv_splitstr INTO lv_xmlstring SEPARATED
BY gc_separator_ui.
    ENDIF.
  ENDIF.
ENDIF.
lv_tmpstr = lv_xmlstring.

* eliminate the first spaces in the string order to
* identify the mime-type
REPLACE ALL OCCURRENCES OF gc_separator_ui IN lv_tmpstr WITH ''.
SHIFT lv_tmpstr LEFT DELETING LEADING ' '.

* check mime-type of the extract
IF strlen( lv_tmpstr ) LT 4.
  lv_str = lv_tmpstr.
ELSE.
  lv_str = lv_tmpstr(4).
ENDIF.
TRANSLATE lv_str TO UPPER CASE.
CASE lv_str.
  WHEN '<HTM'.
    gv_xmlsubtype_ui = 'html'.
  WHEN '<XML' OR '<?XM'.
    gv_xmlsubtype_ui = 'xml'.
  WHEN OTHERS.
    gv_xmlsubtype_ui = 'plain'.
ENDCASE.

lv_err = ''.
CASE gv_xmlsubtype_ui.
  WHEN 'xml'.
    IF NOT iv_xsltname IS INITIAL.
      REPLACE ALL OCCURRENCES OF gc_separator_ui IN lv_xmlstring WITH ''.
      TRY.
* transform the extract
        CALL TRANSFORMATION (iv_xsltname)
          SOURCE XML lv_xmlstring
          RESULT XML html_tab.
* XSLT transformation error. The extract may be not well formed XML.
        CATCH cx_xslt_runtime_error.
          MESSAGE s012(com_eventhistory).
          lv_err = 'X'.
      ENDTRY.
    ENDIF.
  WHEN OTHERS.
    lv_err = 'X'.
ENDCASE.

```

```

        ENDTRY.
    ELSE.
        * if not transformation occurs, simply take existing XML
        SPLIT lv_xmlstring AT gc_separator_ui INTO TABLE html_tab.
    ENDIF.
    WHEN 'html'.
        SPLIT lv_xmlstring AT gc_separator_ui INTO TABLE html_tab.
    WHEN 'plain'.
        * if it is plain text, prepare it to be displayed in html control
        REPLACE ALL OCCURRENCES OF '<' IN lv_xmlstring WITH '&lt;'.
        REPLACE ALL OCCURRENCES OF '>' IN lv_xmlstring WITH '&gt;'.
        SPLIT lv_xmlstring AT gc_separator_ui INTO TABLE html_tab.
        REFRESH lt_html.
        LOOP AT html_tab INTO ls_line.
            APPEND ls_line TO lt_html.
            ls_line-line = '<br>'.
            APPEND ls_line TO lt_html.
        ENDLOOP.
        html_tab = lt_html.
    ENDCASE.

    IF lv_err IS INITIAL.
        * recheck the mime-type
        READ TABLE html_tab INTO ls_line INDEX 1.
        SHIFT ls_line-line LEFT DELETING LEADING ' '.
        * check mime-type of the extract
        lv_str = ls_line-line(4).
        TRANSLATE lv_str TO UPPER CASE.
        CASE lv_str.
            WHEN '<HTM'.
                gv_xmlsubtype_ui = 'html'.
            WHEN '<XML' OR '<?XM'.
                gv_xmlsubtype_ui = 'xml'.
            WHEN OTHERS.
                gv_xmlsubtype_ui = 'html'.
        ENDCASE.

        * call display screen
        CALL SCREEN 500 STARTING AT 20 5.
    ENDIF.
ENDFUNCTION.

```

7.5 External Links

To build external links, you must develop the proper function modules that perform an external call. If you want a link to another SAP R/3 system, you must create an RFC connection. For more information, see the documentation for the following transactions:

- Transaction SM59 - Create RFC target
- Transaction BD54 - Create logical system
- Transaction BD97 - Map logical system to RFC Target

7.6 Building Extracts: Example of Business Transaction Integration

Building an extract is generally up to the person who uses it. It should be provided when the record is created in the proper field. For business transaction integration, a BAdI is already defined to allow history record changes. Within the history record, the extract field can also be updated at this point.

You must implement the BAdI method CRM_IO_IOBJECT_EH -EH_UPDATE_EVHIST. A simple example is shown below:


```
METHOD if_ex_crm_1o_iobject_eh~eh_update_evhist.  
  CONCATENATE  
    '<html><body bgcolor=#ffffdd>'  
    '<center><h4>Extract</h4></center>'  
    '<li> Event Name: '  
    cr_evhist_ctx->event_name  
    '<li> Object Family: '  
    cr_evhist_ctx->object_family  
    '<li> Event Description: '  
    cr_evhist_ctx->ehio_structure-evhist_record-descript  
    '</body></html>'  
  INTO  
    cr_evhist_ctx->ehio_structure-evhist_record-histextract.  
ENDMETHOD.
```