How To…
Implement
Business
Scenarios for
customer
loyalty
Management
Version 1.00 – November 2004

Applicable Releases:
SAP NetWeaver '04
(CRM 4.0, SEM 3.50, BW 3.1/BI Content 3.3.Add-On)

THE BEST-RUN BUSINESSES RUN SAP
° Copyright 2004 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, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

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.

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.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, 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. 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.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided "as is" 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 shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

SAP NetWeaver “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.
1 Business Scenario

Customer Loyalty Management is an important business scenario within Customer Relationship Management. It is of particular importance when businesses deal with consumers (“Business to Consumer scenarios” or B2C). Marketing experts have proposed to introduce “Loyalty Accounts” for this purpose. In this approach each customer gets assigned one or several numbers, which we will call “Loyalty Points”. Within the Marketing Process it is defined how the interaction with the customer will change the number of Loyalty Points. Typically incoming orders will increase the number of Loyalty Points. Depending on the number of points acquired by the customer, the customer will have certain benefits or may get specific offerings. The usage of these benefits will then reduce the number of Loyalty Points.

Within the different industries there is of a wide range of examples for Customer Loyalty Management. We would like to mention two examples, which show the different level of complexity, when dealing with Loyalty Points: Airlines have introduced “Frequent Flyer Programs”. For each flight the customers is granted a number of Loyalty Points (“Miles”). If the customer has acquired a sufficient number of Miles, he may use them to book a new flight. Some Airlines have built collaborative scenarios involving car rental companies, hotel chains and other businesses. The computation of Loyalty Points can be very complicated. For example the Airline may want to grant the customer additional Loyalty Points, if the customer uses a rental car in connection with a business flight. The other example comes from Retail. Many Retailers have introduced a Loyalty Card, which allows to identify the customer at the POS and to interact directly with the customers. Some companies send out Coupons or Vouchers to their customers, others work with rebates and send back money to the customer’s account.

This paper explains how to implement Customer Loyalty Management on the basis of Loyalty Points with low level of complexity. We will therefore exclude the following topics:

1. Usage of Loyalty Points for Payment.

2. Front end applications, which would allow a customer to interact with the company to get information about the number of Loyalty Points or to request corrections. We would like to mention that CRM 4.0 provides sufficient flexibility to build such application for the Customer Interaction Center or for Internet Sales on project level.

3. Complex Condition for the computation of Loyalty Points. By this we mean, that the Loyalty Points for a number of transactions with different items can be computed by computing first the Loyalty Points for each item and then adding them up.

We would like to add several comments:

a) The storage and computation of the Loyalty Points will be done in the Business Information Warehouse. Our restriction on complex conditions allows us to compute the Loyalty Points within the staging process in BW when data are loaded into BW. We will therefore call such rules for the calculation of points “a-priori-rules”. The BW system provides extensive mechanisms to fully control the upload of data and to ensure data integrity and completeness. Therefore standard BW procedures can guarantee the correctness of the Loyalty Data without any enhancement of the control mechanisms. In some businesses like Retail a large number of transactions need to be processed, which can only be done efficiently if the computation of Loyalty Points can be done on item level. For businesses with high transaction volume performance considerations lead to the conclusion, that the marketing process should not allow rules for the computation of Loyalty Points, which link different items of customer transactions. The discussion can be summarized in the following statement: “A-priori-Rules” in Customer Loyalty Management are more suitable for the processing of large volumes of transactions.

b) The business process however may require the implementation of more sophisticated Loyalty Point schemes to allow for rebates. In their simplest version rebate rules state, that a customer gets additional Loyalty Points, if his transaction volume within a specified time period exceeds a certain amount. Such rules can only be applied after all data for a time period are available (“a-posteriori-rules”). Within BW
such "a-posteriori-rules" can be implemented efficiently with the help of the Analysis Process Designer. All the methods explained in this paper apply also for rebate computations. The only difference is that the Upload Process needs to be replaced by a corresponding Analysis Process. A thorough implementation of such a process for productive use however will face two additional problems: The BW Administration has to guarantee that the Analysis Process for the computation of rebates will only be started after all data for the period to be processed are stored in BW and known to be correct. The BW Administration also needs to guarantee that the rebate computation for a certain period will be performed only once. Rebate scenarios therefore require that a sophisticated data quality management is in place on BW side.

c) If Loyalty Points cannot be used for payment, different mechanisms for the reduction of Loyalty Points need to be implemented. In most business scenarios Loyalty points have a limited validity period. Some Retailers send out vouchers. The customer can use them to purchase products at a reduced price, to pick up free gifts or to cash a certain amount of money in the store. In all these cases the POS data will record such events and within the upload process the Loyalty Points can be reduced accordingly. Some businesses require that the Loyalty Points are reduced immediately after a voucher is sent to a customer, even if the customer never uses the voucher. In this case the point reduction needs to be linked to the upload of outbound contact information instead of the POS upload. After these explanations about the scope of the treatment of Customer Loyalty Management and Loyalty Point, we will briefly describe the business process and how a business user will interact with the system.

Our example implementation is a Retail scenario and the Loyalty Points are computed from the incoming POS data. The scenario would be similar, if we would use sales data from R/3 or CRM, because then we could omit the step to determine the Business Partner Number for the Loyalty Card Number used in the store:

- The distribution of Loyalty Points will be defined by creating Marketing Campaigns in CRM. The CRM Marketing Planner will be enhanced by a new Planning Application, which allows the Marketing Specialist to define "Rules" how to compute Loyalty Points on the basis of the POS data, which are stored in the BW system. These rules specify the number of points, a customer would get for each currency unit he spent in the stores, or the number of points, the customer would get for each unit he bought in the store or a combination of both. Different Planning Applications will be created to impose either general rules, which apply for each product and each business partner, or to impose rules, which depend on the product or on the business partner or on both. The rules will apply for time periods specified in the campaign. In our example implementation the rule is applied in the time period specified by the plan data.

- The Loyalty Points for each customer are stored in an InfoProvider in BW, which holds the information when the Loyalty Points were created (date of the POS) and with respect to which campaign the points have been created. In our example implementation the Loyalty Points are updated from Loyalty Marketing ODS Objects defined in the Retail Business Content.

- After the Loyalty Points have been computed in BW, an Analysis Process is used to transfer the actual number of Loyalty Points for each customer to CRM. For this purpose an Attribute Set for the storage of Loyalty Points is created in CRM. This Attribute set will contain an integer valued Marketing Attribute for the storage of Loyalty Points.

- The Marketing Specialist can now use the CRM Segment Builder to identify Customer Segments, which apply for an outbound Loyalty Campaign by selecting those customers for which the number of Loyalty Points has exceeded a certain threshold. In our scenario we assume that the company will send out vouchers to their customers, which they can use during their next shopping in the store. The example implementation does not implement a process for the reduction of points ("point reduction scheme"). Point reduction schemes can be implemented using precisely the same methods, which are used to compute the Loyalty Points from the incoming POS data. Vouchers for Point Reduction can either be identified by the POS system, when they are used by the customer in the store, and processed
with the POS upload. Or the points can be reduced immediately after the customer has received the voucher using the upload of the out bound contact information from CRM.

2 Introduction

The setup for the application described in the previous chapter requires preparatory work in BW, SEM and CRM:

- Create Planning InfoCube in BW for the storage of the rules
- Create Planning Area and Planning Level in SEM
- Define Planning Profile Groups in CRM Customizing
- Enhance Planning Layouts in SEM
- Create InfoCube for the Storage of Loyalty Points in BW
- Create Coding for the retrieval of rules from the Planning InfoCube
- Define Update Rules for the InfoCube with Loyalty Points using the Coding of the previous step. This requires the creation of a start routine and several update routines
- Define Marketing Attributes and Attribute Set in CRM for the storage of Loyalty Points
- Define Analysis Process to transfer Loyalty Points from BW to CRM

After this, a Marketing Specialist can define “Loyalty Campaigns” in CRM, choose one of the Planning Layouts defined for “Loyalty Management” and the simply enter the data for the rules in the Key Figure Planning for the Campaign Elements. Then be the setup of the whole application in BW the rules will be applied, when the POS data in BW are uploaded. The creation of outbound campaign, in which all Business Partner will be contacted and receive a voucher, for which the Number of Loyalty Points exceeds a certain threshold, is a standard task in CRM and can be completely automated using the Campaign Automation features of CRM 4.0. Since this is documented elsewhere, we will omit this step here.

We would like to add one remark, which can be useful for an implementation of the scenario in Retail. The Business Partners in CRM may be used to represent a Retail Store. If such Business Partners are assigned to a Loyalty Campaign, the Retailer could define rules for Loyalty Campaigns, which are specific to the stores assigned to the campaign. For this we would need to identify in BW the Store Number from the POS data with the Business Partner Number in CRM which represents the Retail Store. We did not include this feature in our example implementation, but our coding, which retrieves the rules form the Planning InfoCube, would already be able to deal with this. We are only missing the identification of stores and Business Partners in BW. But this could easily be added.

Another remark is related to the storage of the Campaign Reference in the InfoCube for the Loyalty Points. For the measurement of a Loyalty Management Process we believe that it is important to be able to report on the number of Loyalty Points, which were created with respect to a specific Loyalty Campaign. Now rules from several Loyalty Campaigns may apply to one specific item within a customer’s transaction. So from a single item may produce multiple entries of Loyalty Points with reference to different Loyalty Campaigns. For the processing of Loyalty Points in the staging process this implies that those lines in the data package need to be replicated, which will produce multiple lines for the Loyalty Points. In the example implementation we have introduced a simplified treatment to deal with the replication problem, which we briefly want to describe. Within the start routine we first determine the first and the last date for all lines in the data package. We then retrieve all Loyalty Campaigns, for which the time interval for the plan data has an overlap with time interval from the data package. Next we replicate each line in the data package by the number of Loyalty Campaigns. When processing a line in the new data package from the position of the line in the package we determine the Loyalty Campaign. If Loyalty Points are created with respect to the Loyalty Campaign, we update the line, otherwise the line is
rejected. This procedure will work without risk even with large data package size if the number of concurrent Loyalty Campaigns is small. Otherwise more sophistication needs to be put in the start routine to only replicate those lines in the data package, for which multiple output is required. In this case also the computations of the Loyalty Points should be done in the start routine, the results should be stored in an internal table with a reference to the line in the data package. The update rules will then distribute the values.

Since the whole application requires preparation in several components (BW, CRM, SEM) we will try to be as detailed as possible. This approach will hopefully make it easy to understand and implement “Customer Loyalty Management” even if the knowledge with respect to some of the components is limited. From a system perspective, our implementation of “Customer Loyalty Management” requires no modification. The most delicate part of the implementation is located in BW in Update Rules, where the computation of the Loyalty Points is performed on the basis of the customer transactions (in our example implementation: POS data). The definition of a new planning application in BW / SEM and the integration of the planning application in the CRM Marketing Planner is a standard procedure.
3 The Step By Step Solution

The following list replicates the list for the preparatory work and links them to the objects of our example implementation:

1. Create Planning InfoCube ZCRMLOM in BW for the storage of the rules
2. Create Planning Area ZCRMLOM and Planning Level ZCRMART in SEM
3. Define Planning Profile Groups ZLOM in CRM Customizing
4. Enhance Planning Layouts in SEM (ZLOM, ZLOMART)
5. Create InfoCube ZLOMPTS for the Storage of Loyalty Points in BW
6. Create Coding for the retrieval of rules from the Planning InfoCube
   a. DDIC Structure ZLOM_S_POINTS_SCHEME and table type ZLOM_T_POINTS_SCHEME
   b. Function Group ZLOM and function module ZLOM_POINT_CALCULATION
7. Define Update Rules for the InfoCube with Loyalty Points using the Coding of the previous step
8. Define Marketing Attributes ZNBRPOINTS and Attribute Set in CRM for the storage of Loyalty Points
9. Define Analysis Process ZYLPTS to transfer Loyalty Points from BW to CRM
10. User perspective: Loyalty Points as properties of a Business Partner
11. User perspective: How to define Loyalty Campaigns in the CRM Marketing Planner
1 Create Planning InfoCube ZCRMLOM in BW for the storage of the rules

Transaction: RSDCUBE “Edit InfoCube” -> Create
Create InfoCube “ZCRMLOM” with description “Plan cube for Loyalty Marketing”:

Include the following characteristics:

<table>
<thead>
<tr>
<th>InfoObject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0CRM_MKTELM</td>
<td>CRM Marketing Element (Campaign and Marketing Plan)</td>
</tr>
<tr>
<td>0BPARTNER</td>
<td>Business Partner</td>
</tr>
<tr>
<td>0CRMVERSION</td>
<td>CRM Plan Version</td>
</tr>
<tr>
<td>0CRM_PROD</td>
<td>Product</td>
</tr>
<tr>
<td>0VERSION</td>
<td>Version</td>
</tr>
</tbody>
</table>

(The other characteristics in the InfoCube were not be used in the scenario described here)

Assign characteristics to dimensions:

<table>
<thead>
<tr>
<th>InfoObject</th>
<th>Description</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>0CRM_MKTELM</td>
<td>CRM Marketing Element</td>
<td>Campaign</td>
</tr>
<tr>
<td>0BPARTNER</td>
<td>Business Partner</td>
<td>Store</td>
</tr>
<tr>
<td>0CRMVERSION</td>
<td>CRM Plan Version</td>
<td>Campaign</td>
</tr>
<tr>
<td>0CRM_PROD</td>
<td>Product</td>
<td>Material</td>
</tr>
<tr>
<td>0VERSION</td>
<td>Version</td>
<td>Campaign</td>
</tr>
</tbody>
</table>
Include time characteristics if you want to impose different rules in different periods within one campaign. In the scenario implemented here however no such time dependence was implemented (Remember that at least one time characteristic need to provided to define an InfoCube even if no time characteristics is used in the planning.)

Assign Key Figures:

<table>
<thead>
<tr>
<th>InfoObject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZBONUSP</td>
<td>Bonus points per Piece</td>
</tr>
<tr>
<td>ZBONUSA</td>
<td>Bonus points per Amount</td>
</tr>
<tr>
<td>0KEYFG_FLAG</td>
<td>Flag for Planning Layout Display</td>
</tr>
</tbody>
</table>
(The Key Figure ZRVALUE, “Rule type value”, was not used in the scenario described here.)
ZBONUSP and ZBONUSA are key figures specifically created for this scenario to store the number of bonus points per currency or per unit spent by the customer. Note that 0KEYFG_FLAG needs to be included.

2 Create Planning Area ZCRMLOM and Planning Level ZCRMART in SEM

Transaction BPS0 “Business Planning and Simulation”
Create Planning Area: “ZCRMLOM” with description “Loyalty Marketing Bonus Points Planning”.
(Specify “local” if SEM and BW system are the same, otherwise “remote” and the RFC Destination for the BW system,)
Assign InfoCube “ZCRMLOM” “Plan cube for Loyalty Marketing”,
Create one or several Planning Levels for the Planning Area, for example “ZCRMART” with description “Bonus Points Planning for Articles”,
Choose characteristics. In the example implementation for the planning level ZCRMART we want to impose rules on specific products, but independent from Customers or Stores (which are Business Partners in CRM). We assume that in our Loyalty Campaign we want to fix the currency to “EUR”, CRM Version needs to be included for technical reasons. Version is included to distinguish the different Planning Layouts:

<table>
<thead>
<tr>
<th>InfoObject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0CRM_MKTELM</td>
<td>CRM Marketing Element (Campaign and Marketing Plan)</td>
</tr>
<tr>
<td>0CRMVERSION</td>
<td>CRM Plan Version</td>
</tr>
<tr>
<td>0P_CURRENCY</td>
<td>Planned Currency</td>
</tr>
<tr>
<td>0CRM_PROD</td>
<td>Product</td>
</tr>
<tr>
<td>0VERSION</td>
<td>Version</td>
</tr>
</tbody>
</table>
(0CRM_MKTELM and 0CRM_VERSION should be marked as “Selection in Package”)
Maintain Selection: Currency=“EUR”, Version=“1”.

Select as Key Figures:

<table>
<thead>
<tr>
<th>InfoObject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZBONUSA</td>
<td>Bonus points per Amount</td>
</tr>
<tr>
<td>ZBONUSP</td>
<td>Bonus points per Piece</td>
</tr>
<tr>
<td>0KEYFG_FLAG</td>
<td>Flag for Planning Layout Display</td>
</tr>
</tbody>
</table>

(The “Flag for Planning Layout Display” needs to be included for technical reason; the other two Key Figures present the two different methods for Point Calculation: Assign Loyalty Points by the amount of money or by the number of articles).
The procedure is similar for other Planning Levels, which you may want to define.
3 Define Planning Profile Groups ZLOM in CRM Customizing

Transaction SPRO “Customizing”
→ Customer Relationship Management
→ Marketing
→ Marketing Planning and Campaign Management
→ Key Figure Planning
→ Define Planning Profile Groups:
Provide PlanProfGp “ZLOM”
Provide Description “Loyalty Marketing Planning Group”
Assign Category: Campaign

<table>
<thead>
<tr>
<th>Key Figure Planning Groups</th>
<th>Description</th>
<th>New KPI Flag</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZCRMLOM1</td>
<td>SAP: Loyalty Marketing Planning Group</td>
<td>✔️</td>
<td>Campaign</td>
</tr>
</tbody>
</table>

Use F4-Help to assign one or several Profiles: “ZCRMLOM1” (and possibly others).
Assign Plan Type “General Key Figures”
Provide Version: “0000000001”
(The assignment of a CRM Version is necessary for the Planning in CRM)
4 Enhance Planning Layouts in SEM (ZLOM, ZLOMART)

Transaction: UPX_MNTN1 “Settings for Planning Services”

SEM:
Business Analytics
→ Customer Relationship Analytics
→ Planning Services
→ Settings for Planning Services:

Enhance Planning Layout: “Bonus Points Planning Layout for Articles” (ZCRMART), Layout for bonus points (ZCRMLOM1)
Header Area tab strip: Disable display of Version and CRM Version.
Lead Column tab strip: Assign Product to rows (vertical arrow) and Key Figures to columns (horizontal arrow).

Key Figure Scheme tab strip: enter “Rule type value”, “Rule type unit”, “Layout Flag”.

Maintain Enhancements for “Rule type value”: Set the “Hide Currency” flag.

Maintain Enhancements for “Layout Flag”: set the “Hidden” flag.
Then the Preview tab strip displays how the planning area would roughly look. Note that in our example the “Planning Currency” will be displayed in the header, since it was fixed to EUR in the planning level ZCRMART.

5 Create InfoCube ZLOMPTS for the Storage of Loyalty Points in BW

Transaction RSDCUBE “Edit InfoCube”: ZLOMPTS, “Loyalty Points” -> Create

<table>
<thead>
<tr>
<th>InfoObject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0BPARTNER</td>
<td>Business Partner</td>
</tr>
<tr>
<td>0CRM_MKTELM</td>
<td>CRM Marketing Element (Campaign and Marketing Plan)</td>
</tr>
</tbody>
</table>
In the example implementation we assume, that the Marketing Department wants to report on the number of Loyalty Points created for each Loyalty Campaign.

Then assign to respective dimensions.

**Assign time characteristics:**

<table>
<thead>
<tr>
<th>InfoObject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0CALDAY</td>
<td>Calendar day</td>
</tr>
<tr>
<td>0CALWEEK</td>
<td>Calendar Year/Week</td>
</tr>
<tr>
<td>0CALMONTH</td>
<td>Calendar Year/Month</td>
</tr>
<tr>
<td>0CALYEAR</td>
<td>Calendar year</td>
</tr>
</tbody>
</table>
Assign Key Figures:

<table>
<thead>
<tr>
<th>InfoObject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZLOMPTD</td>
<td>Loyalty Points (Dec)</td>
</tr>
</tbody>
</table>

In the InfoCube for the prototype the Key Figures 0RT_POSSAL (“Sales (POS Receipt)”), 0RT_SALRESA (“Quantity in Sales Unit (POS Rcpt)”), were included to verify that the computation of Loyalty Points is correct. This should not be done in a productive customer environment. We would like to point out that it is possible that multiple rules apply to the same sales item. Therefore the aggregated Key Figures for Sales and Quantity are expected to be different from the corresponding aggregates in the sales data (in our example implementation: POS-Data from a retailer's stores).
In our scenario we have defined the key figure for the loyalty points to be of type DEC. This would allow fractional values for the Loyalty Points. The usage of a Key Figure of type INT would mean that points are rounded off for each sales item. This would result in unpredictable results for multiple items within one order or receipt: Multiple items may appear in one line of the order / receipt or in several lines. This can lead to different round-offs.

The number of points transmitted to CRM however will be integers.

6 Create Coding for the retrieval of rules from the Planning InfoCube

- DDIC Structure ZLOM_S_POINTS_SCHEME
The details for the implementation are included in the Appendix 1: Coding for the Update Rules and Comments. For convenience we have included a number of screenshots from the ABAP Workbench (Transaction SE80)

7 Define Update Rules for the InfoCube with Loyalty Points using the Coding of the previous step

In the example implementation the data for the InfoCube ZLOMPTS are retrieved from one the ODS Objects from “Customer Loyalty Marketing”: 0LOM_DS12 “POS Data for Loyalty Program: December”. “Customer Loyalty Marketing” is a complete Scenario in the BW Business Content for Retail. This scenario gives recommendations how to do the staging of POS data in BW. For performance reasons 12 ODS Objects for the initial storage of POS data are defined, one for each month (0LOM_DS01 ”POS Data for Loyalty Program: January” – 0LOM_DS12 ”POS Data for Loyalty Program: December”), from which the data will be staged into secondary InfoProviders. The idea behind this was that the storage of POS data on item level may be too expensive for a retail business. So the data should only be kept for some time for detail analysis (like Cross-Selling Analysis) and only aggregated data will be kept for periods longer than a year. The ODS Objects should therefore be used in a circular way, with the beginning of a new month last year POS data will be replaced by the new data. The “Loyalty Points” InfoCube is nothing but an
additional secondary InfoProvider, which should receive its data each month from the corresponding Loyalty ODS Object. In the example implementation we only defined Update Rules for the December ODS Object.

In this step the creation of a Start Routine and several Update Routines is required. The coding for this you will find in the appendix. Some Screenshots from the Administrator Workbench are included for convenience.

Transaction: RSA1 Administrator Workbench

→ Modelling
→ InfoCube ZLOMPTS “Loyalty Points”
→ Right mouse click: “Create Update Rules”
→ Select “ODS Object”, fill in “0LOM*” and push F4 to select one of the ODSs from “Loyalty Marketing”

![Administrator Workbench: Modeling](image-url)
Choose ODS Object OLOM_DS12 as DataSource:

Afterwards, you get to a screen that displays the mapping of source fields to the respective key figures in the fact table. The icon indicates that the key figure has been derived via an ABAP routine. For details about the coding of the ABAP routines please refer to the appendix.
The screen shows the start routine of the update rule, which can be reached by pressing the push button on the previous screen.

To change (or create) the mapping (in our case the ABAP routine) for the respective key figure, double-click on the respective key figure and choose the 'Update method' as needed.
This picture shows an example of such an ABAP routine.

8 Define Marketing Attributes ZNBRPOINTS and Attribute Set in CRM for the storage of Loyalty Points

The example implementation uses an Attribute Set in CRM, which includes a number of different Marketing Attributes used for this specific Loyalty Marketing Scenario. In general the number of Loyalty Points will only be one of several properties, which are used in “Customer Loyalty Management” and which be bundled in a customer specific Attribute Set in CRM.

Marketing
  ➔ Segmentation of Business Partners
  ➔ Marketing Attributes
  ➔ Maintain Marketing Attributes
Transaction CRMD_PROF_CHAR “Maintain Attributes”
Define Marketing Attribute ZNBR_POINTS “Loyalty Points” with properties: Numeric Format, No of digits 15, decimals 0.

Marketing
  ➔ Segmentation of Business Partners
  ➔ Marketing Attributes
  ➔ Maintain Attribute Set
Transaction CRMD_PROF_TEMPL “Maintain Attribute Set”
Find or create a suitable Attribute Set and add ZNBR_POINTS.
9 Define Analysis Process ZYLPTS to transfer Loyalty Points from BW to CRM

Define Analysis Process to transfer Loyalty Points from BW to CRM (Full Upload)
Transaction RSANWB “Model the Analysis Process”
→ Fill CRM Attributes
→ Create Analysis Process ZYLPTS “Analysis Process: Loyalty Points for Customers (BW->CRM)”
In the simplest “Full Upload Scenario” define as source a query, which for each Business Partner retrieves the integral part number of Loyalty Points from ZLOMPTS. As Target assign the CRM System and the Marketing Attribute ZNBR_POINTS. Then maintain the field assignment between Source and Target. For productive use more efficient Upload processes should be defined which also restrict the number business partners to be transferred to CRM (“Delta Upload”). Also the upload should incorporate some Business Rules. An example for this would be the maximum time period, for which Loyalty Points are valid.
To define the source of the data, double-click on the icon in the Analysis Process Designer.

To define the target of the data, double click on the icon in the Analysis Process Designer.
To define how the field from the source of the data should be mapped to the data target, double-click on the icon.

In the example implementation the Loyalty Points for each Business Partner are selected without any time constraint. For productive use more sophisticated selections are required to only transfer the Loyalty Points for those Business Partners, for which the number of Points has changed ("Delta Upload"). Within an Analysis Process several methods are available to retrieve data from an InfoProvider. The most powerful option is to use a Query, but for the processing of very large numbers of Business Partners the preferred method is to read the data directly from the InfoCube and apply appropriate filtering afterwards. For simplicity we have chosen the Query Interface.
The InfoCube Loyalty Points stores fractional parts of the Loyalty Points. This is done to avoid round off errors during the computation of the Loyalty Points. Within CRM Marketing only the integer part of the Loyalty Points will be used.
10 User perspective: Loyalty Points as properties of a Business Partner
11  User perspective: How to define Loyalty Campaigns in the CRM Marketing Planner

Transaction CRM_MKTPL “Marketing Planner”
Create Campaign, “MCR-2003-01-02”, “Campagne Fidélité”.
Assign Planning Profile Group: “Loyalty Marketing Planning Group”

Create Campaign Element: “MCR-2003-01-02-01”, “Campagne Fêtes des Mères”.
Maintain “Plan Start Date”, “Plan End Date” to specify in which time interval the rule should be applied
Assign products to campaign elements (“promoted articles”)
Maintain Key Figures per Product depending on the rule you would like to impose:
Customers will get 2 Loyalty Points for each Euro they spend for one of the assigned 4 products and in addition 1 Loyalty Point for each unit of the second article (“Fragrances”).
Appendix 1: Coding for the Update Rules and Comments
This section provides the complete coding used in the Update Rules of the example implementation.

11.1 DDIC Structure
Create Structure ZLOM_S_POINT_SCHEME “Loyalty Point Scheme” with fields

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Fieldtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRM_MKTELM</td>
<td>RSCHAVL</td>
</tr>
<tr>
<td>START_DATE</td>
<td>DATS</td>
</tr>
<tr>
<td>END_DATE</td>
<td>DATS</td>
</tr>
<tr>
<td>BPARTNER</td>
<td>RSCHAVL</td>
</tr>
<tr>
<td>CRM_PROD</td>
<td>RSCHAVL</td>
</tr>
<tr>
<td>AMOUNT</td>
<td>RSCHAVL</td>
</tr>
<tr>
<td>UNIT</td>
<td>RSDRI_UNIT</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>RSCHAVL</td>
</tr>
</tbody>
</table>

11.2 DDIC Table Type
Create table type ZLOM_T_POINT_SCHEME “Loyalty Point Scheme” of line type ZLOM_S_POINT_SCHEME.

11.3 Function Module ZLOM_POINT_CALCULATION
Create function group ZLOM “Loyalty Marketing (point calculations)” and function module ZLOM_POINT_CALCULATION “Get Calculation Scheme from SEM InfoCube”:

FUNCTION ZLOM_POINT_CALCULATION.
*"---------------------------------------------------------------------
**" IMPORTING
**"  VALUE(I_INFOCUBE) TYPE RSINFOCUBE
**"  VALUE(I_CRM_MKTELM_CHAR) TYPE RSIOBJNM
**"  VALUE(I_START_CHAR) TYPE RSIOBJNM
**"  VALUE(I_END_CHAR) TYPE RSIOBJNM
**"  VALUE(I_CRMCAMPTYP_CHAR) TYPE RSIOBJNM OPTIONAL
**"  VALUE(I_CRMVERSION_CHAR) TYPE RSIOBJNM OPTIONAL
**"  VALUE(I_BPARTNER_CHAR) TYPE RSIOBJNM OPTIONAL
**"  VALUE(I_CRM_PROD_CHAR) TYPE RSIOBJNM OPTIONAL
**"  VALUE(I_START_VALUE) TYPE DATS
**"  VALUE(I_END_VALUE) TYPE DATS
**"  VALUE(I_CRMCAMPTYP_VALUE) TYPE RSCHAVL OPTIONAL
**"  VALUE(I_CRMVERSION_VALUE) TYPE RSCHAVL OPTIONAL
**"  VALUE(I_BONUSAMOUNT_KEYF) TYPE RSIOBJNM
**"  VALUE(I_BONUSQUANTITY_KEYF) TYPE RSIOBJNM
**"  VALUE(I_LAYOUTFLAG_KEYF) TYPE RSIOBJNM
**"  TABLES
**"    T_POINT_SCHEME TYPE ZLOM_T_POINT_SCHEME
**"  EXCEPTIONS
**"    INFOCUBE_ERROR
**"    CRM_MKTELM_ERROR
type-pools: rs, rsd, rsmd, sbiw.

constants: c_resulttype type RSDRI_RESULTTYPE value 'V'.
constants: c_USE_DB_AGGREGATION type rs_bool value 'X'.
constants: c_rollup_only type rs_bool value ' '.

* structure for check of infocube
  data: l_t_cube_iobj type rsd_t_cube_iobj.
  data: w_cube_iobj type rsd_s_cube_iobj.

data: i_infoprov type RSINFOPROV.

* flag for end of data from BW read interface
  data: l_end_of_data type RS_BOOL.

* internal table infoobject to be retrieved
  data: w_sfc TYPE RSDRI_S_SFC.
  data: l_t_sfc type RSDRI_T_SFC.

* internal table with key figures to be retrieved
  data: w_sfk TYPE RSDRI_S_SFK.
  data: l_t_sfk type RSDRI_T_SFK.

* internal table with selections (combined by AND)
  data: w_range type RSDRI_S_RANGE.
  data: l_t_range type RSDRI_T_RANGE.

* data: w_tablesel type RSDRI_S_SELT.
  data: l_t_tablesel type RSDRI_T_SELT.

* internal table with requested data
  data: w_rfcdatav type RSDRI_S_RFCDATAV.
  data: l_t_rfcdatav type standard table of RSDRI_S_RFCDATAV.

  data: i_start_nav type rsiobjnm.
  data: i_end_nav   type rsiobjnm.
  data: i_camptyp_nav type rsiobjnm.
  data: i.crmversion_nav type rsiobjnm.
  data: i.crmcamptyp_nav  type rsiobjnm.
data: i_char type rsiobjnm.

data: i.crm.mktel_flag type rs_bool value rs_c_false.
data: i.start_flag type rs_bool value rs_c_false.
data: i.end_flag type rs_bool value rs_c_false.
data: i.crm.version_flag type rs_bool value rs_c_false.
data: i.crm.camptype_flag type rs_bool value rs_c_false.
data: i.bpartner_flag type rs_bool value rs_c_false.
data: i.crm.prod_flag type rs_bool value rs_c_false.
data: i.bonus.amount_flag type rs_bool value rs_c_false.
data: i.bonus.quantity_flag type rs_bool value rs_c_false.
data: i.layoutflag_flag type rs_bool value rs_c_false.

data: i.id type i value 0.
data: i.id_old type i value -1.
data: i.keyfigure_found type rs_bool value rs_c_false.
data: i.fl type p.

data: w_point_scheme type zlom_s_point_scheme.

* check if characteristics and key figures are available
* in infoprovider

CALL FUNCTION 'RSD_CUBE_GET'
EXPORTING
  I_INFOCUBE = i_infocube
  I_OBJVERS = RS_C_OBJVERS-ACTIVE
* I_BYPASS_BUFFER = RS_C_FALSE
I_WITH_ATR_NAV = RS_C_TRUE
IMPORTING
* E_S_CUBE
* E_S_TBHD
* E_TBHD_NOT_FOUND
* E_T_CUBET
  E_T_CUBE_IOBJ = l_t_cube_iobj
* E_T_CUBE_DIME
* E_T_DIME
* E_T_DIMET
* E_T_DIME_IJOB
* E_T_ICHA_PRO
* E_T_IKYF_PRO
* E_T_IC_VAL_IJOB
* E_T_CUBE_PART
* E_T_CUBE_PART_IJOB
* E_T_MULTI_IJOB
EXCEPTIONS
  INFOCUBE_NOT_FOUND = 1
  ILLEGAL_INPUT = 2
  OTHERS = 3

IF SY-SUBRC <> 0.
* could no info about the infocube
  raise infocube_error.
ENDIF.
concatenate i.crm_mktelm_char '__' i_start_char into i_start_nav.
concatenate i.crm_mktelm_char '__' i_end_char into i_end_nav.
concatenate i.crm_mktelm_char '__' i.crmversion_char into i.crmversion_nav.
concatenate i.crm_mktelm_char '__' i.crmcamptyp_char into i.crmcamptyp_nav.

loop at l.i_cube_iobj into w_cube_iobj.
if w_cube_iobj-iobjnm = i.crm_mktelm_char and w_cube_iobj-iobjtp = rsd_c_objtp-charact.
i.crm_mktelm_flag = rs_c_true.
endif.
if w_cube_iobj-iobjnm = i_start_nav and w_cube_iobj-iobjtp = rsd_c_objtp-charact.
i_start_flag = rs_c_true.
endif.
if w_cube_iobj-iobjnm = i_end_nav and w_cube_iobj-iobjtp = rsd_c_objtp-charact.
i_end_flag = rs_c_true.
endif.
if not i.crmversion_char is initial.
if w_cube_iobj-iobjnm = i.crmversion_nav and w_cube_iobj-iobjtp = rsd_c_objtp-charact.
i.crmversion_flag = rs_c_true.
endif.
endif.
if not i.crmcamptyp_char is initial.
if w_cube_iobj-iobjnm = i.crmcamptyp_nav and w_cube_iobj-iobjtp = rsd_c_objtp-charact.
i.crmcamptyp_flag = rs_c_true.
endif.
endif.
if not i.bpartner_char is initial and 
   w_cube_iobj-iobjnm = i.bpartner_char and w_cube_iobj-iobjtp = rsd_c_objtp-charact.
i.bpartner_flag = rs_c_true.
endif.
if not i.crm_prod_char is initial and 
   w_cube_iobj-iobjnm = i.crm_prod_char and w_cube_iobj-iobjtp = rsd_c_objtp-charact.
i.crm_prod_flag = rs_c_true.
endif.
if w_cube_iobj-iobjnm = i.bonusamount_keyf and w_cube_iobj-iobjtp = rsd_c_objtp-keyfigure.
i.bonusamount_flag = rs_c_true.
endif.
if w_cube_iobj-iobjnm = i.bonusquantity_keyf and w_cube_iobj-iobjtp = rsd_c_objtp-keyfigure.
i.bonusquantity_flag = rs_c_true.
endif.
if not i.layoutflag_keyf is initial and 
   w_cube_iobj-iobjnm = i.layoutflag_keyf and
\[ w_{\text{cube}}_{\text{obj-iobjtp}} = \text{rsd}_{\text{c-objtp-keyfigure}}. \]
\[ i_{\text{layoutflag-flag}} = \text{rs}_{\text{c-true}}. \]
endif.
endloop.

if \( i_{\text{crm-mktelm-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* Marketing Element InfoObject not found as navigational attribute.
raise \( \text{crm-mktelm-error} \).
endif.
if \( i_{\text{start-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* start InfoObject not found as navigational attribute.
raise \( \text{start-error} \).
endif.
if \( i_{\text{end-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* end InfoObject not found as navigational attribute
raise \( \text{end-error} \).
endif.
if not \( i_{\text{crmversion-char}} \) is initial and
\( i_{\text{crmversion-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* Campaign Version InfoObject not found as navigational attribute
raise \( \text{crmversion-error} \).
endif.
if not \( i_{\text{crmcamptyp-char}} \) is initial and
\( i_{\text{crmcamptyp-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* Campaign Type InfoObject not found as navigational attribute
raise \( \text{crmcamptyp-error} \).
endif.
if not \( i_{\text{bpartner-char}} \) is initial and
\( i_{\text{bpartner-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* Business Partner InfoObject not found as navigational attribute
raise \( \text{bpartner-error} \).
endif.
if not \( i_{\text{crm-prod-char}} \) is initial and
\( i_{\text{crm-prod-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* CRM Product InfoObject not found as navigational attribute
raise \( \text{crm-prod-error} \).
endif.
if \( i_{\text{bonusamount-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* Bonusamount InfoObject not found as navigational attribute
raise \( \text{bonusamount-error} \).
endif.
if \( i_{\text{bonusquantity-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* Bonusquantity InfoObject not found as navigational attribute
raise \( \text{bonusquantity-error} \).
endif.
if not \( i_{\text{layoutflag-keyf}} \) is initial and
\( i_{\text{layoutflag-flag}} \) eq \( \text{rs}_{\text{c-false}} \).
* Layoutflag InfoObject not found as navigational attribute
raise \( \text{layoutflag-error} \).
endif.
* fill table with the InfoObjects for the characteristics
\[ w_{\text{sfc-chanm}} = i_{\text{crm-mktelm-char}}. \]
append \( w_{\text{sfc}} \) to \( l_{\text{t-sfc}} \).
w_sfc-chanm = i_start_nav.
append w_sfc to l_t_sfc.
w_sfc-chanm = i_end_nav.
append w_sfc to l_t_sfc.
if i_bpartner_flag eq rs_c_true.
w_sfc-chanm = i_bpartner_char.
append w_sfc to l_t_sfc.
endif.
if i_crm_prod_flag eq rs_c_true.
w_sfc-chanm = i_crm_prod_char.
append w_sfc to l_t_sfc.
endif.
clear w_range.
w_range-chanm = i_start_nav.
w_range-sign = 'I'.
w_range-compop = 'LE'.
w_range-low = i_start_value.
append w_range to l_t_range.
clear w_range.
w_range-chanm = i_end_nav.
w_range-sign = 'I'.
w_range-compop = 'GE'.
w_range-low = i_end_value.
append w_range to l_t_range.
* crmversion
if i_crmversion_flag = rs_c_true and not i_crmversion_value is initial.
clear w_range.
w_range-chanm = i_crmversion_nav.
w_range-sign = 'I'.
w_range-compop = 'EQ'.
w_range-low = i_crmversion_value.
append w_range to l_t_range.
endif.
* crmcamptyp
if i_crmcamptyp_flag = rs_c_true and not i_crmcamptyp_value is initial.
clear w_range.
w_range-chanm = i_crmcamptyp_nav.
w_range-sign = 'I'.
w_range-compop = 'EQ'.
w_range-low = i_crmcamptyp_value.
append w_range to l_t_range.
endif.
* fill table with the InfoObjects for the keyfigures
if i_bonusquantity_flag eq rs_c_true.
w_sfk-kyfnm = i_bonusquantity_keyf.
append w_sfk to l_t_sfk.
endif.
if i_bonusamount_flag eq rs_c_true.
w_sfk-kyfnm = i_bonusamount_keyf.
append w_sfk to l_t_sfk.
endif.
if i_layoutflag_flag eq rs_c_true.
  w_sfk-kyfnm - i_layoutflag_keyf.
append w_sfk to l_t_sfk.
endif.
i_infoprov - i_infocube.
CALL FUNCTION 'RSDRI_INFOPROV_READ_RFC'
  EXPORTING
    I_INFOPROV                   = i_infoprov
    I_REFERENCE_DATE             = SY-DATUM
*    I_SAVE_IN_TABLE             = ' '  
*    I_TABLENAME                  =
*    I_SAVE_IN_FILE               = ' '  
*    I_FILENAME                   =
*    I_Authority_Check            - RSDRC_C_AUTHCHK-READ
*    I_CURRENCY_CONVERSION        - RS_C_TRUE
*    I_S_RFCMODE                  =
*    I_MAXROWS                    = 0
*    I_USE_DB_AGGREGATION         - RS_C_TRUE
*    I_USE_AGGREGATES             - RS_C_TRUE
*    I_ROLLUP_ONLY                - c_rollup_only
*    I_READ_ODS_DELTA             - RS_C_FALSE
*    I_RESULTTYPE                 = c_resulttype
*    I_DEBUG                      - RS_C_FALSE
IMPORTING
  E_END_OF_DATA                = l_end_of_data
*    E_AGGREGATE                  =
*    E_RFCDATA_UC                 =
*    E_SPLIT_OCCURRED             =
TABLES
  I_T_SFC                      - l_t_sfc
  I_T_SFK                      - l_t_sfk
  I_T_RANGE                    - l_t_range
  I_T_TABLESEL                 - l_t_tablesel
*    I_T_RTIME                   =
*    I_T_REQUID                   =
*    E_T_RFCDATA                  =
  E_T_RFCDATAV                 = l_t_rfcdatav
*    E_T_FIELD                    =
EXCEPTIONS
  ILLEGAL_INPUT                - 1
  ILLEGAL_INPUT_SFC            - 2
  ILLEGAL_INPUT_SFK            - 3
  ILLEGAL_INPUT_RANGE          - 4
  ILLEGAL_INPUT_TABLESEL       - 5
  NO_AUTHORIZATION             - 6
  NCUM_NOT_SUPPORTED           - 7
  GENERATION_ERROR             - 8
  ILLEGAL_DOWNLOAD             - 9
  ILLEGAL_TABLENAME            - 10
  ILLEGAL_RESULTTYPE           - 11
  TRANS_NO_WRITE_MODE          - 12
if sy-subrc ne 0.
  raise read_infocube_error.
endif.
loop at l_t_rfcdatav into w_rfcdatav.
i_id = w_rfcdatav-id.
if i_id ne i_id_old and i_id_old gt 0.
  if i_keyfigure_found eq rs_c_true.
    append w_point_scheme to t_point_scheme.
  endif.
i_keyfigure_found = rs_c_false.
clear w_point_scheme.
endif.
if w_rfcdatav-iobjnm = i_crm_mktelm_char.
  w_point_scheme-crm_mktelm = w_rfcdatav-value.
endif.
if w_rfcdatav-iobjnm = i_start_nav.
  w_point_scheme-start_date = w_rfcdatav-value.
endif.
if w_rfcdatav-iobjnm = i_end_nav.
  w_point_scheme-end_date = w_rfcdatav-value.
endif.
if not i_bpartner_char is initial and
  w_rfcdatav-iobjnm = i_bpartner_char.
  w_point_scheme-bpartner = w_rfcdatav-value.
endif.
if not i_crm_prod_char is initial and
  w_rfcdatav-iobjnm = i_crm_prod_char.
  w_point_scheme-crm_prod = w_rfcdatav-value.
endif.
if w_rfcdatav-iobjnm = i_bonusamount_keyf and
  not w_rfcdatav-value is initial.
i_fl = w_rfcdatav-value.
  w_point_scheme-amount = w_rfcdatav-value.
  w_point_scheme-unit   = w_rfcdatav-unit.
  if i_fl ne 0.
    i_keyfigure_found = rs_c_true.
  endif.
endif.
if w_rfcdatav-iobjnm = i_bonusquantity_keyf and
  not w_rfcdatav-value is initial.
i_fl = w_rfcdatav-value.
  w_point_scheme-quantity = w_rfcdatav-value.
  if i_fl ne 0.
    i_keyfigure_found = rs_c_true.
  endif.
endif.
i_id_old = i_id.
endloop.
if i_keyfigure_found eq rs_c_true.
  append w_point_scheme to t_point_scheme.
endif.

ENDFUNCTION.
11.4 Update Rules (1): Start Routine

PROGRAM UPDATE_ROUTINE.

TYPES:
   BEGIN OF ty.crm.prod_material,
      CRM.PROD type /BI0/OICRM_PROD,
      MATERIAL type /BI0/OIMATERIAL,
   END OF ty.crm.prod_material.

DATA:
   l.crm.prod_material type ty.crm.prod_material.
   l.t.crm.prod_material type sorted table of ty.crm.prod_material
      with unique key MATERIAL.
   i.min_value type DATS value '99993112'.
   i.max_value type DATS value '00000000'.
   l.t.point_scheme type zlom.t.point_scheme.
   w_point_scheme type zlom.s_point_scheme.
   lom_infocube type rsinfocube value 'ZCRMLOM'.
   i.crm.mktelm_char type rsiobjnm value '0CRM_MKTELM'.
   i.start_char type rsiobjnm value '0CRM_PLFR'.
   i.end_char type rsiobjnm value '0CRM_PLTO'.
   i.bpartner_char type rsiobjnm value '0BPARTNER'.
   i.crm.prod_char type rsiobjnm value '0CRM_PROD'.
   i.bonusamount_keyf type rsiobjnm value 'ZBONUSA'.
   i.bonusquantity_keyf type rsiobjnm value 'ZBONUSP'.
   i.layoutflag_keyf type rsiobjnm value '0KEYFG_FLAG'.
   i.no_rules type I.
   i.tmp type I.

TYPES:
   BEGIN OF DATA_PACKAGE_STRUCTURE.
   INCLUDE STRUCTURE /BIC/CS80LOM_DS12.
   END OF DATA_PACKAGE_STRUCTURE.

DATA:
   DATA_PACKAGE TYPE STANDARD TABLE OF DATA_PACKAGE_STRUCTURE
      WITH HEADER LINE
      WITH NON-UNIQUE DEFAULT KEY initial size 0.

FORM startup
TABLES   MONITOR STRUCTURE RSMONITOR "user defined monitoring
         MONITOR_RECNO STRUCTURE RSMONITORS " monitoring with record n
DATA_PACKAGE STRUCTURE DATA_PACKAGE
USING    RECORD_ALL LIKE SY-TABIX
         SOURCE_SYSTEM LIKE RSUPDSIMULH-LOGSYS
CHANGING ABORT LIKE SY-SUBRC. *set ABORT <> 0 to cancel update
*\$ begin of routine - insert your code only below this line.\*$

* fill the internal tables "MONITOR" and/or "MONITOR_RECNO",
* to make monitor entries

* retrieve crm_product from material
refresh l_t_crm_prod_material.
select
  material crm_prod from /BI0/PMATERIAL
into corresponding fields of table l_t_crm_prod_material
for all entries in DATA_PACKAGE
where material = DATA_PACKAGE-material.

* retrieve rules from planning InfoCube
refresh l_t_point_scheme.
loop at DATA_PACKAGE.
  if DATA_PACKAGE-calday lt i_min_value.
    i_min_value = DATA_PACKAGE-calday.
  endif.
  if DATA_PACKAGE-calday gt i_max_value.
    i_max_value = DATA_PACKAGE-calday.
  endif.
endloop.
endloop.

CALL FUNCTION 'ZLOM_POINT_CALCULATION'
EXPORTING
  I_INFOCUBE               - lom_infocube
  I_CRM_MKTELM_CHAR       - i_crm_mktelm_char
  I_START_CHAR            - i_start_char
  I_END_CHAR              - i_end_char
  I_CRM_PROD_CHAR         - i_crm_prod_char
  I_START_VALUE           - i_max_value
  I_END_VALUE             - i_min_value
  I_BONUSAMOUNT_KEYF      - i_bonusamount_keyf
  I_BONUSQUANTITY_KEYF    - i_bonusquantity_keyf
  I_LAYOUTFLAG_KEYF       - i_layoutflag_keyf
TABLES
  T_POINT_SCHEME          - l_t_point_scheme
EXCEPTIONS
  INFOCUBE_ERROR          = 1
  CRM_MKTELM_ERROR       = 2
  START_ERROR             = 3
  END_ERROR               = 4
  CRM_CAMPTYP_ERROR       = 5
  CRM_VERSION_ERROR       = 6
  BPARTNER_ERROR          = 7
  CRM_PROD_ERROR          = 8
  BONUSAMOUNT_ERROR       = 9
BONUSQUANTITY_ERROR = 10
LAYOUTFLAG_ERROR = 11
READ_INFOCUBE_ERROR = 12
OTHERS = 13.

IF SY-SUBRC <> 0.
  * if abort is not equal zero, the update process will be canceled
  ABORT = 1.
ENDIF.

describe table l_t_point_scheme lines i_no_rules.

loop at DATA_PACKAGE.
  i_tmp = i_no_rules.
  while i_tmp gt 1.
    insert DATA_PACKAGE.
    i_tmp = i_tmp - 1.
  endwhile.
endloop.

*$*$ end of routine - insert your code only before this line         *-*

ENDFORM.
11.5 Update Rules (2): Update Routine for ZLOMPTD “Loyalty Points”

PROGRAM UPDATE_ROUTINE.
*$$ begin of global - insert your declaration only below this line *.*
types: begin of ty_crm_prod_material,
  crm_prod type /BI0/OICRM_PROD,
  material type /BI0/OIMATERIAL,
end of ty_crm_prod_material.
data: l_crm_prod_material type ty_crm_prod_material.
data: l_t_crm_prod_material type sorted table of ty_crm_prod_material
  with unique key material.
data: i_min_value type dats value '99993112'.
data: i_max_value type dats value '00000000'.
data: l_t_point_scheme type zlom_t_point_scheme.
data: w_point_scheme type zlom_s_point_scheme.
data: lom_infocube type rsinfocube value 'ZCRMLOM'.
data: i_crm_mktelm_char type rsiobjnm value '0CRM_MKTELM'.
data: i_start_char type rsiobjnm value '0CRM_PLFR'.
data: i_end_char type rsiobjnm value '0CRM_PLTO'.
data: i_bpartner_char type rsiobjnm value '0BPARTNER'.
data: i_crm_prod_char type rsiobjnm value '0CRM_PROD'.
data: i_bonusamount_keyf type rsiobjnm value 'ZBONUSA'.
data: i_bonusquantity_keyf type rsiobjnm value 'ZBONUSP'.
data: i_layoutflag_keyf type rsiobjnm value '0KEYFG_FLAG'.
data: i_no_rules type i.
data: i_tmp type i.
*$$ end of global - insert your declaration only before this line   *-*

FORM compute_data_field
    TABLES   MONITOR STRUCTURE RSMONITOR "user defined monitoring
    USING    COMM_STRUCTURE LIKE /BIC/CS80LOM_DS12
        RECORD_NO LIKE SY-TABIX
        RECORD_ALL LIKE SY-TABIX
        SOURCE_SYSTEM LIKE RSUPDSIMULH-LOGSYS
    CHANGING RESULT LIKE /BIC/VZLOMPTST-/BIC/ZLOMPTD
    RETURNCODE LIKE SY-SUBRC
    ABORT LIKE SY-SUBRC. *set ABORT <> 0 to cancel update
*
*$$ begin of routine - insert your code only below this line   *.*
* fill the internal table "MONITOR", to make monitor entries
*
RESULT = 0.
RETURNCODE = 0.
if i_no_rules eq 0.
* if the returncode is not equal zero, the result will not be updated
RETURNCODE = 1.
else.
* result value of the routine
  i_tmp = RECORD_NO mod i_no_rules.
  i_tmp = i_tmp + 1.
  clear w_point_scheme.
* at the moment no explicit treatment of different currencies
  read table l_t_point_scheme into w_point_scheme index i_tmp.
  if w_point_scheme-start_date le COMM_STRUCTURE-calday and
     w_point_scheme-end_date ge COMM_STRUCTURE-calday and
     w_point_scheme-bpartner is initial.
* at the moment no campaigns with specific stores
* also a more elaborate treatment would exclude specific products
* because they are not eligible for loyalty points
  if w_point_scheme-crm_prod is initial.
* rule applies to all products
  RESULT = COMM_STRUCTURE-rt_salresa * w_point_scheme-quantity
           + COMM_STRUCTURE-rt_possal * w_point_scheme-amount.
else.
  * rule applies to a specific CRM Product, so check if the Material
  * corresponds to the CRM Product
  read table l_t.crm_prod_material into l.crm_prod_material
  with key material = COMM_STRUCTURE-material.
  if sy-subrc = 0.
   * CRM Product found, so now check if the rule applies:
     if l.crm_prod_material-crm_prod = w_point_scheme-crm_prod.
       RESULT = COMM_STRUCTURE-rt_salresa * w_point_scheme-quantity
                 + COMM_STRUCTURE-rt_possal * w_point_scheme-amount.
     else.
       RETURNCODE = 1.
     endif.
  else.
    RETURNCODE = 1.
  endif.
  endif.
  RETURNCODE = 1.
  endif.
  endif.
  endif.
  endif.
* if abort is not equal zero, the update process will be canceled
  ABORT = 0.

$*$ end of routine - insert your code only before this line $*-$
*

ENDFORM.

11.6 Update Rules (3): Update routine for ORT_P ossal “Sales
(POS Receipt)”
(This routine is only necessary for verification and demonstration)
PROGRAM UPDATE_ROUTINE.
$*$ begin of global - insert your declaration only below this line $*-$
types: begin of ty.crm_prod_material,
FORM compute_data_field
    TABLES MONITOR STRUCTURE RSMONITOR "user defined monitoring
    USING COMM_STRUCTURE LIKE /BIC/CS80LOM_DS12
    RECORD_NO LIKE SY-TABIX
    RECORD_ALL LIKE SY-TABIX
    SOURCE_SYSTEM LIKE RSUPDSIMULH-LOGSYS
    CHANGING RESULT LIKE /BIC/VZLOMPTST-RT_POSSAL
    RETURNCODE LIKE SY-SUBRC
    ABORT LIKE SY-SUBRC. * set ABORT <> 0 to cancel update
*"$*$ begin of routine - insert your code only below this line *. *
* fill the internal table "MONITOR", to make monitor entries

RESULT = 0.
RETURNCODE = 0.
if i_no_rules eq 0.
* if the returncode is not equal zero, the result will not be updated
    RETURNCODE = 1.
else.
* result value of the routine
    i_tmp = RECORD_NO mod i_no_rules.
    i_tmp = i_tmp + 1.
    clear w_point_scheme.
* at the moment no explicit treatment of different currencies
read table l_t_point_scheme into w_point_scheme index i_tmp.
if w_point_scheme-start_date le COMM_STRUCTURE-calday and
  w_point_scheme-end_date ge COMM_STRUCTURE-calday and
  w_point_scheme-bpartner is initial.
* at the moment no campaigns with specific stores
* also a more elaborate treatment would exclude specific products
* because they are not eligible for loyalty points
  if w_point_scheme-crm_prod is initial.
* rule applies to all products
  RESULT = COMM_STRUCTURE-rt_possal.
else.
* rule applies to a specific CRM Product, so check if the Material
* corresponds to the CRM Product
  read table l_t_crm_prod_material into l_crm_prod_material
  with key material = COMM_STRUCTURE-material.
  if sy-subrc = 0.
* CRM Product found, so now check if the rule applies:
    if l_crm_prod_material-crm_prod = w_point_scheme-crm_prod.
      RESULT = COMM_STRUCTURE-rt_possal.
    else.
      RETURNCODE = 1.
    endif.
  else.
    RETURNCODE = 1.
  endif.
else.
  RETURNCODE = 1.
endif.
else.
  RETURNCODE = 1.
endif.
else.
  RETURNCODE = 1.
endif.
* if abort is not equal zero, the update process will be canceled
ABORT = 0.

*$*$ end of routine - insert your code only before this line  *-*
*  
ENDFORM.

11.7 Update Rules (4): Update routine for ORT_SALRESA
“Quantity in Sales Unit (POS Rcpt)”
(This routine is only necessary for verification and demonstration)
PROGRAM UPDATE_ROUTINE.
*$*$ begin of global - insert your declaration only below this line  *-*
types: begin of ty_crm_prod_material,
  crm_prod type /BI0/OICRM_PROD,
  material type /BI0/OIMATERIAL,
end of ty_crm_prod_material.
data: l_crm_prod_material type ty_crm_prod_material.
data: l_t_crm_prod_material type sorted table of ty_crm_prod_material
    with unique key material.
FORM compute_data_field
   TABLES   MONITOR STRUCTURE RSMONITOR "user defined monitoring
   USING    COMM_STRUCTURE LIKE /BIC/CS80LOM_DS12
      RECORD_NO LIKE SY-TABIX
      RECORD_ALL LIKE SY-TABIX
      SOURCE_SYSTEM LIKE RSUPDSIMULH-LOGSYS
      CHANGING RESULT LIKE /BIC/VZLOMPTST-RT_SALRESA
      RETURNCODE LIKE SY-SUBRC
      ABORT LIKE SY-SUBRC. "set ABORT <> 0 to cancel update

   i_no_rules = 0.
   RETURNCODE = 0.
   if i_no_rules eq 0.
   * if the returncode is not equal zero, the result will not be updated
      RETURNCODE = 1.
   else.
   * result value of the routine
      i_tmp = RECORD_NO mod i_no_rules.
      i_tmp = i_tmp + 1.
      clear w_point_scheme.
      * at the moment no explicit treatment of different currencies
      read table l_t_point_scheme into w_point_scheme index i_tmp.
      if w_point_scheme-start_date le COMM_STRUCTURE-calday and
         w_point_scheme-end_date ge COMM_STRUCTURE-calday and
         w_point_scheme-bpartner is initial.
      * at the moment no campaigns with specific stores
      * also a more elaborate treatment would exclude specific products
      * because they are not eligible for loyalty points
   endif.
   endif.
   *$*$ begin of routine - insert your code only below this line   *-*
   * fill the internal table "MONITOR", to make monitor entries

RESULT = 0.
RETURNCODE = 0.
if i_no_rules eq 0.
* if the returncode is not equal zero, the result will not be updated
   RETURNCODE = 1.
else.
* result value of the routine
   i_tmp = RECORD_NO mod i_no_rules.
   i_tmp = i_tmp + 1.
   clear w_point_scheme.
* at the moment no explicit treatment of different currencies
   read table l_t_point_scheme into w_point_scheme index i_tmp.
   if w_point_scheme-start_date le COMM_STRUCTURE-calday and
      w_point_scheme-end_date ge COMM_STRUCTURE-calday and
      w_point_scheme-bpartner is initial.
* at the moment no campaigns with specific stores
* also a more elaborate treatment would exclude specific products
* because they are not eligible for loyalty points

*$*$ end of global - insert your declaration only before this line   *$*
if w_point_scheme-crm_prod is initial.
* rule applies to all products
  RESULT = COMM_STRUCTURE-rt_salresa.
else.
* rule applies to a specific CRM Product, so check if the Material
* corresponds to the CRM Product
  read table l_t_crm_prod_material into l_crm_prod_material
  with key material = COMM_STRUCTURE-material.
  if sy-subrc = 0.
    * CRM Product found, so now check if the rule applies:
      if l_crm_prod_material-crm_prod = w_point_scheme-crm_prod.
        RESULT = COMM_STRUCTURE-rt_salresa.
      else.
        RETURNCODE = 1.
      endif.
    else.
      RETURNCODE = 1.
    endif.
    endif.
  else.
    RETURNCODE = 1.
  endif.
* if abort is not equal zero, the update process will be canceled
  ABORT = 0.

*$*$ end of routine - insert your code only before this line   *-*
*
ENDFORM.

11.8 Update Rules (5): Update Routine for the Business
Partner

This routine computes the Business Partner from the Card Id. The same methods is applies in "Customer
Loyalty Marketing"

PROGRAM UPDATE_ROUTINE.
*$*$ begin of global - insert your declaration only below this line *-*
types: begin of ty_crm_prod_material,
  crm_prod type /BI0/OICRM_PROD,
  material type /BI0/OIMATERIAL,
end of ty_crm_prod_material.

data: l_crm_prod_material type ty_crm_prod_material.
data: l_t_crm_prod_material type sorted table of ty_crm_prod_material
  with unique key material.

data: i_min_value type datS value '99993112'.
data: i_max_value type datS value '00000000'.
data: l_t_point_scheme type zlom_t_point_scheme.
data: w_point_scheme type zlom_s_point_scheme.
data: lom_infocube type rsinfocube value 'ZCRMLOM'.
data: i_crm_mktelel_char  type rsobjnm value '0CRM_MKTELM'.
data: i_start_char        type rsobjnm value '0CRM_PLFR'.

data: i_end_char type rsiobjnm value '0CRM_PLTO'.
data: i_bpartner_char type rsiobjnm value '0BPARTNER'.
data: i_crm_prod_char type rsiobjnm value '0CRM_PROD'.
data: i_bonusamount_keyf type rsiobjnm value 'ZBONUSA'.
data: i_bonusquantity_keyf type rsiobjnm value 'ZBONUSP'.
data: i_layoutflag_keyf type rsiobjnm value '0KEYFG_FLAG'.
data: i_no_rules type i.
data: i_tmp type i.

* set ABORT <> 0 to cancel update

* fill the internal table "MONITOR", to make monitor entries

CALL FUNCTION 'RS_BCT_LOM_MAP_BP_ASSIGN'
EXPORTING
  BP_ID_NUM  = COMM_STRUCTURE-BP_ID_NUM
  BP_ID_TYPE = COMM_STRUCTURE-BP_ID_TYPE
IMPORTING
  BPARTNER   = RESULT
  RCODE      = RETURNCODE
  ABORT      = ABORT.

if RESULT is initial.
  RETURNCODE = 1.
endif.

ENDFORM.

Update Rules for 0CRM_MKTELM “CRM Marketing Element”
(This routine is for the assignment of campaigns to loyalty points):
PROGRAM UPDATE_ROUTINE.

* begin of global - insert your declaration only below this line *-*
types: begin of ty_crm_prod_material,
  crm_prod type /BI0/OICRM_PROD,
  material type /BI0/OIMATERIAL,
end of ty_crm_prod_material.

data: l_crm_prod_material type ty_crm_prod_material.
data: l_t_crm_prod_material type sorted table of ty_crm_prod_material.

* end of routine - insert your code only before this line *-*

ENDFORM.
with unique key material.

data: i_min_value type dats value '99993112'.
data: i_max_value type dats value '00000000'.
data: l_t_point_scheme type zlom_t_point_scheme.
data: w_point_scheme type zlom_s_point_scheme.
data: lom_infocube type rsinfocube value 'ZCRMLOM'.
data: i.crm_mktelm_char type rsiobjnm value '0CRM_MKTELM'.
data: i_start_char type rsiobjnm value '0CRM_PLFR'.
data: i_end_char type rsiobjnm value '0CRM_PLTO'.
data: i_bpartner_char type rsiobjnm value '0BPARTNER'.
data: i.crm_prod_char type rsiobjnm value '0CRM_PROD'.
data: i_bonussamount_keyf type rsiobjnm value 'ZBONUSA'.
data: i.bonussquantity_keyf type rsiobjnm value 'ZBONUSP'.
data: i.layoutflag_keyf type rsiobjnm value '0KEYFG_FLAG'.
data: i_no_rules type i.
data: i_tmp type i.

*$*$ end of global - insert your declaration only before this line

FORM compute_key_field
  TABLES MONITOR STRUCTURE RSMONITOR "user defined monitoring
  USING COMM_STRUCTURE LIKE /BIC/CS80LOM_DS12
  RECORD_NO LIKE SY-TABIX
  RECORD_ALL LIKE SY-TABIX
  SOURCE_SYSTEM LIKE RSUPDSIMULH-LOGSYS
  CHANGING RESULT LIKE /BIC/VZLOMPTST-CRM_MKTELM
  RETURNCODE LIKE SY-SUBRC
  ABORT LIKE SY-SUBRC. "set ABORT <> 0 to cancel update
*
*$*$ begin of routine - insert your code only below this line
*
* fill the internal table "MONITOR", to make monitor entries
if i_no_rules eq 0.
* if the returncode is not equal zero, the result will not be updated
  RETURNCODE = 1.
else.
  i_tmp = RECORD_NO mod i_no_rules.
i_tmp = i_tmp + 1.
clear w_point_scheme.
read table l_t_point_scheme into w_point_scheme index i_tmp.
* result value of the routine
  RESULT = w_point_scheme-crm_mktelm.
endif.
* if abort is not equal zero, the update process will be canceled
  ABORT = 0.
*$*$ end of routine - insert your code only before this line
*
* ENDFORM.
www.sdn.sap.com/irj/sdn/howtoguides