How to Use Definitions in Rules Composer

Applies to:
SAP NetWeaver Business Rules Management. For more information, visit the Business Rules Management homepage.

Summary
This tutorial helps you work with Definitions in the Rules Composer. A business use case has been created for the purpose.

Given details such as buyer’s credit information, buyer’s spending habit and set the discount based on this information.

This tutorial guides you to create Fixed Definitions and Variable Definitions and use them in IF-Then rules and to set the discount based on this information.

Authors: Arti Gopalan and Sarada Ustili
Company: SAP Labs, India
Created on: 28 May 2008
Table of Contents

Prerequisites.......................................................................................................................................................3

Knowledge Required.......................................................................................................................................3

Software Requirements...................................................................................................................................3

Procedure ...........................................................................................................................................................3

Creating the Rules Composer DC ..................................................................................................................3

Creating the Ruleset .......................................................................................................................................3

Creating Definitions.........................................................................................................................................4

Creating the Fixed Definition ..........................................................................................................................4

Creating the Variable Definition ....................................................................................................................4

Adding the Classes .........................................................................................................................................5

Creating the Web Module ...............................................................................................................................5

Creating Classes in the Web Module ................................................................................................................6

Defining the Public Parts ....................................................................................................................................9

Adding Dependencies to the Rules Composer DC ..........................................................................................11

Adding Java Classes to the Rules Composer DC ............................................................................................12

Renaming the Class Aliases ............................................................................................................................15

Creating the Rules ........................................................................................................................................15

Creating the GoodCreditBuyer rule ..................................................................................................................15

Creating the BadCreditBuyer rule ....................................................................................................................19

Creating the ChkSetDiscount rule ...................................................................................................................20

Deploying the Rules ........................................................................................................................................21

Executing the Rules .......................................................................................................................................22

Creating the Web Module ...............................................................................................................................22

Adding Dependency to the Web Module ...........................................................................................................22

Creating the Enterprise Application ................................................................................................................22

Adding Dependencies to the Enterprise Application ......................................................................................23

Creating the application.xml ...........................................................................................................................23

Building and Deploying ................................................................................................................................24

Running the Web Module ..............................................................................................................................24

Related Content.............................................................................................................................................27

Copyright..........................................................................................................................................................28
Prerequisites

Knowledge Required
- You have basic knowledge in rules modeling
- You are familiar with Business Rules Management System

Software Requirements
- You work in the SAP NetWeaver Developer Studio
- Your SAP NetWeaver Developer Studio version includes the Rules Composer perspective
- You should have a running instance of SAP AS, and should have configured the SAP NetWeaver Developer Studio with this instance

Note: In the SAP NetWeaver Developer Studio, choose Window -> Open Perspective -> Other. In the dialog box that appears, choose Rules Composer and choose OK.

Procedure

Creating the Rules Composer DC
1. In the SAP NetWeaver Developer Studio, choose File -> New -> Project.
2. In the wizard that appears, expand the Rules Composer node and choose Rules Composer Development Component. Choose Next.
3. In the screen that appears, choose the software component where you want to create the DC. For example the software component could be MyComponents [demo.sap.com] under the ‘Local Development’ node. Choose Next.
4. In the screen that appears, enter buyerrules in the Name field and choose Finish.
You should see the buyerrules node in the Project Explorer view.

Creating the Ruleset
1. In the Project Explorer view, expand the Rules Composer DC:buyerrules node and in the context menu of the Rules Modeling node, choose New Ruleset.
2. In the dialog box that appears, enter DiscountRules in the field. Choose OK.
You should see the ruleset: DiscountRules window with the Overview tab page open.

If the DiscountRules window does not appear, in the Project Explorer view, expand the Rules Composer DC:buyerrules node, the Rules Modeling node and double-click the DiscountRules node.
How to Use Definitions in Rules Composer

Creating Definitions

Creating the Fixed Definition

1. In the DiscountRules window, choose the Definitions tab.
2. In the Definitions Editor, under Fixed Definitions section, choose Add icon, and in the drop down that appears, choose double.
3. In the dialog box that appears, enter summerDiscount definition and choose OK. The definition double summerDiscount = 0 appears.
4. Choose RValue: 0 and in the drop down menu that appears enter 10 in the inline text box.
5. Repeat the steps 1 to 4 and two more Fixed Definitions such as double goodBuyerDiscount = 15, double badBuyerDiscount = 5.

The result must be as follows:

Creating the Variable Definition

1. In the DiscountRules window, choose the Definitions tab.
2. In the Definitions Editor, under the Variable Definitions section, choose Add icon and in the drop down that appears, choose double.
3. In the dialog box that appears, enter `buyerDiscount` and choose `OK`. The definition `double buyerDiscount = 0` appears as shown below:

![Image of the dialog box with the definition `double buyerDiscount = 0`]

### Adding the Classes

#### Creating the Web Module

1. In the SAP NetWeaver Developer Studio, choose `File -> New -> Project`.
2. In the wizard that appears, expand the `Development Infrastructure` node and choose `Development Component`. Choose `Next`.
3. In the screen that appears, expand `J2EE` node and choose `Web Module`. Choose `Next`.
4. In the screen that appears, choose the software component where you want to create the DCs.
   - For example, expand the `Local Development` node and choose `MyComponents [demo.sap.com]`. Choose `Next`.  

---

© 2008 SAP AG

SAP DEVELOPER NETWORK | sdn.sap.com

BUSINESS PROCESS EXPERT COMMUNITY | bpx.sap.com
5. In the screen that appears, enter `buyer_wm` in the *Name* field and choose *Finish*.

The *Java EE perspective* opens, and you should see the `buyer_wm` node in the *Project Explorer* view as shown below:

![Java EE perspective with buyer_wm node](image)

**Note:** If the *Project Explorer* view does not open, choose *Window* -> *Show View* -> *Other* and in the dialog box that appears expand *General* node and choose *Project Explorer*. Choose *OK*.

---

### Creating Classes in the Web Module

Make sure you are in the *Java EE* perspective.

1. In the *Project Explorer* view, expand the `buyer_wm` node, and in the context menu of *Java Resources: source* node, choose *New* -> *Class*.

**Note:** If you do not see the *Class* option in the context menu of the *Java Resources: source* node, choose *New* -> *Other*. In the dialog box that appears, choose *Class*.

2. In the screen that appears, enter `com.sap.buyer` in the *Package* field.

3. Enter *Buyer* in the *Name* field and choose *Finish*.

You should see the `com.sap.buyer` node under the `src` node.
4. Double-click *Buyer.java* under the *com.sap.buyer* node.

The Buyer.java window appears with the lines as shown below:

```java
package com.sap.buyer;
import java.io.Serializable;
public class Buyer implements Serializable {
    public Buyer() {
    }
    String buyersname;
    String buyersspendinghabit;
    String buyerscredit;
    double setdiscount;
    public String getBuyersname() {
        return buyersname;
    }
    public void setBuyersname(String buyersname) {
        this.buyersname = buyersname;
    }
}
```

5. Delete all the existing lines and copy the following lines into the window:

```java
package com.sap.buyer;
import java.io.Serializable;
public class Buyer implements Serializable {
    public Buyer() {
    }
    String buyersname;
    String buyersspendinghabit;
    String buyerscredit;
    double setdiscount;

    public String getBuyersname() {
        return buyersname;
    }
    public void setBuyersname(String buyersname) {
        this.buyersname = buyersname;
    }
}
```
public String getBuyersspendinghabit() {
    return buyersspendinghabit;
}
public void setBuyersspendinghabit(String buyersspendinghabit) {
    this.buyersspendinghabit = buyersspendinghabit;
}

public String getBuyerscredit() {
    return buyerscredit;
}
public void setBuyerscredit(String buyerscredit) {
    this.buyerscredit = buyerscredit;
}

public double getSetdiscount() {
    return setdiscount;
}
public void setSetdiscount(double setdiscount) {
    this.setdiscount = setdiscount;
}

6. Press Ctrl+Shift+F.
You should see the result as shown below:

7. Save the changes.

**Defining the Public Parts**

Make sure you are in the **Development Infrastructure** perspective.

1. In the **Component Browser** view, expand the **Local Development** node, **MyComponents [demo.sap.com]** node (the software component in which Development Component has been created), and choose the **buyer_wm** node.

   **Note:** If the **Component Browser** view is not open, choose **Window -> Show View -> Other** and in the dialog box that appears expand the **Development Infrastructure** node and choose **Component Browser**. Choose **OK**.

2. In the **Component Properties** view, choose **Public Parts** and in the page that appears, choose **Add**.
3. In the screen that appears, enter public in the Name field and choose Finish.
You should see the public node under the Defined Public Parts section as shown below:

4. In the context menu of the public node, choose Manage Entities.
5. In the screen that appears, under the *Entities* section, expand the *Java Class, com, sap and buyer* nodes, select the *Buyer java class* checkbox as shown below:

![Manage Entities dialog box](image)

6. Choose *Finish*.

**Adding Dependencies to the Rules Composer DC**

1. In the *Component Browser* view, expand the *Local Development node, MyComponents [demo.sap.com] node* (the software component in which Development Component has been created) and double-click the *buyerrules node*.

2. In the *Component Properties* view, choose *Dependencies* and in the page that appears, choose *Add*.

3. In the dialog box that appears, expand the *MyComponents node and select the buyer_wm checkbox. Choose Next.*
4. In the screen that appears, under **Dependency Details** section, select the **Design Time**, **Deploy Time** and **Run Time** checkboxes as shown below:

![Adding Dependencies](image)

5. Choose **Finish**.

6. In the **Dependencies** tab page, expand the **buyer_wm** node and choose the **war** node. Choose **Remove** and in the dialog box that appears choose **Yes**.

7. In the **Component Browser** view, expand the **MyComponents [demo.sap.com]** node and in the context menu of the **buyerrules** node, choose **Build**. Choose **OK**.

8. Check the **Infrastructure Console** for message.

**Adding Java Classes to the Rules Composer DC**

Make sure you are in the **Rules Composer** perspective.

1. In the **Project Explorer** view, expand the **buyerrules** node, the **Rules Modeling** node and double-click the **Aliases** node.

2. In the Project Aliases Editor that appears, choose the **Class Aliases** tab and in the tab page that appears, choose the **Add Classes** tab.
3. In the dialog box that appears, expand the `com.sap.buyer` node and double-click `Buyer`. The class appears under `Selected Classes` section as shown below:

5. In the *Aliases Name* section, expand the *Buyer* node and select all the relevant classes as shown below:

![Rules Composer screenshot](image)

6. Save the changes.
Renaming the Class Aliases

In the *Alias Name* table, click each of the aliases. The aliases become editable. Enter an alternative name for the alias.

<table>
<thead>
<tr>
<th>Alias Name</th>
<th>Rename as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buyer.getbuyerscredit</td>
<td>get buyers credit</td>
</tr>
<tr>
<td>Buyer.getBuyersname</td>
<td>get buyers name</td>
</tr>
<tr>
<td>Buyer.getBuyersspendinghabit</td>
<td>get buyers spending habit</td>
</tr>
<tr>
<td>Buyer.getSetdiscount</td>
<td>get set discount</td>
</tr>
<tr>
<td>Buyer.setBuyerscredit ( {String} )</td>
<td>set buyers credit {String}</td>
</tr>
<tr>
<td>Buyer.setBuyersname ( {String} )</td>
<td>set buyers name {String}</td>
</tr>
<tr>
<td>Buyer.setSetdiscount ( {double} )</td>
<td>set Set discount</td>
</tr>
</tbody>
</table>

Creating the Rules

Creating the GoodCreditBuyer rule

1. In the *Project Explorer* view, expand the Rules Composer DC:*buyerrules* node, the *Rules Modeling* node and in the context menu of the ruleset: *SetCallCharge* node, choose *New Rule*.

2. In the dialog box that appears, enter *GoodCreditBuyer* in the field and choose *OK*.

In the *Outline* view, the *GoodCreditBuyer* node appears as shown below:

3. In the Rule Editor that appears, under *if* section, choose the *Add* icon.

4. The default condition: *Operation.isSuccessful()* *Equals true* appears.
5. Edit the default condition as follows:
   a. Choose the LValue: `Operation.isSuccessful()` and in the drop down menu expand the **Buyer** node and choose **get Buyers credit**.
   b. Leave the comparator: **Equals** as it is.
   c. Choose the RValue: **Default Value** and in the inline textbox enter **GOOD**.
6. Under *Then* section, choose the *Add* icon and in the drop down menu that appears, expand the *Assign* node and choose `buyerDiscount(double)` as shown below:

![Image of Rules Composer with Assign node highlighted]

Assign : : buyerDiscount = 0 appears.

7. Choose 0 and in the drop down menu expand the *Definitions* node and choose `summerDiscount`. 
8. Choose `summerDiscount` and in the drop down menu choose `<Add a new expression>` as shown below:

9. Choose the value 0 and in the drop down menu expand Definitions node and choose `goodBuyerDiscount` as shown below:
10. Save the changes.

The result must be as shown below:

Creating the BadCreditBuyer rule

1. In the Project Explorer view, expand the Rules Composer DC: buyerrules node, the Rules Modeling node and in the context menu of the ruleset: discountrules node, choose New Rule.

2. In the dialog box that appears, enter BadCreditBuyer in the field and choose OK.

3. In the Outline view, choose the BadCreditBuyer node.

4. In the Rule Editor that appears, under If section, choose the Add icon that looks like a plus sign.

5. The default condition: Operation.isSuccessful Equals() true appears.

6. Edit the default condition as follows:
   a. Choose the LValue: Operation.isSuccessful() and in the drop down menu choose get Buyers credit.
   b. Leave the comparator :Equals as it is.
   c. Choose the RValue: Default Value and in the inline textbox enter BAD.

7. Under Then section, choose the Add icon and in the drop down menu that appears, expand the Assign node and choose buyerDiscount[double].

Assign :: buyerDiscount = 0 appears.

8. Choose the value 0 and in the drop down menu expand Definitions node and choose summerDiscount.
9. Choose *summerDiscount* and in the drop down menu choose &lt;Add a New Expression&gt;.

10. Choose the RValue: 0 and in the drop down menu expand the Definitions node and choose *badBuyerDiscount*.

11. Choose the mathematical operator (+) and in the drop down menu choose (-).

12. Save the changes.

You should see the BadCreditBuyer rule as shown below:

![BadCreditBuyer rule image]

Creating the ChkSetDiscount rule

1. In the Project Explorer view, expand the Rules Composer DC: *buyerrules* node, the Rules Modeling node and in the context menu of the ruleset: *discountrules* node, choose New Rule.

2. In the dialog box that appears, enter *ChkSetDiscount* in the field and choose OK.

   In the Outline view, the ChkSetDiscount node appears.

3. In the Rule Editor that appears, under *If* section, choose the Add icon.

4. The default condition: *Operation.isSuccessful()* Equals true appears.

5. Edit the default condition as follows:

   a. Choose the LValue: *Operation.isSuccessful()* and in the drop down menu choose *buyerDiscount*.

   b. Choose the comparator: *Equals* and in the drop down menu choose *Greater Than*.

      *buyerDiscount Greater Than 0* appears.

6. Under *Then* section, choose the Add icon and in the drop down menu that appears, choose Execute &lt;Method Name&gt;.

7. Choose &lt;Action Method&gt; and in the drop down menu choose *set Set discount* (*{double}*).

8. Choose *{double}* and in the drop down menu, expand Definitions node choose the variable definition *buyerDiscount*.

9. Save the changes.
How to Use Definitions in Rules Composer

You should see the ChkSetDiscount rule as shown below:

Deploying the Rules

1. In the Project Explorer view, in the context menu of the Rules Composer DC:buyerrules node, choose Development Component -> Build.

2. In the dialog box that appears, select the buyerrules checkbox and choose OK.

Note: If the Infrastructure Console is not open, choose Window -> Show View -> Other and in the dialog box that appears, expand the Development Infrastructure node and choose Infrastructure Console and then choose OK.

3. In the context menu of the Rules Composer DC: buyerrules node, choose Development Component -> Deploy.

4. In the dialog box that appears, select the buyerrules checkbox and choose OK.

Note: Open the Infrastructure Console, to check if the deployment has happened successfully.
Executing the Rules
In the SAP Netweaver Developer Studio choose Window -> Open Perspective -> JavaEE.

Creating the Web Module
We have already created a Web Module named buyer_wm.

Adding Dependency to the Web Module
Make sure you are in the Development Infrastructure perspective.

1. In the Component Browser view, expand the MyComponents[demo.sap.com] node and choose the buyer_wm node.
2. In the Component Properties view, choose the Dependencies tab.
3. Choose the Add button and in the wizard that appears, expand the BRMS-FACADE[sap.com] node and select the tc/brms/facade checkbox. Choose Next.
4. In the screen that appears, select the Design Time, Deploy Time, Run Time checkboxes. Choose Finish.

Note: In the context menu of the buyer_wm node, choose Sync / Create Project > Sync Used DCs. In the dialog box that appears, choose OK.

Unzip the project file and do the following.
1. Make sure that you are in the Java EE perspective
2. Expand the web module: buyer_wm node and in the context menu of the Java Resources: source node, choose New ->Other.
3. In the wizard that appears, expand the Java node and choose Package. Choose Next.
4. In the screen that appears, enter com.sap.helper in the Name field.
5. Choose Finish.
7. In the context menu of the Web Content node copy the following files: BuyerDemo.jsp, index.jsp, invoker.jsp.

Creating the Enterprise Application
1. In the SAP NetWeaver Developer Studio, choose File -> New -> Project.
2. In the wizard that appears, expand the Development Infrastructure node and choose Development Component. Choose Next.
3. In the screen that appears, expand the J2EE node and choose Enterprise Application. Choose Next.
4. In the screen that appears, choose the software component where you want to create the DCs. For example expand the 'Local Development' node and choose MyComponents [demo.sap.com]. Choose Next.
5. In the screen that appears, enter buyer_ear in the Name field. Choose Next.
6. Choose Next.
7. In the screen that appears, select the LocalDevelopment~LocalDevelopment~buyer_wm~demo.sap.com checkbox. Choose Finish.
8. In the Project Explorer view, you should see the buyer_ear node.
Adding Dependencies to the Enterprise Application

Make sure that you are in the Development Infrastructure perspective.

1. In the Component Browser view, expand the MyComponents[demo.sap.com] node and choose the buyer_ear node.
2. In the Component Properties view, choose the Dependencies tab.
3. Choose the Add button and in the wizard that appears, expand the BRMS-FACADE[sap.com] node and select the tc/brms/facade checkbox. Choose Next.
4. In the screen that appears, select the Design Time, Deploy Time, Run Time checkboxes. Choose Finish.
5. Choose the Add button again and in the wizard that appears, expand the My Components[demo.sap.com] node and select the buyer_wm checkbox. Choose Next.

Note: In the context menu of the buyer_ear node, choose Sync / Create Project > Sync Used DCs. In the dialog box that appears, choose OK.

Creating the application.xml

Make sure that you are in the Java EE perspective.

1. Expand the enterprise application: buyer_ear node and in the context menu of the Deployment Descriptor: LocalDevelopment~LocalDevelopment~buyer_ear~demo.sap.com node, choose Create application.xml.

Note: Expand the enterprise application: buyer_ear node, META-INF node double click the application.xml node.

You should see the application.xml window with the following lines:

```xml
<?xml version = "1.0" encoding = "ASCII"?>
  <display-name>LocalDevelopment~LocalDevelopment~buyer_ear~demo.sap.com</display-name>
  <module>
    <web>
      <web-uri>demo.sap.com~buyer_wm.war</web-uri>
      <context-root>LocalDevelopment~LocalDevelopment~buyer_wm~demo.sap.com</context-root>
    </web>
  </module>
</application>
```


Note: Instead of LocalDevelopment~LocalDevelopment~buyer_wm~demo.sap.com, you need to enter the customized application name i.e in this tutorial, the name of the application is BuyerRule.
Building and Deploying

Make sure you are in the Java EE perspective.

1. In the Project Explorer, in the context menu of the buyer_wm and buyer_ear nodes, choose Development Component -> Build.
2. In the dialog box that appears, choose OK.
3. In the context menu of the buyer_ear node, choose Development Component -> Deploy.
4. Check the Infrastructure Console for messages.

**Note:** You can also build and deploy in the Development Infrastructure perspective.

1. In the context menu of the buyer_wm and buyer_ear nodes, choose build.
2. In the dialog box that appears, choose OK.
3. In the context menu of the buyer_ear node choose Deploy.
4. In the dialog box that appears, choose OK.

Running the Web Module

1. Open the browser and enter the Application Server Address followed by the port number and the application name: **BuyerRule**.
2. Enter the following data in the respective fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>User Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of the Buyer</td>
<td>Tom</td>
</tr>
<tr>
<td>Spending Habit of the Buyer</td>
<td>LOW</td>
</tr>
<tr>
<td>Buyer’s Credit</td>
<td>GOOD</td>
</tr>
</tbody>
</table>

3. Choose Submit.

You should get the Discount Set as 25.0
Here is the snapshot of the web module:

Also try this:

1. Enter the following data in the respective fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>User Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of the Buyer</td>
<td>Tom</td>
</tr>
<tr>
<td>Spending Habit of the Buyer</td>
<td>LOW</td>
</tr>
<tr>
<td>Buyer’s Credit</td>
<td>BAD</td>
</tr>
</tbody>
</table>

2. Choose Submit.

You should get the Call Charges as 5.0.
Here is the snapshot of the web module:
Related Content

For more information, visit the Business Rules Management homepage.
Copyright

© 2008 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, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, POWER5+, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated 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.

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.