Wage Type Processing Basic Concepts

by Steve Bogner, Managing Partner, Insight Consulting Partners

Relationship of Payroll Driver, Schema (Functions), Rules (Operations)

Each country payroll version supported by SAP has a program called the "payroll driver" that calculates payrolls. For example, in the U.S., the payroll driver is RPCALCU0, in Mexico it is HMXCALC0, and in Canada RPCALCK0. Each one is different, but they share a common core of functionality. The job of the payroll driver is to process payroll functions as specified in a payroll schema. These payroll functions each perform a specific job, for example - reading data from infotypes, calculating taxes, and processing wage types. Some functions process payroll rules. Rules are a collection of payroll operations. Each operation does a small unit of work, such as multiplying a wage type's number by a rate to get an amount.

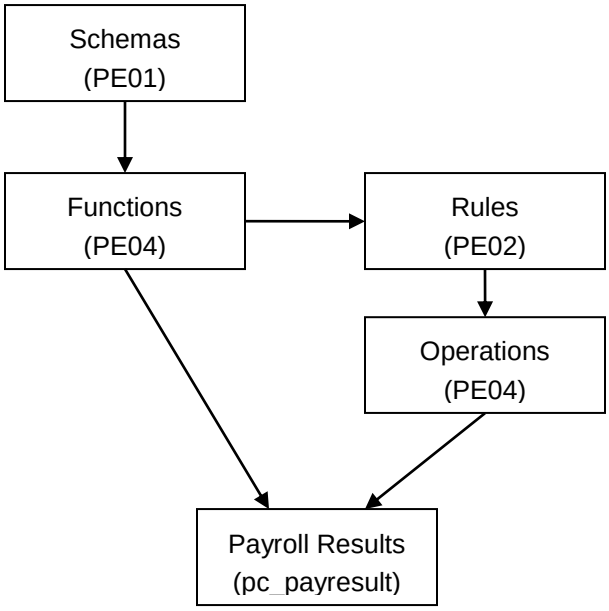

Figure 1 – Processing relationships

Schemas are edited with transaction PE01, and rules with PE02. Functions and operations are maintained with transaction PE04. To view payroll results, use transaction pc_payresult (or in earlier R/3 releases go to **Tools>Payroll result>Display in the Payroll menu**). (See **Figure 1**.)The standard payroll schema for a country can be derived from table t500l. If the country in table t500l has an **X** in the **Old Naming Conv** field, then the schema is **HR Country Indicator** plus **000**. Otherwise, it is the **ISO Code** plus **00**. So the U.S. has schema U000 and for Mexico it is MX00.

Header and Table Wage Type Concept

When calculating payroll, wage types are read from infotypes and the Time Management cluster and stored in an internal table called the Input Table (IT). (See **Table 1**.) In ABAP terms, this is simply an internal table. Various payroll functions and operations can read and update data in this table. Similar to ABAP internal tables, there is a header row. That header row defines which row of data can be accessed by the payroll operations. After manipulating the data in the header row, you can save the row back to the IT, save it to another payroll table, or ignore it. In Table 1 there are three wage types, and wage type 2100 is currently in the header row. After you are done with wage type 2100, wage type 4200 is moved into the header row.

Table 1			
Wage type	Number	Rate	Amount
2100	0	0	100.00
2100	0	0	100.00
4200	0	0	20.00
1500	40	10.30	412.00

Table 1 Input Table

Creating Custom Schemas and Rules

Schema and Rule Naming Conventions

Customer modified schemas and rules need to begin with **Z**. Many customers simply replace the first letter of the standard schema with a **Z** – i.e., their modified copy of UAP0 becomes ZAP0. But there can be problems with that convention. For example, you might later implement Canadian payroll and need a modified version of schema KAP0, but ZAP0 is already used for the U.S. For many years, I've used a naming convention of Z plus the country identifier and a sequential number for modified rules and schemas. So a modified UAP0 would become ZU01 and a modified KAP0 becomes ZK01.

Editor Documentation

Documentation for the function, operation, schema, and rule editors is available online at http://help.sap.com. Click on **SAP R/3 and R/3 Enterprise** and then select your release level and language. Then navigate to the **Human Resources>HR Tools** section.

F1 Help

In the schema and rule editors, place your cursor on a function or operation and press F1 to get help text. A schema or a rule's documentation is available in the editor via the **Goto>Documentation** menu. In the schema editor, the F4 key shows possible values for each of the four parameters for whatever function is entered on that line. The same documentation – and more – is available via transaction `PDSY`.

Creating a Test Schema

For testing purposes, it is useful to have a version of the payroll schema that does not care about the control record (transaction `PA03`) settings. Bypassing the control record lets you run and save the results for any pay period needed, without having to update the control record. There's no problem with having such a schema around, since the payroll driver does not save payroll results from a schema that ignores the control record in a production system. For examples, I will show you how to create two schemas – ZUA0, which will be used in production and will check the control record, and ZUAT, which ignores the control record and is used for testing purposes only.

First, create a copy of SAP's schema UIN0 and name it ZUA2. In the schema editor (transaction `PE02`) enter schema UIN0, and click the copy button, or **Schema>Copy** in the menu. Enter ZUA2 for the **To schema**. Then edit ZUA2 and make the **`CHECK ABR`** line executable by removing the asterisk in the **D** column. (See **Figure 2**.) The CHECK function is commented out by SAP in the standard schema, so you uncomment it here for use in the main ZUA0 schema.

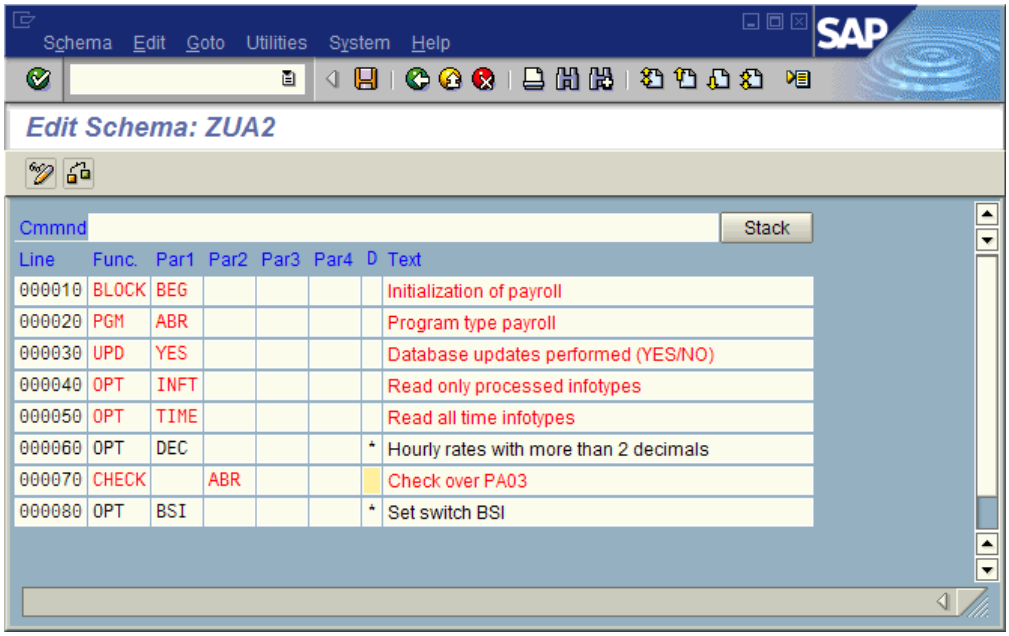

Figure 2 Making the CHECK ABR line executable

Now copy the SAP-standard schema U000 to ZUA1 and comment out the initialization schema UIN0 (**Figure 3**).

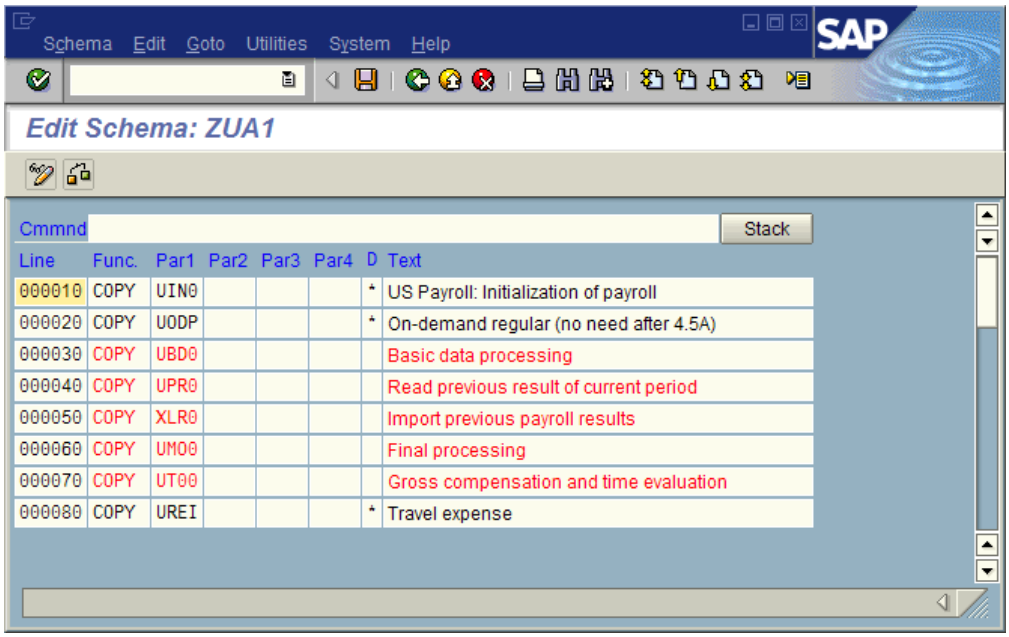

Figure 3 Comment out the initialization schema UIN0

In the schema editor, create the production schema (don't copy it from anything) in my example ZUA0. Be sure to check the **Schema can be executed** checkbox. (See **Figure 4**.) Only executable schemas can be entered into the payroll driver selection screen.

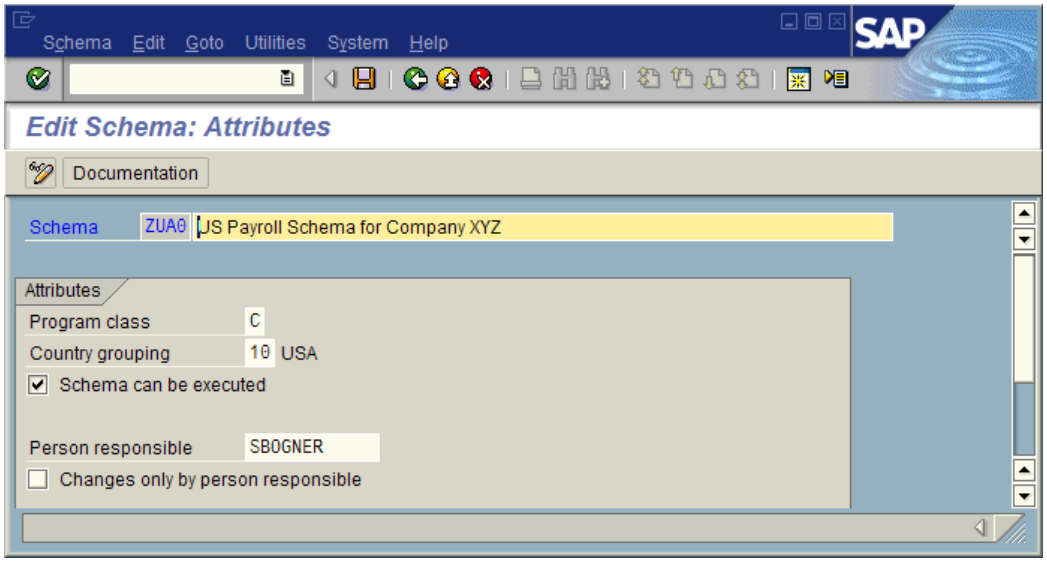

Figure 4 Check the **Schema can be executed** checkbox

The production schema ZUA0 is a simple one, just two lines. (See **Figure 5**.) First, you call the initialization schema, and then you call the main calculation schema ZUA1.

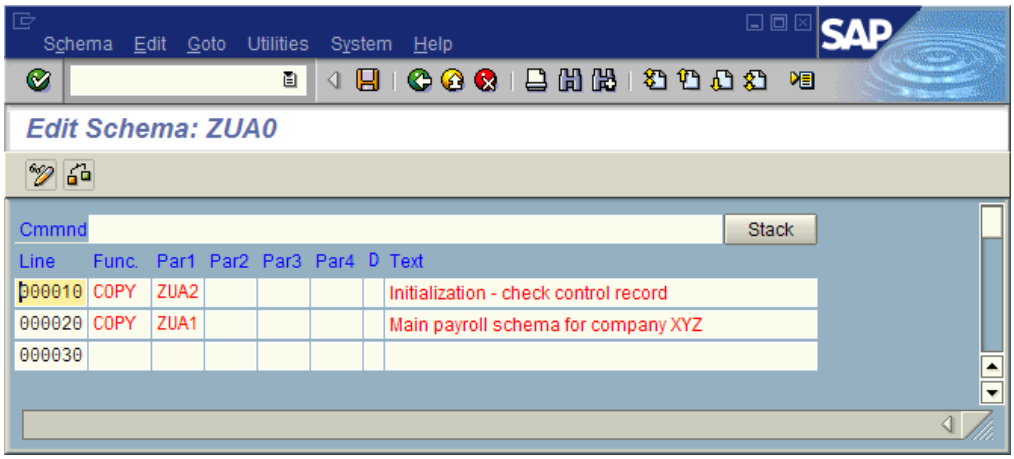

Figure 5 Schema ZUAO

Copy schema ZUA0 to your test schema ZUAT. (See **Figure 6**.) You want ZUAT to ignore the control record, so have it use schema UIN0 for initialization. Remember that `CHECK ABR` is commented out in UIN0. Therefore, both the production and test schemas now use the same calculation logic in schema ZUA1 – which keeps them in sync.

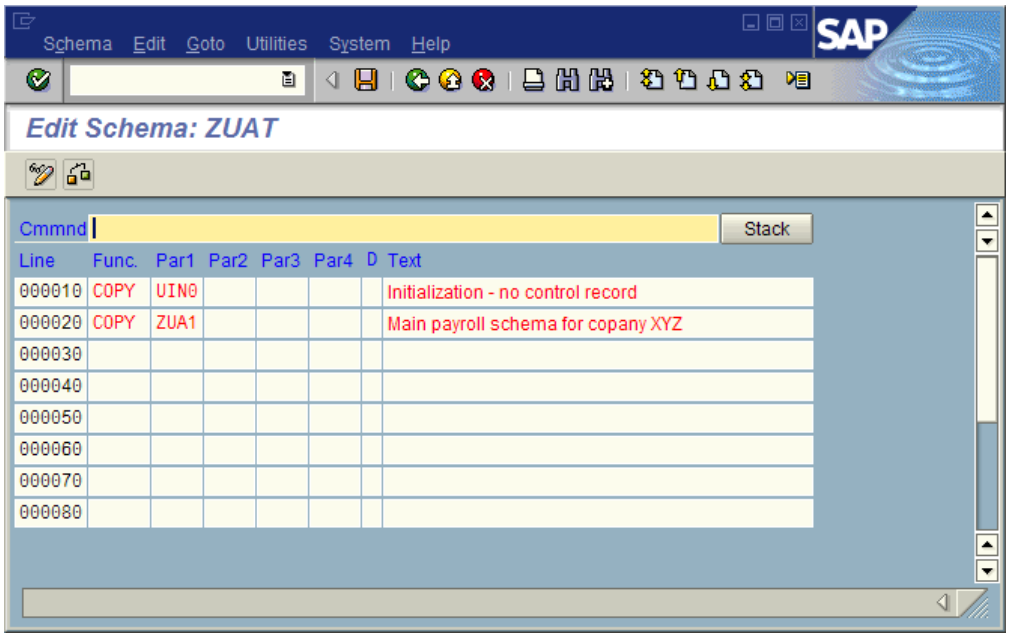

Figure 6 Copy schema ZUA0 to test schema ZUAT

Your custom rules for these examples will go in a copy of schema UAP0. Copy UAP0 to ZUA3 and add lines for each of the five examples. (See **Figure 7**.) Edit schema ZUA1 to `COPY ZUA3` instead of `COPY UAP0` (not shown).

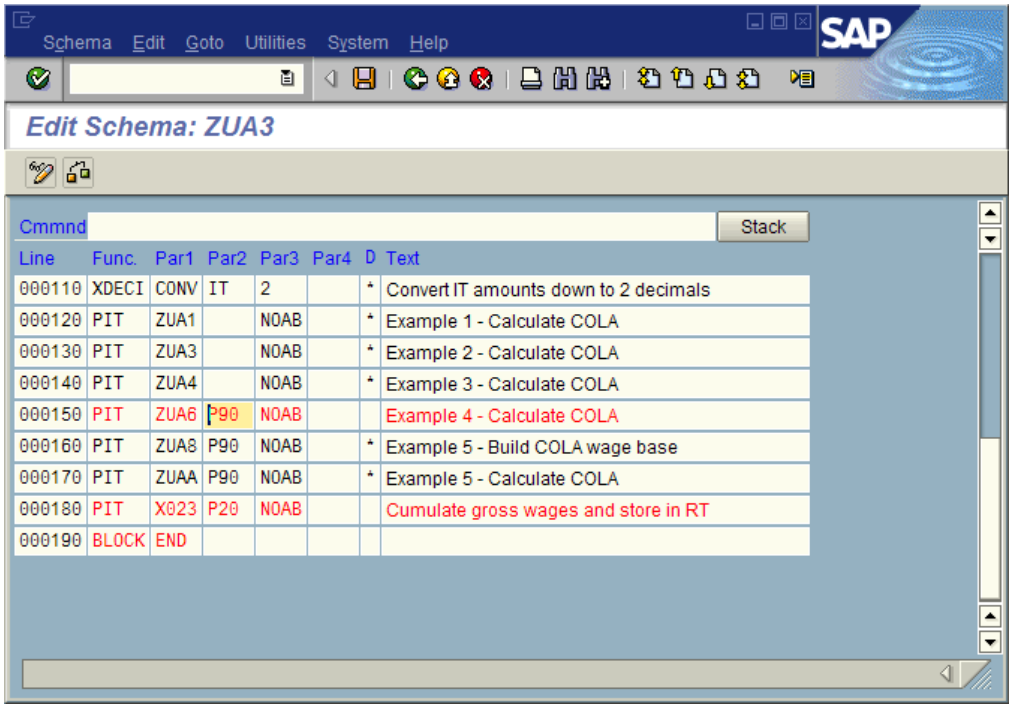

Figure 7 Add lines for each of the five examples

Schema and Rule Documentation

Creating Documentation Objects

Many customizing objects in Payroll can be documented online in SAP. The documentation is added to a transport so that it can be migrated to each client in the development, QA, and production environments. There are three ways to document an object online in SAP.

When changing schemas and rules, you can go directly to the **Documentation** section at the first screen of the editor. Or, while editing, use the **Goto>Documentation** menu. Either method takes you to an editor. The editor shown in **Figure 8** is from the Enterprise release, and the appearance and functions can change from one release to another. Rule ZUA0 has no documentation.

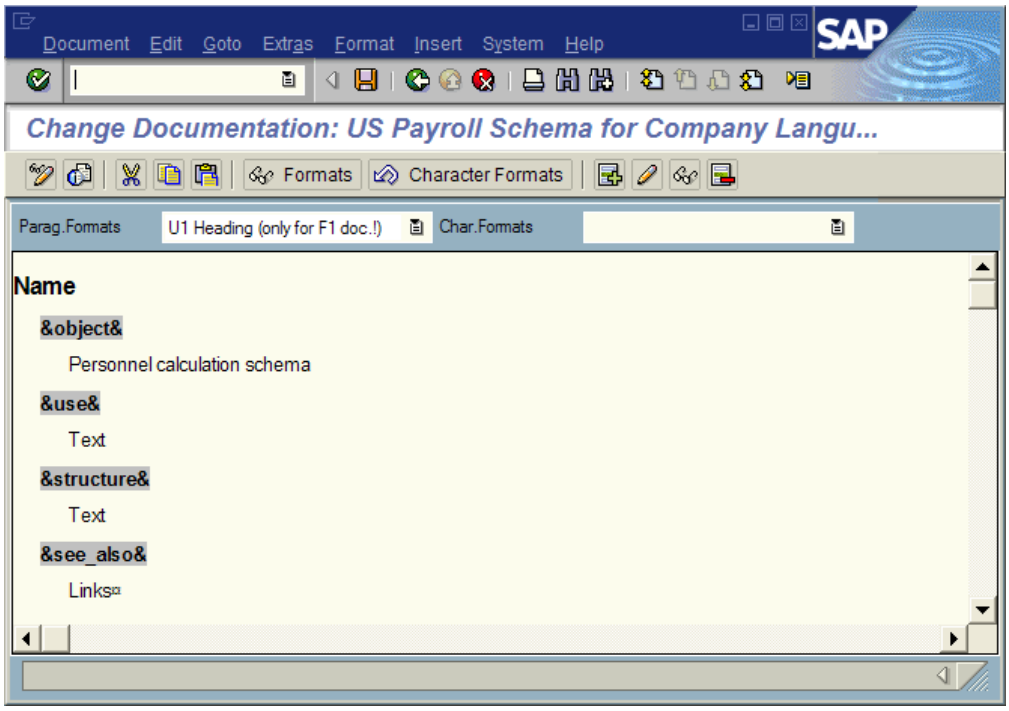

Figure 8 Editor from the Enterprise release

Typical documentation might look like what is shown in **Figure 9**. Another U2 heading was added for **Modification History**.

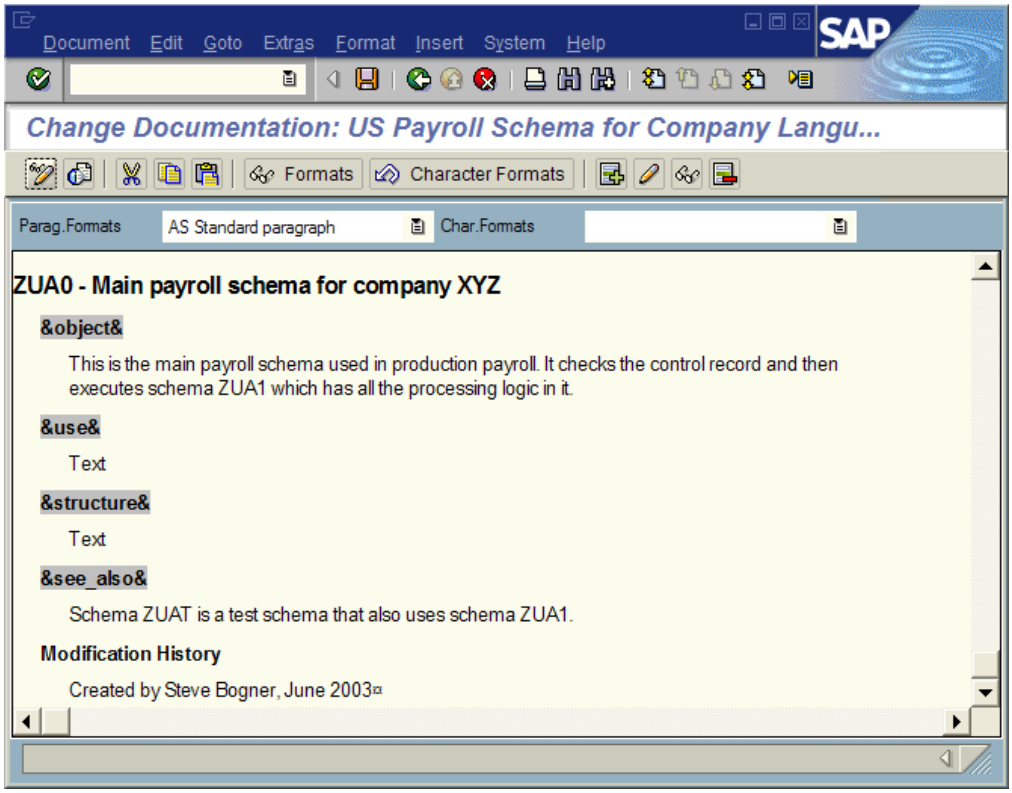

Figure 9 Modification History header added

When modifying wage types, constants, and other data via views in transaction `SM30` or the IMG, click on the blue **i** or **Info** button for documentation. The current documentation is brought up in display mode. Click on the edit button or press F5 to maintain it.

Transaction `PDSY` is a universal way to access all this documentation, regardless of where it comes from. Via `PDSY`, you can view and edit documentation, using the same editor, for many different objects. Early releases of R/3 used report RPDSYS00 to view and maintain documentation, but it has been replaced with transaction `PDSY`. While RPDSYS00 may continue to exist, you cannot be assured it is showing you the most recent documentation.

Document the 'Why'

Each rule, schema, wage type, and constant can be documented online. Click on the **Goto** menu and select **Documentation**. From there, you can enter documentation on why you are making the changes, reference the transport it is assigned to, and perhaps enter other change management information (i.e. issue number, ticket number and so on). The critical part is to define *why* you are making the change, not just saying *what* the rule does. Most people who have to come behind you to modify the rule can see what the rule is doing, but may not know why it was done a certain way or if there are other dependencies in the schema. This sort of documentation requires very little effort, and saves much time in the future.

Expanding the Schema and Rules

Report RPDASC00 can be used to pull all the **COPY** schemas and rules into one list. This is very useful when you are looking for certain things in the schema – for example, where is wage type 0COL used? Running the report as shown in **Figure 10** lists all schemas, sub-schemas, rules, and sub-rules for the main schema ZUAT.

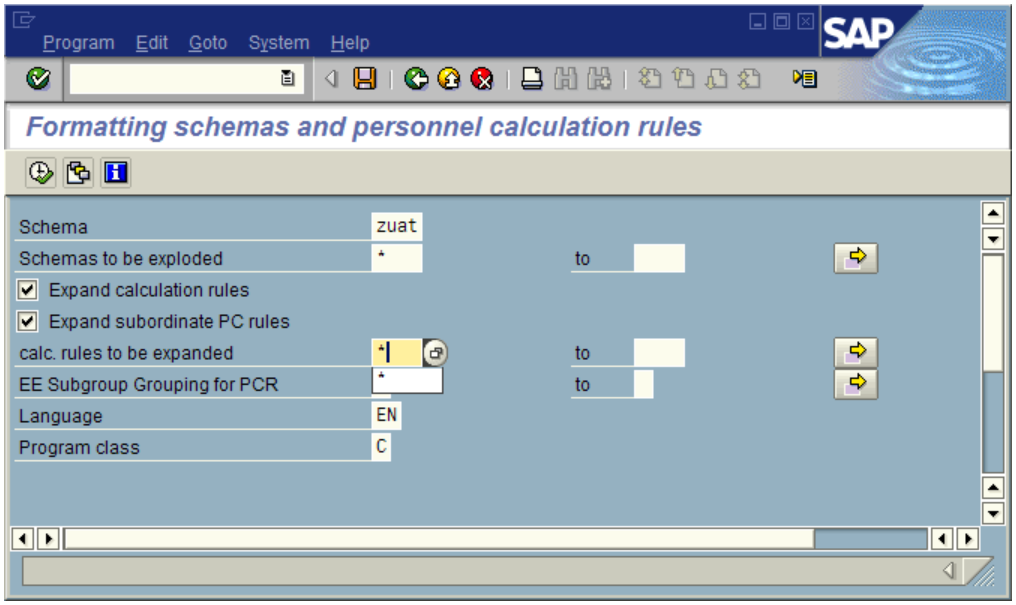

Figure 10 Report for main schema ZUAT

Running the Payroll Driver

The payroll driver can be run a number of different ways. From the payroll menu for a specific country, select the **Payroll>Start payroll** option. Or use transaction `PC00_Mxx_CALC`, where `xx` is the country identifier (or MOLGA). The payroll driver can also be executed from transactions `SE38` and `SA38`. Any way you go, this selection screen comes up (using U.S. payroll and the test employee as an example). (See **Figure 11**.)

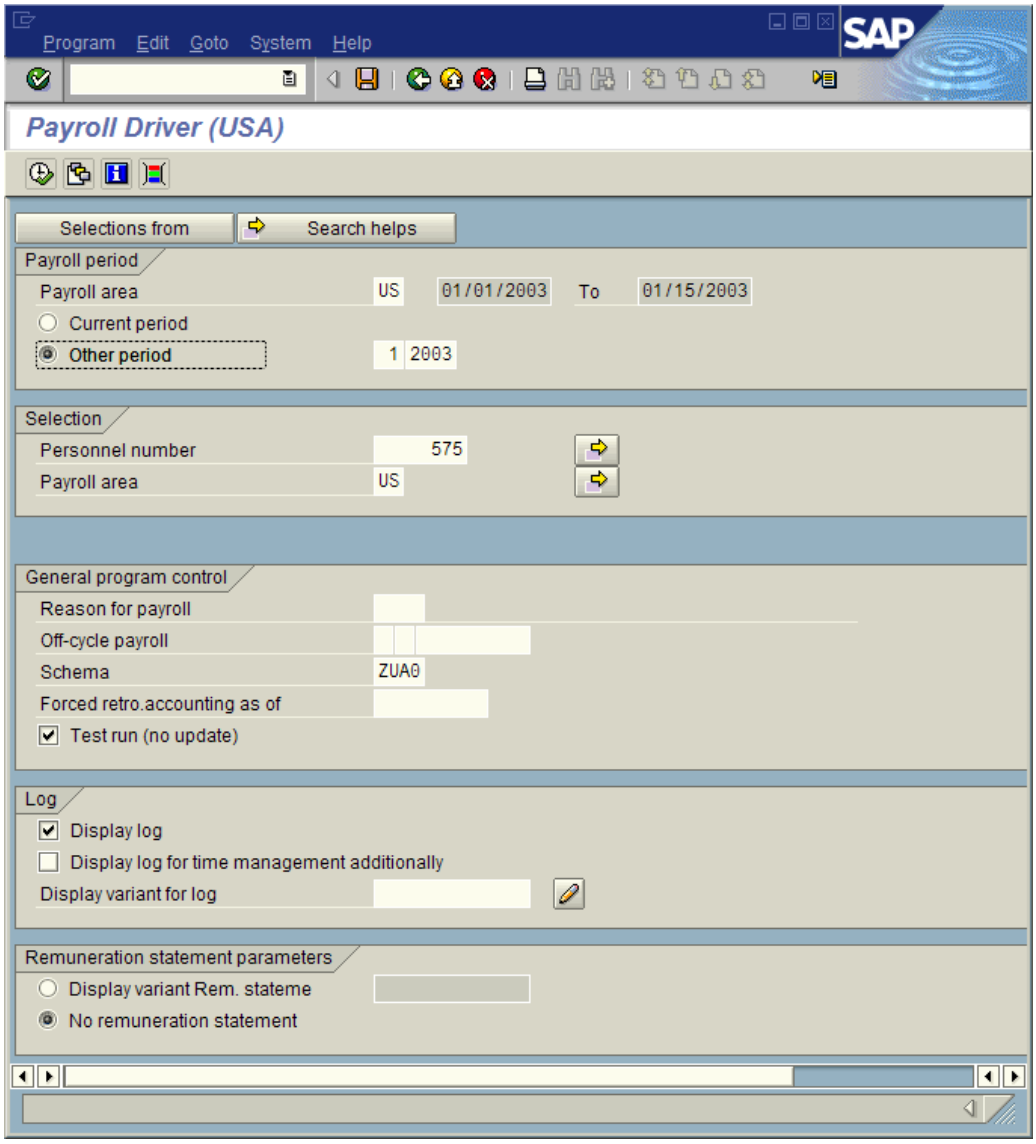

Figure 11 Payroll driver selection screen for the United States

If you are using the schema ZUA0 for production payroll, the payroll period is always left at **Current period** unless you are only running simulations, which tells the payroll driver to get the current pay period from the payroll control record. If you use the test schema ZUAT, you could select **Other period** and then enter whatever period and year you want to run, and save those results (only in a non-production system). The payroll area in this section is required and tells the payroll driver which payroll area to use for deriving the current period, the period begin and end dates, and other payroll area-related data.

The selection section specifies which groups of employees you include in the payroll run. The payroll area entered here determines which ones are included in the payroll run. These payroll areas need to have the same setup as the one entered at the top of the screen – the same period definition and pay dates, for example.

General program control is used to tell the payroll driver how to run the payroll. Off-cycle payroll uses various reason codes, and those codes can be entered here. If running an off-cycle payroll, you specify the type of payroll and the date of that payroll in the **Off-cycle payroll** line. The schema line specifies the main schema to use for the calculation. If you want to force a payroll run to go back to a certain date and recalculate everything since then, that date is entered in the **Force retro.accounting as of** field. And finally, there is an option for a test run – also known as a simulation run.

Each payroll function and operation has the ability to put additional diagnostic information in the payroll log. For testing payroll, it is good to have the log shown. In a production payroll, showing the log for a whole payroll area will severely affect system performance. The log requires a lot of temporary storage and significant processing resources. If the **Display log for time management additionally** is selected, then you will be able to jump from the payroll log into the day processing (DAYPR function) part of the gross payroll schema UT00. Otherwise, that part of the log is not generated. Display variants can be created to customize the way the payroll log is displayed, and if one has been defined you can select it in the **Display variant for log** line.

Finally, the remuneration statement parameters are used to define which, if any, HR Form is displayed at the end of the payroll calculation. If a form is specified here and the log is not displayed, then the payroll driver displays the results via that HR Form. If the log is displayed, it first displays the log but also gives you a button at the top to display the HR Form.

Reading the Payroll Driver Log

The payroll log varies from one release to another and from one country to another. It can also vary from one customer to another depending on how the schemas were customized. For these examples, the payroll log follows the standard schema U000 very closely, on the Enterprise release. (See **Figure 12**.)

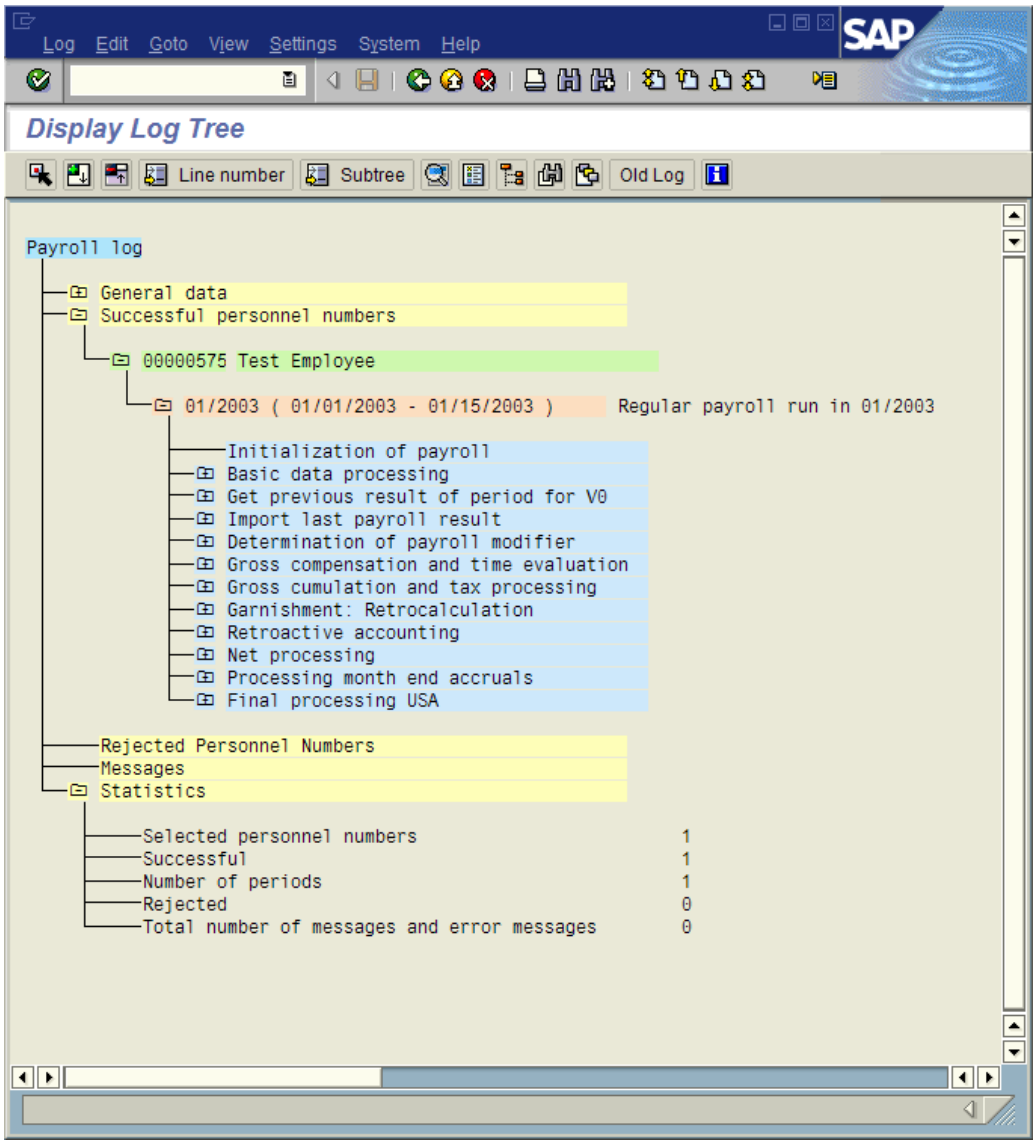

Figure 12 Payroll log in Enterprise

The log tells you that one employee was selected, and one employee was successfully calculated. The number of periods calculated may be more than the number of employees, since it counts the current and retrocalculated periods. Employees rejected from payroll would be counted in the **Rejected** line. The payroll driver rejects employees for two primary reasons – a lack of basic and consistent master data, and customization errors.

Each node can be opened to view deeper levels of processing, until you get down to a particular function and/or rule. To get to my example, you expand the nodes to the point shown in **Figure 13**.

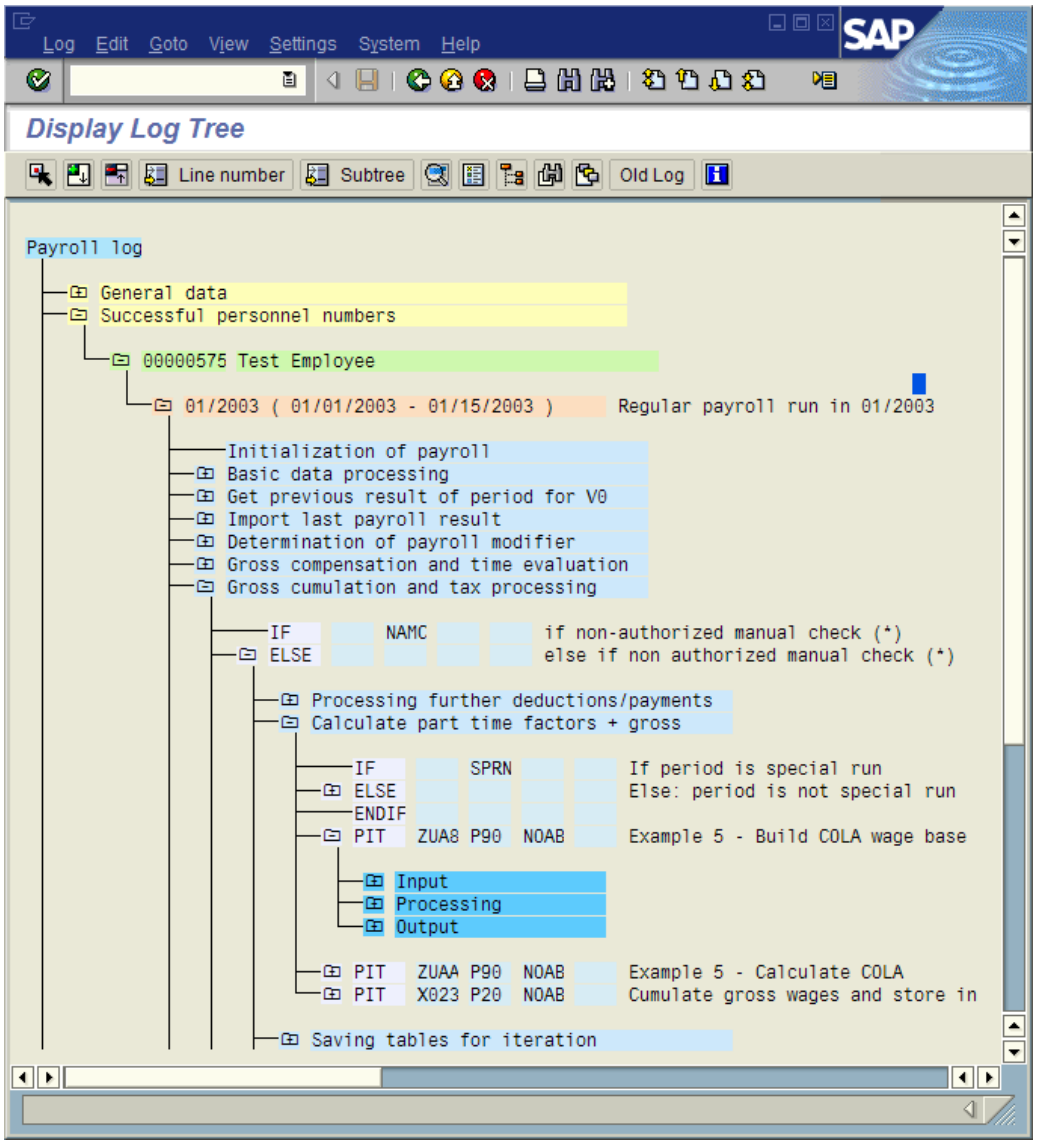

Figure 13 Expand the nodes

For the PIT function and rule ZUA8, you can see the **Input**, **Processing**, and **Output** sections. These sections vary by function and operation, but are vital to testing and troubleshooting. If you double-click the **PIT** function, the payroll driver displays the contents of the input, processing, and output sections together in one list. Or you can double-click any one section to see its contents alone. Likewise, you can further expand each section to see individual components – in this case, wage types.

In many of the detailed lists, you can see two icons – a green plus sign with a down arrow and a red minus sign with an up arrow (**Figure 14**). These icons control the level of detail that is displayed in the list. Click the green plus to see more detail and the red minus to see less.

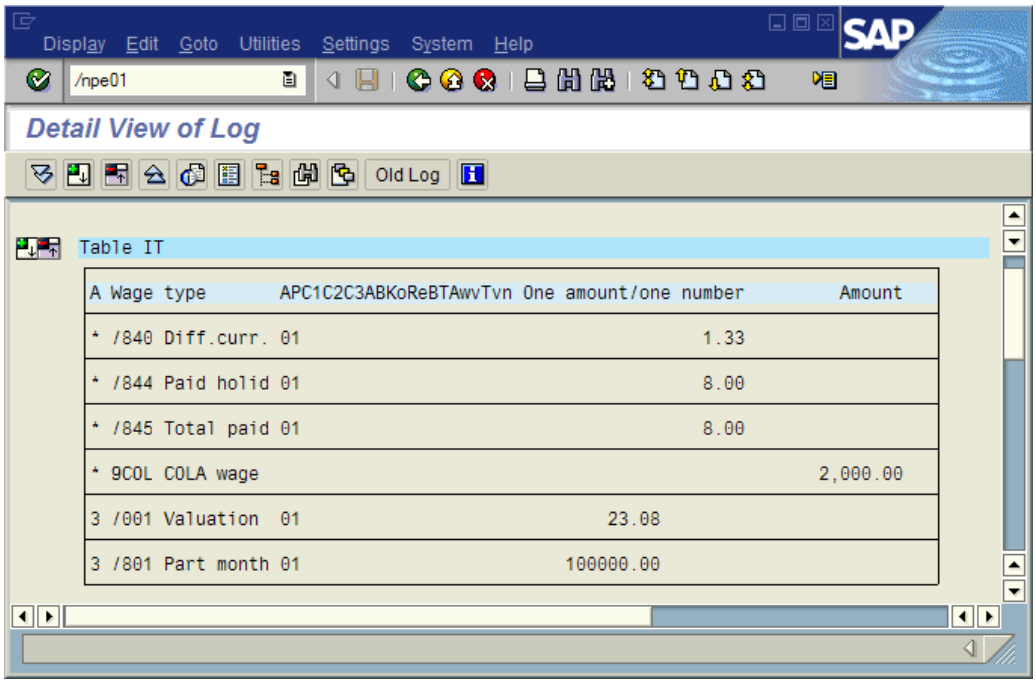

Figure 14 Green and red icons control level of detail

Understanding Which Wage Types Are Processed

A rule can also filter wage types, or it can process all wage types. It can filter by employee-subgroup grouping and by wage type. This is an attribute of the rule, and different logic can be executed for each set of filtered wage types. In **Figure 15**, you tell the editor you want to edit the section of the rule that will process wage type 0BAS for all employee subgroup groupings. To see all the various wage types a rule is set up for, click the overview button once you are editing the rule. The various sections can be managed by the **Edit>ES Grouping wage or time type** menu path inside the rule. If the employee subgroup grouping or wage type are set to asterisks, then they are valid for all values.

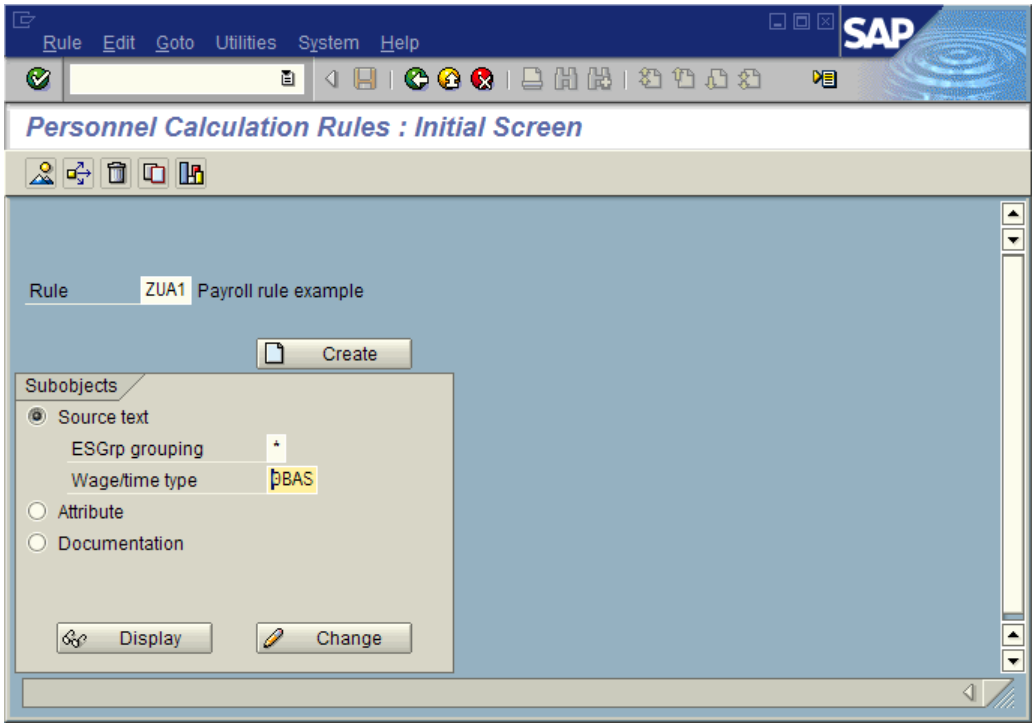

Figure 15 Select the section of the rule that will process wage type 0BAS

The rule's treatment of wage types is also affected by how it is called by the function. The rule gets its wage types from the function, so the function controls what the rule can process. Parameter 2 has the following values:

- GEN – Only access the '****' wage type section of the rule.
- Pxx – Like GEN but brings along the value of the processing class 'xx'.
- Exx – Same as Pxx except for evaluation classes.
- (blank) – Access the section of the rule that corresponds to the current wage type

Parameter 3 can have two values:

- NOAB – Only access the '*' employee subgroup grouping section of the rule.
- (blank) – Access the rule specifically for each employee subgroup grouping

The employee subgroup grouping for personnel calculations rules is maintained via view v_503_b.

Example 1 – Calculating a COLA Wage Type

The first example multiplies wage type 0BAS by 0.15 and puts the result into wage type 0COL. **Figure 16** shows the IT before rule ZUA1.

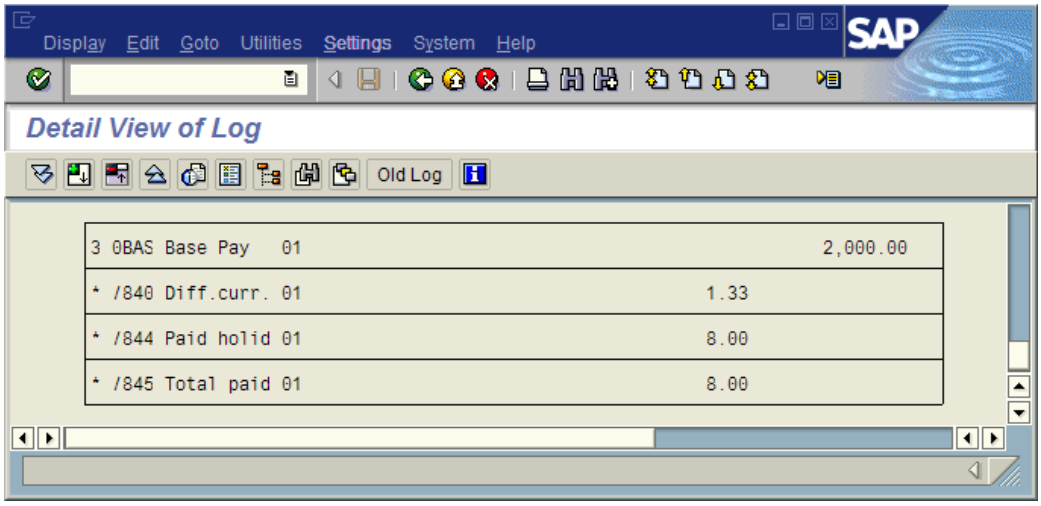

Figure 16 IT before rule ZUA1

Figure 17 shows the processing section for rule ZUA1, showing calculation done on wage type 0BAS.

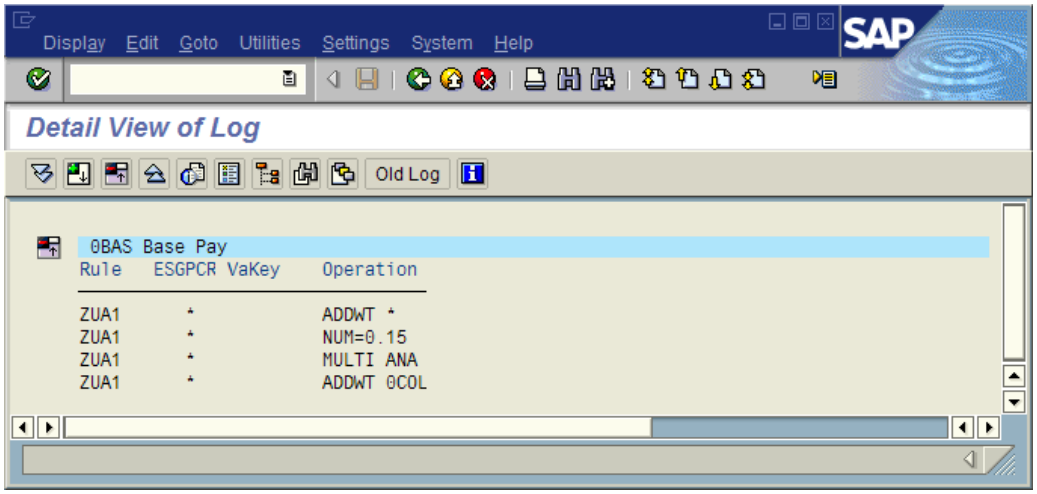

Figure 17 Processing section for rule ZUA1

See **Figure 18** for the IT after rule ZUA1 was processed, showing the new wage type 0COL with the same A and AP splits as wage type 0BAS, and 0.15 in the NUM field to reflect the percentage used in calculation.

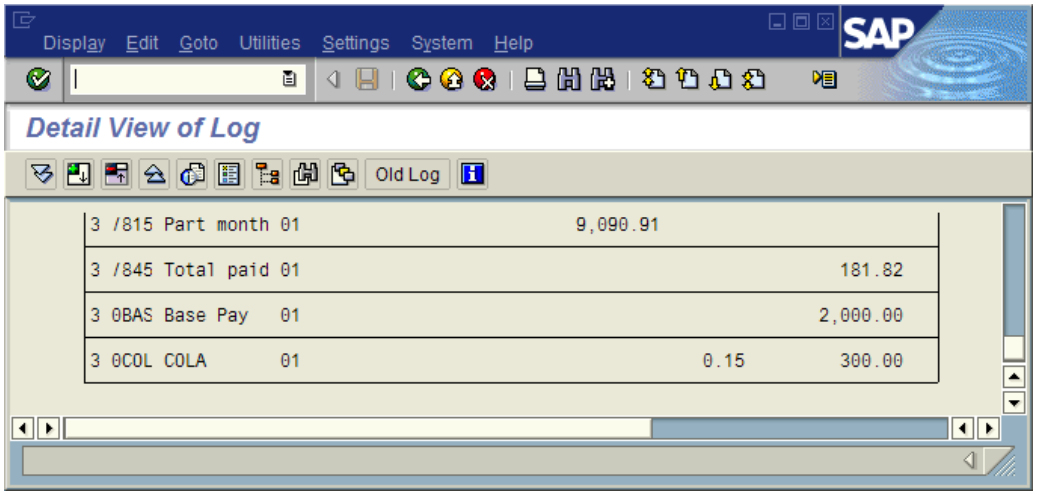

Figure 18 IT after processing of rule ZUA1

Example 2 - Restricting COLA to Specific Groups of Employees

Rule ZUA3 shows how to restrict COLA based on personnel area and employee subgroup. The calculation is the same – 15 percent of wage type 0BAS is put into wage type 0COL. The IT is the same as rule ZUA1, both before and after processing. The difference is in the processing section (**Figure 19**).

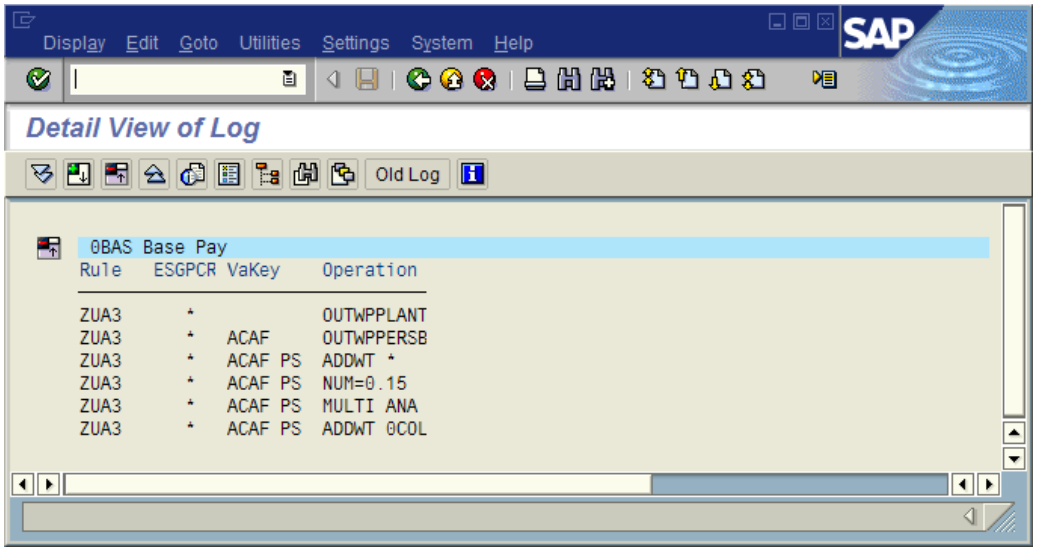

Figure 19 Processing section for rule ZUA3

If the employee is in personnel area ACAF, then you look at which employee subgroup they are in (OUTWP operation with the PERSB option). If they are in employee subgroup PS, then you calculate COLA.

For employees who are not in personnel area ACAF, or are in ACAF but not employee subgroup PS, you perform the ADDWT * operation. This copies wage type 0BAS from the header row to the IT. If you didn't do this, that wage type would be dropped from the payroll calculation at this point.

Example 3 - Bracketed Calculations & Flat-Amount Override

The third example shows how to use the AMT operation to make decisions based on a wage type's value. A nested decision structure is used calculate COLA with various rates. If wage type 0COL already has a value then you keep

that amount instead of doing the calculation. Since you ran out of room in the variable key in rule ZUA4, you break the logic into two rules, calling rule ZUA5 from ZUA4 when needed. The IT before processing is the same as in the previous examples, but the processing is significantly different (**Figure 20**).

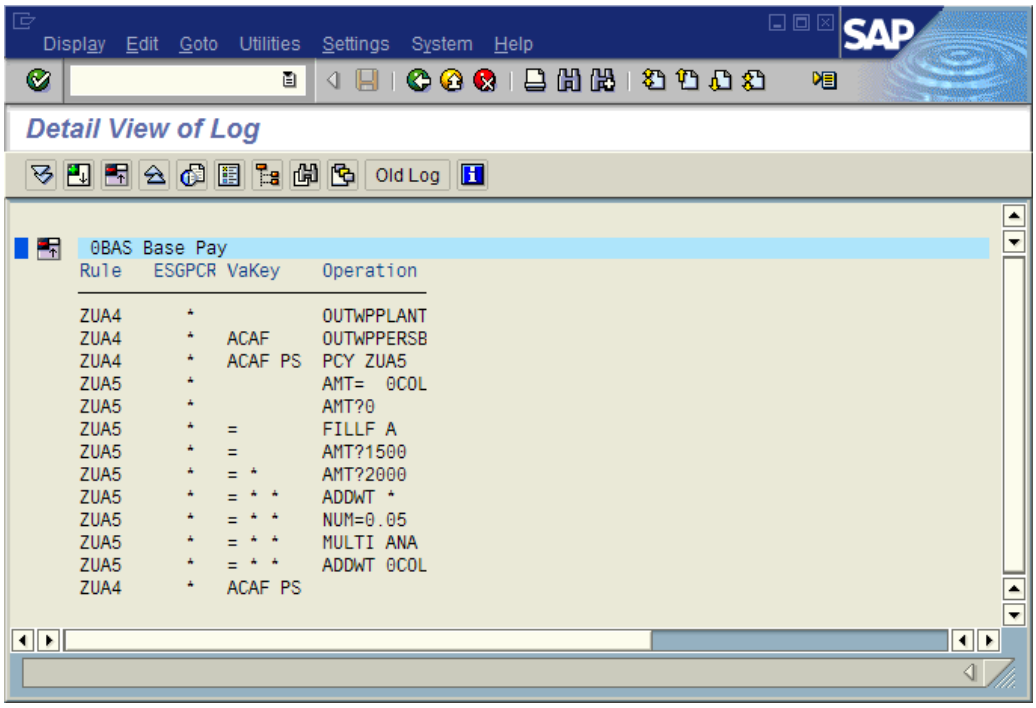

Figure 20 Processing using AMT operation

Notice how the PIT function starts with rule ZUA4, branches to ZUA5 and then returns. When wage type 0COL is entered via infotype 2010, the rule's behavior changes. The IT before processing now has 0COL in it (**Figure 21**).

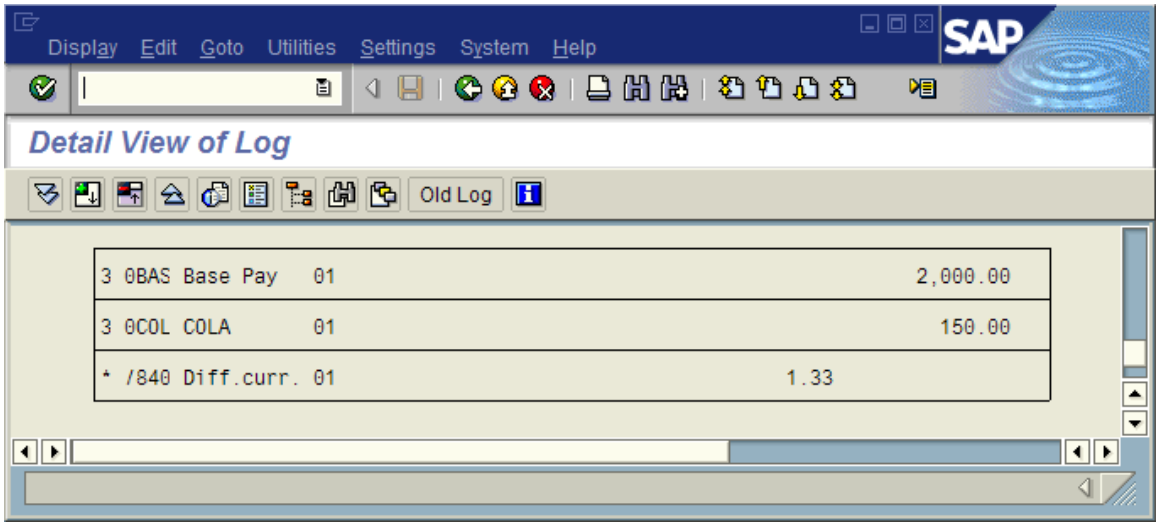

Figure 21 IT with 0COL

The processing section now sees 0COL and simply passes it on to the outgoing IT table (**Figure 22**).

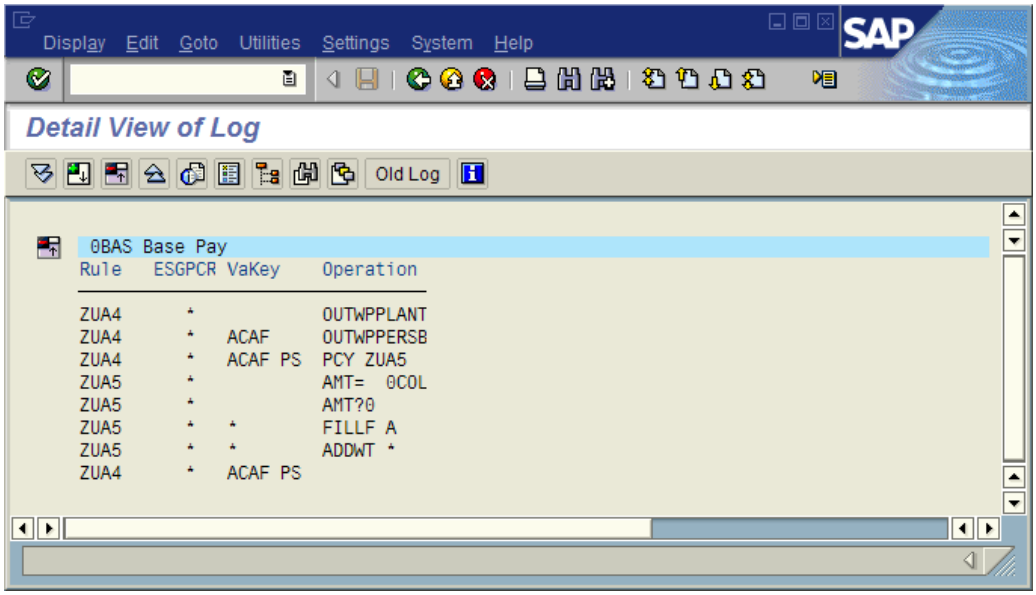

Figure 22 Outgoing IT table with 0COL

Notice that if wage type 0COL is entered for someone who is not in personnel area ACAF and employee subgroup PS, then it would be paid anyway. Rules ZUA4 and ZUA5 are processed only for wage type 0BAS. So if wage type 0COL existed in the IT before this rule, it would simply be passed on to the next function. This can be restricted by customizing views v_511_b, v_503_g and v_001p_k so that 0COL cannot be entered for this combination of personnel areas and employee subgroups.

Example 4

Hard-coding amounts and wage types into rule is a bad practice. Example 4 is the same calculation logic as Example 3, but the amounts are moved into constants, and use a processing class to specify the COLA wage base.

Table t511k, or view v_t511k, holds payroll constants. These constants are date-effective, and are read with the end-date of the pay period being calculated.

First you set up the new processing class. For customer-defined processing classes, start with 99 and work down. For this case, use processing class 90. In the IMG, go to the section for processing and evaluation classes (**Figure 23**).

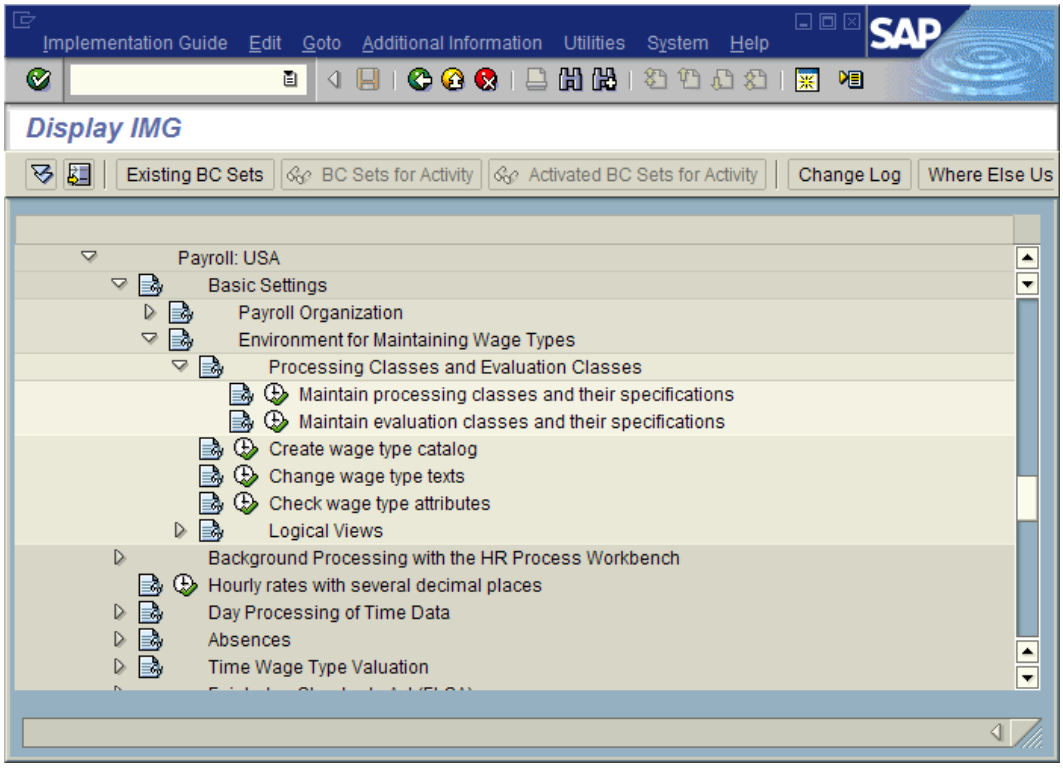

Figure 23 IMG section for processing and evaluation classes

Execute the **Maintain processing classes and their specifications** item, and create processing class **90**. Note that you can click on the blue "**i**" icon to document the processing class (**Figure 24**).

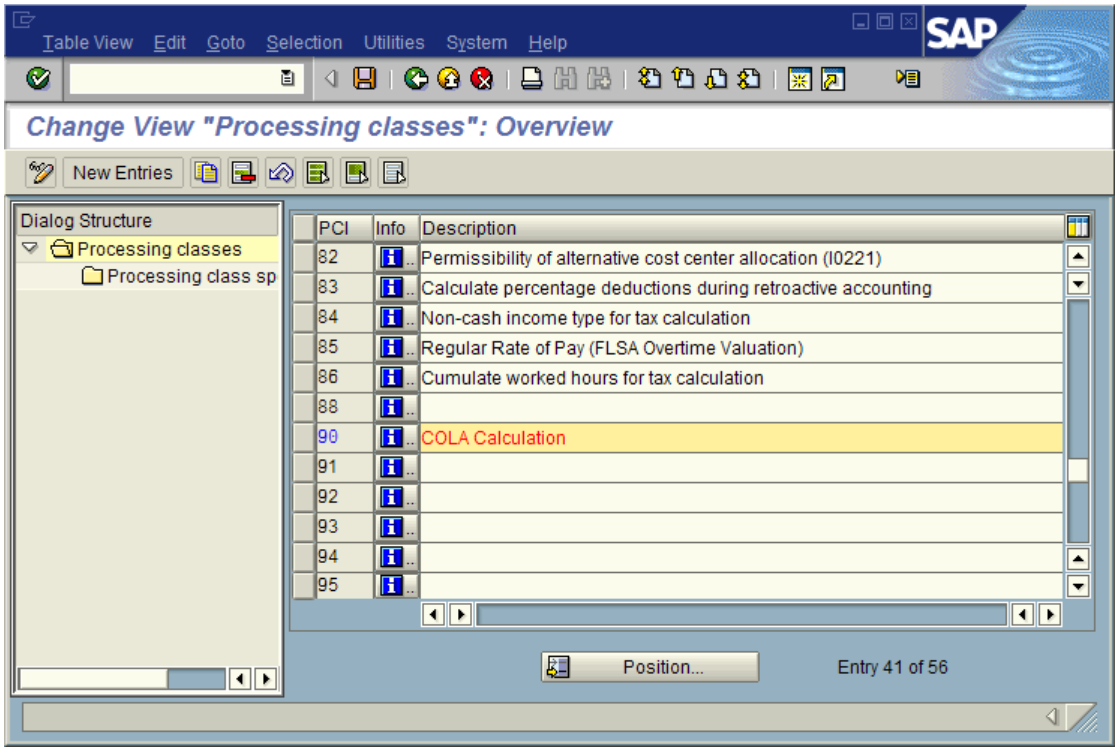

Figure 24 Click on the blue "**i**" icon to document the processing class

Select the processing class 90 row and then double-click on **Processing class specifications**. Here is where you define each value of processing class 90 (**Figure 25**). You can also document each value. Then go to view v_512w_o to set wage type 0BAS processing class 90 equal to 1.

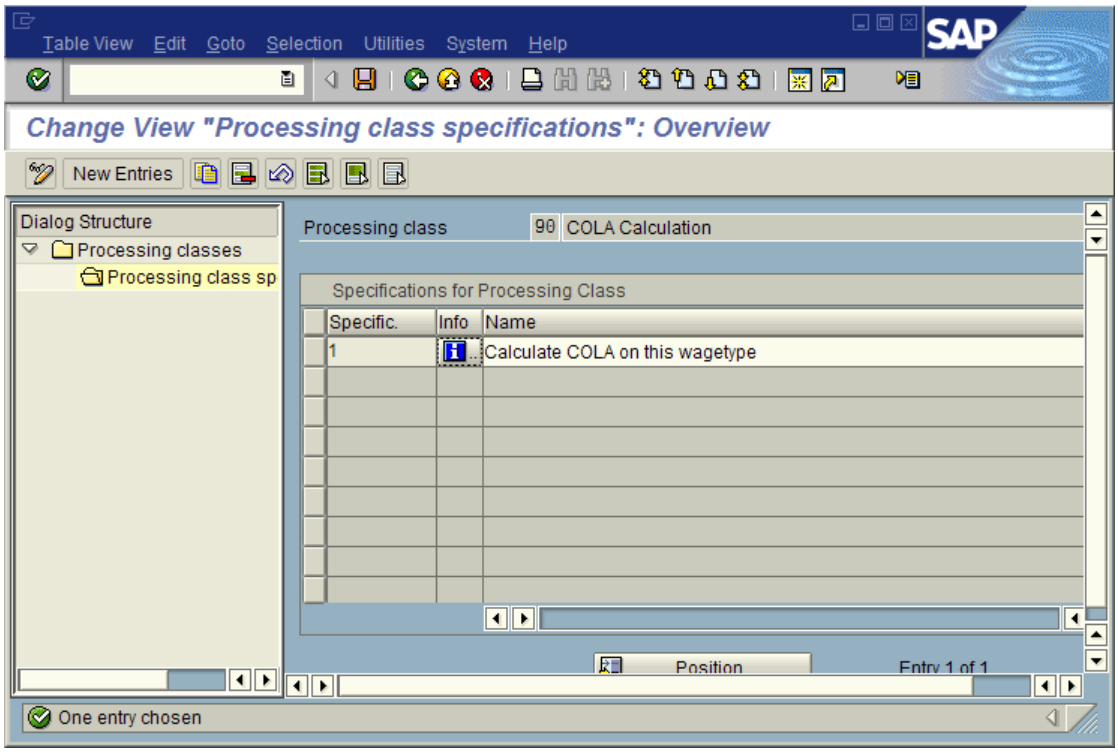

Figure 25 Set wage type 0BAS processing class **90** equal to **1**

Now you create the constants in t511k (using transaction sm30 and view v_t511k). Each constant can be documented (**Figure 26**).

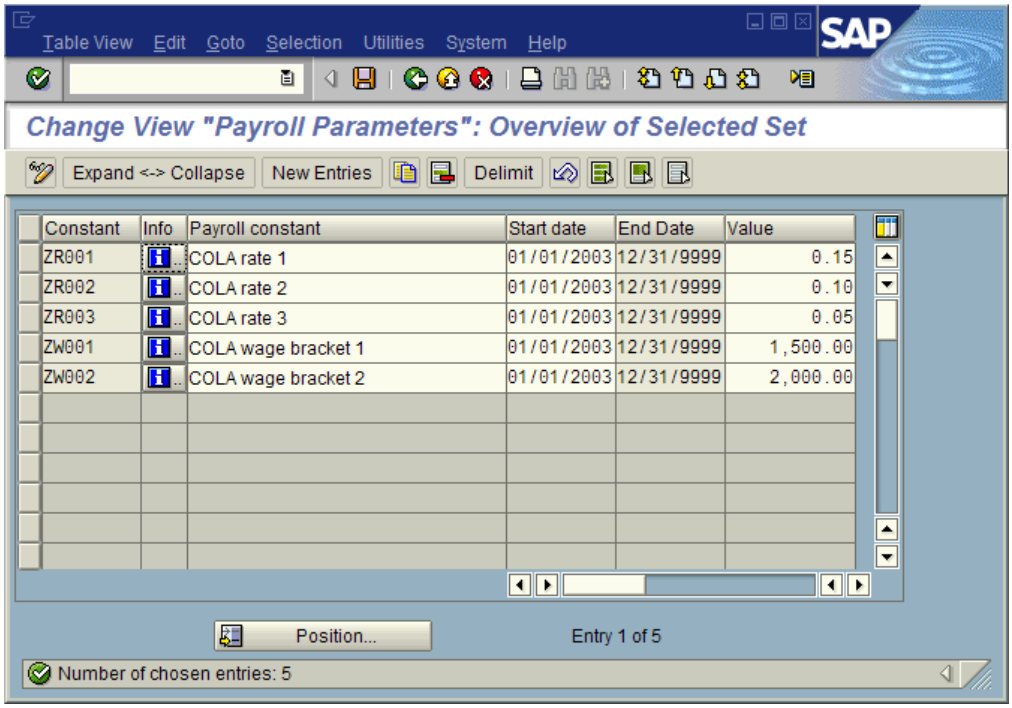

Figure 26 Creating constants

Rule ZUA3 also changes since you now have to execute a rule for a specific processing class. Use the **P90** value for parameter 1 (line **150**). (See **Figure 27**.)

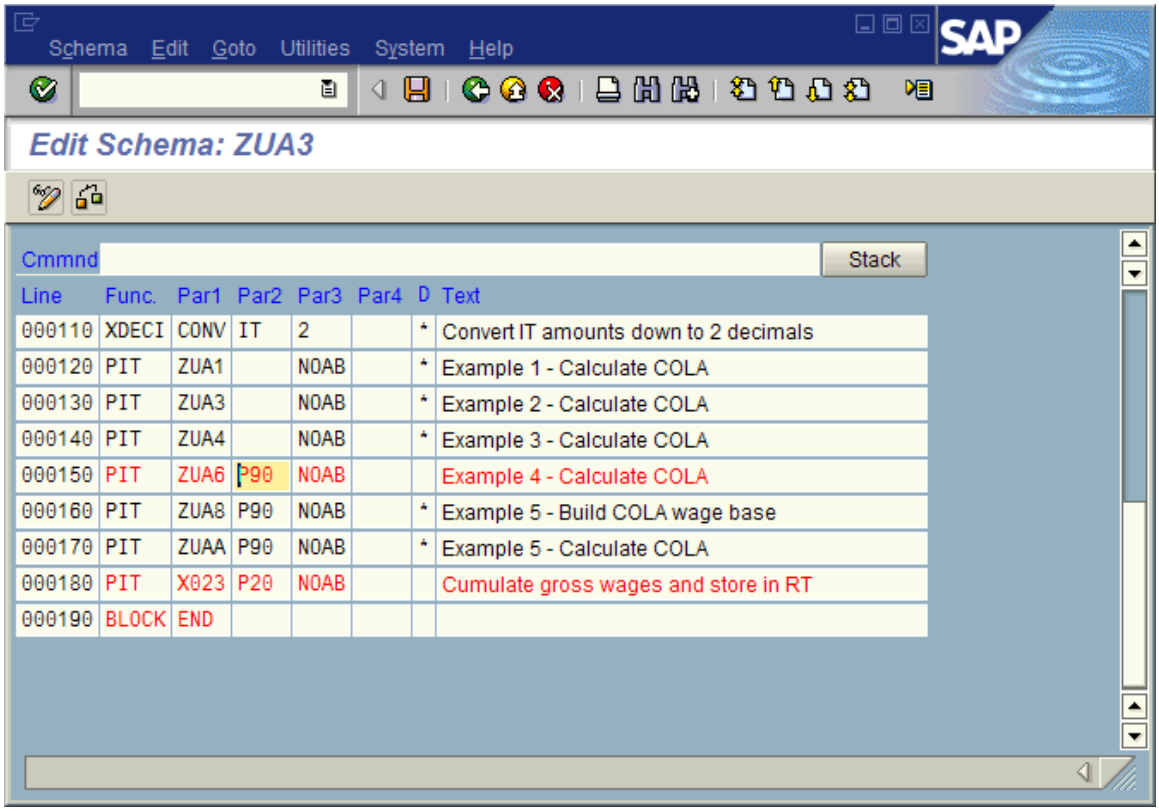

Figure 27 Use the **P90** value for parameter 1 (line **150**)

When payroll runs, the processing section for rules ZUA6 and ZUA7 look much the same as before, except you see constants and processing classes (**Figure 28**).

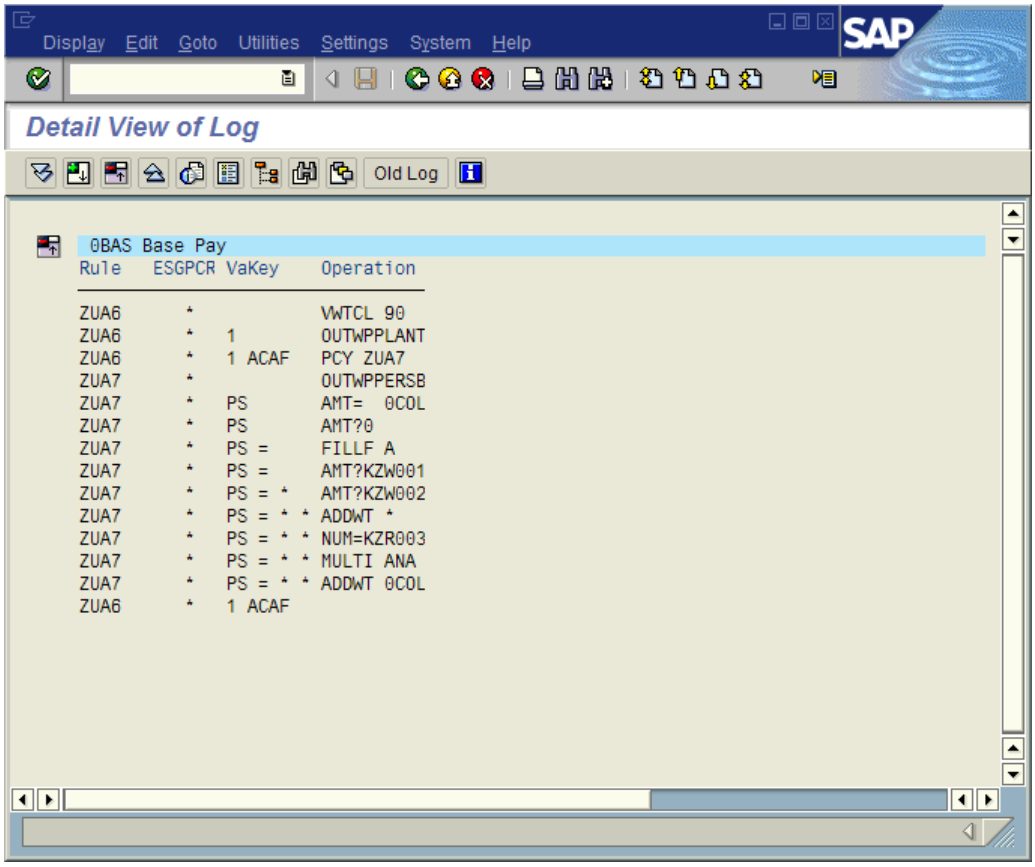

Figure 28 Processing section for rules ZUA6 and ZUA7 with constants and processing classes

Example 5

The last example involves the handling of wage type splits. Some of the splits have documentation, others don't. When viewing the RT table with the pc_payresult transaction (or the Display Results option in the Tools menu in Payroll), select a split value with your cursor and then press F1 for help. This gives you the full name for the split indicator. Some documentation is also available in transaction `PDSY`, using operations ELIMI and SPLIT. See **Table 2** for common split indicators.

Table 2 - Common Split Indicators		
Split indicator	Links to table in payroll results	Description
A	N/A	This is the ES grouping for the wage type, and it is matched up to an ES grouping section of a payroll rule for processing
AP	WPBP	Table WPBP holds basic data from infotypes 1, 7, and 8.

C1, C2, C3	Country-dependent	This varies by country, but it is typically used to link a wage type to tax authorities. In the U.S., split C1 links the wage type to a tax authority in table TAX
KoRe	C1	Cost assignment overrides can be entered for wage types and time data, and that override data is stored in table C1.
vTvn	V0	The variable assignment type (vT) and number (vn) link a wage type to table V0, and the purpose varies per country. In the U.S., this is the link to a benefit plan.

Table 2 Common split indicators

In the article in the August/September 2003 issue of *HR Expert*, I gave an example of an employee changing cost centers in the middle of a pay period. This causes wage type 0BAS to split – allocating a certain amount to each cost center via the AP split. The challenge, though, is to calculate COLA for each split value, based on the whole-period amount of 0BAS. To do this, you create another wage type to hold the whole-period amount. When you cumulate to that wage type – 9COL – you eliminate the splits, and that is what causes it to store the amount for the whole period. By using 9COL and processing class 90, you can now have multiple wage types cumulate to the COLA wagebase, which is more realistic.

The other thing you did was to move the 0COL wage type from infotype 2010 to infotype 14. This was so that you could allocate the 0COL wage type across the multiple splits. Infotypes differ on how they work with splits. (See **Table 3**.)

Table 3 – how Infotypes 2010, 15 and 14 handle splits	
Infotype	Split Handling
2010	Allocates the AP split based on whatever day it falls within the pay period. For example, the pay period is Jan 16 to Jan 31, and it splits on Jan 20. So Jan 16 to Jan 19 is AP split = 01, and Jan 20 to Jan 31 is split AP=02. If the infotype 2010 wage type is dated Jan 19, it will be assigned to AP split 01; if infotype 2010 is dated Jan 20 it is assigned to AP split 02.
15	When read into payroll the AP split is eliminated, so its AP split is blank. For accounting and reporting purposes, it will be assigned to the last organization assignment (table WPBP in particular), but for payroll operations it is not associated with either split.
14	If processing class 10 is set to something other than zero, then it will be split across the pay periods. Otherwise it is handled like infotype 15. So if there is an AP split in the period, infotype 14 will create multiple wage types in the IT – one for each AP split.

Table 3 How infotypes 2010, 15, and 14 work with splits

The behavior of each infotype can be customized in the payroll rules that are processed when functions P2010, P0015, and P0014 are executed. The logic listed in Table 3 is the standard logic from SAP (for the U.S.) – it could be customized in your system. So to get COLA in my example to work for splits, I moved 0COL to infotype 14 and set its processing class 10 to a 1. In this case, processing class 10 is used to tell payroll to allocate wage types across payroll splits.

Now, instead of cumulating the COLA wage base to wage type 9COL, you could cumulate it into a payroll variable. This is similar to a wage type, but it is not stored in the payroll result. Instead of doing `ADDWT 9COL`, you would do `ADDWT&9COL`. The "&" adds it to the 'variable' table VAR. To retrieve the value of the variable, you would do `AMT=& 9COL`. There are good and bad points with variables. The good point is that they are easily created dynamically in the payroll rules. The bad parts are that they do not have splits on them so you can not track values per split indicator, they are not stored in the payroll result so it makes auditing and explaining calculations more difficult, and if you do not clear them (`ZERO=& 9COL` for example), then the values from one gross-to-net loop flow into the next loop.

Period 2 had a split in it – a new event (infotype 0000) was processed on 1/20/03. Base pay was pro-rated, giving two wage types 0BAS to deal with (**Figure 29**).

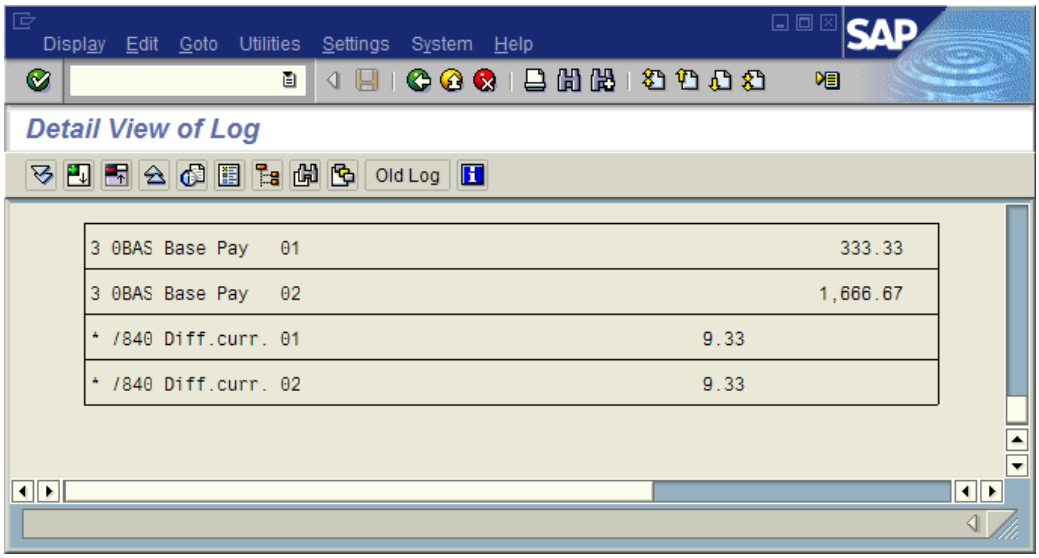

Figure 29 0BAS with two wage types

In the processing section of rule ZUA8, you can see that it was called once for each 0BAS (**Figure 30**).

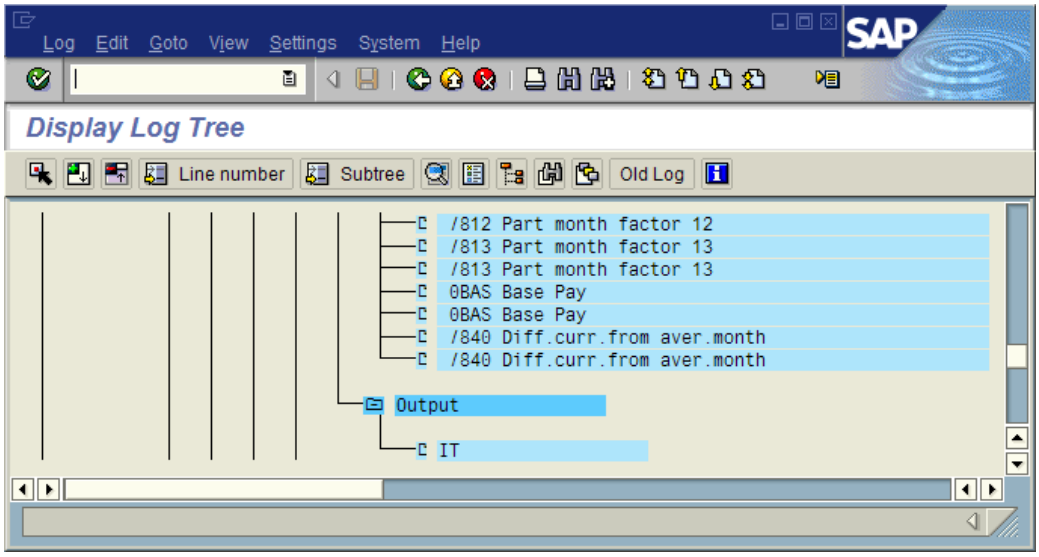

Figure 30 Processing section of rule ZUA8 called once for each 0BAS

For each of those executions, the same logic is performed. First, check to see if the current wage type has a processing class 90 of 1. Then, check to see if the employee is in a valid personnel area and employee subgroup. If it passes those two tests, then it first passes the current wage type along into the output – you don't want to lose the base pay wage types. Eliminate all the splits and add the wage type to the output as 9COL. Wage type 9COL then has no splits for the whole pay period, and holds the sum of all wage types whose processing class 90 equals 1. (See **Figure 31**.)

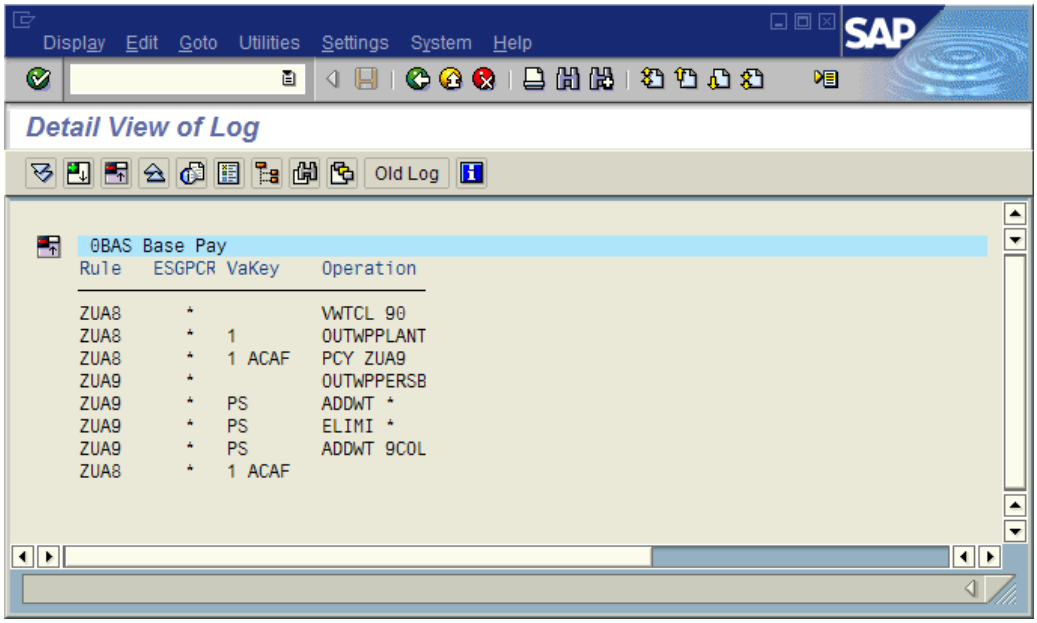

Figure 31 Wage Building wagetype 9COL

You can see that even though 0BAS had splits, wage type 9COL in the resulting IT does not have splits, and represents the sum of the two (**Figure 32**).

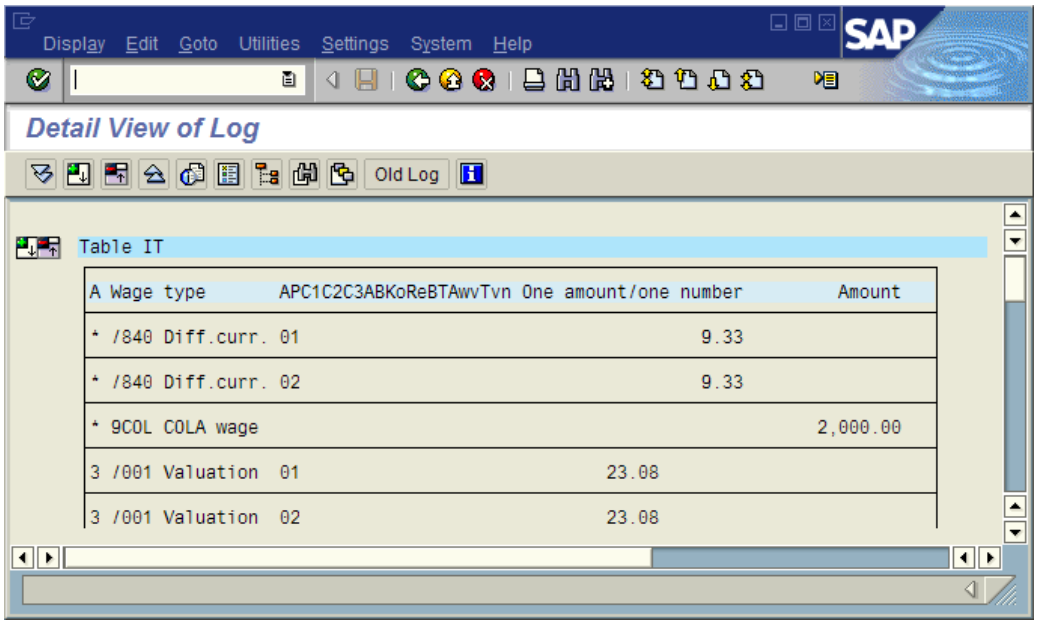

Figure 32 IT representing the sum of the splits

Now that you have the whole-period amount for the COLA wage base, you can process COLA for each of the wage base components. Rule ZUAA is also processed for all wage types that have processing class 90 equal to 1. In this case, that is the two wage types 0BAS. Each one is processed as shown in **Figure 33**.

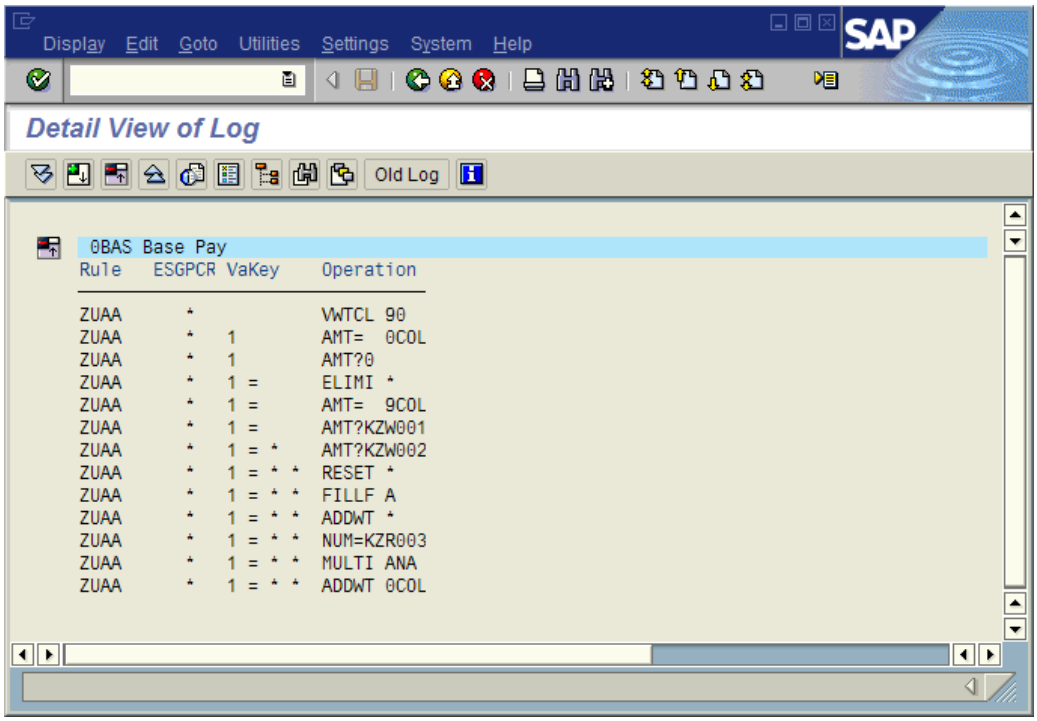

Figure 33 Processing of 0BAS wage types

First, set the amount equal to 0COL to see if there is an override COLA amount. The AMT operation looks for wage type 0COL in the IT where the split indicators in the IT are the same as in the header of the IT. So wage type 0BAS is in the header with an AP split of 01. It looks for wage type 0COL with a split AP of 01 in the IT. This is why you read 0COL into the IT via infotype 14 and processing class 10 = 1.

If the amount is equal to zero, then you don't have a 0COL wage type. Now, eliminate the splits on 0BAS in the header row, and go look for 9COL in the IT. If you find 9COL, then test the amount to see what wage bracket you fit into. Then restore the original splits and reset the amount in the header back to the original amount – i.e., from the 9COL amount to the 0BAS amount. Now you are back to the original 0BAS in the header row. Multiply that amount by the COLA rate (constant ZR003) and create wage type 0COL with the result. Wage type 0COL in this case has all the same attributes – including the split values – as wage type 0BAS. So the COLA amounts are split and allocated in the same manner as the wage types making up the COLA wage base (**Figure 34**).

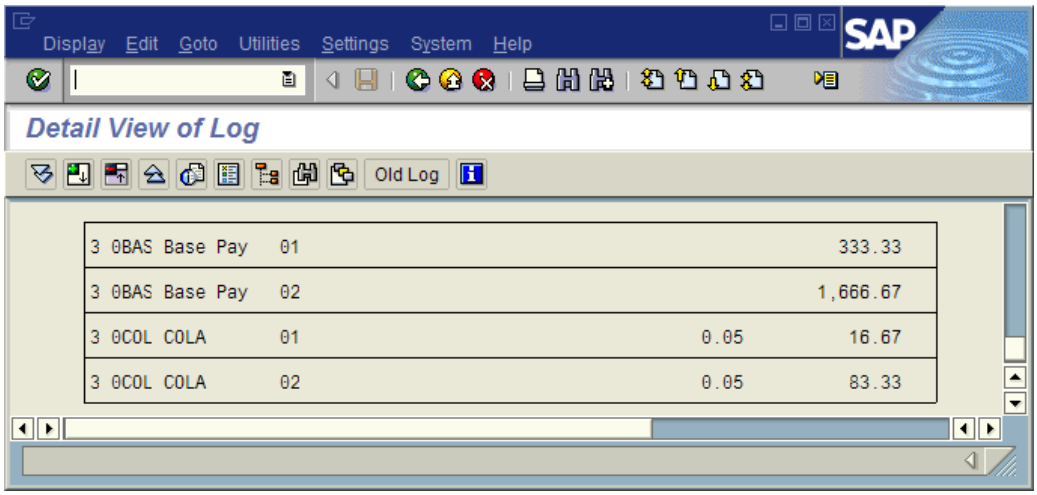

Figure 34 COLA amounts split and allocated in the same manner as the wage types making up the COLA wage base.

Steve Bogner is managing partner at Insight Consulting Partners and has been working with SAP HR since 1993. He has consulted various public, private, domestic, and global companies on their SAP HR/Payroll implementations; presented at the SAP user's group ASUG; and been featured on the SkyRadio Network program regarding SAP HR. You may reach Steve via email at sbogner@insightcp.com.