

# Oracle HR/Payroll – Top 10 Challenges

Ken Conway and Bill Stratton  
BOSS Corporation

## Introduction

This paper will describe our list of the Top 10 challenges we've observed from several different implementations of Oracle HR and Payroll. The subjects to be addressed are: Paid Time Off Accruals, Year to Date Balance Conversion, Time and Attendance Integration with Payroll, Overtime FLSA Compliance, Union Pay Scales/Pay Progression, Laser Printing of Payroll Checks, Costing on Earnings, Oracle Fast Formula, 401k Company Match, and Special Benefit Eligibility Dates.

## Time and Attendance Integration with Payroll

An external Time and Attendance (T&A) system can be integrated with Oracle Payroll via PayMIX (Payroll Mass Information Exchange).

To begin this process, an analysis must be done to map Oracle Payroll earnings to appropriate earning codes from the T&A system. While it would be ideal for the T&A system earning codes to be named exactly the same as the earning elements in Oracle Payroll, this is often unrealistic. For example, the T&A system may have limitations on the number of characters allowed in an earnings code name.

The characteristics of the application program interface (API) from the T&A system can influence this mapping analysis. Consider a T&A API where a single output record contains hours and time worked. In this example, consider a record containing 12 hours worked from 10pm to 10am. The 12 hours worked may need to be split within Oracle as 8 hours Time Entry Wages and 4 hours Overtime. Moreover, a nighttime shift differential record may be necessary in Oracle for all, or a portion, of those 12 hours.

Thus, one record from the T&A system could produce 3 records within Oracle PayMIX:

Oracle Earning Element	Hours
Time Entry Wages	8
Overtime	4
Shift Differential	8

Mapping can be affected by costing requirements. Within Oracle PayMIX, the costing on a single record cannot be split. Consider a variation from the example above where the 12 hours should have been split evenly across two cost centers. This might cause a need for 6 records within PayMIX.

Oracle Earning Element	Hours	Cost Center
Time Entry Wages	4	100
Time Entry Wages	4	200
Overtime	2	100
Overtime	2	200
Shift Differential	6	100
Shift Differential	6	200

The frequency of data transfer should be considered. Determine how often batches should automatically be transferred into PayMIX. Over the course of a payroll period, it is not necessary to transfer data from the T&A system every minute. A transfer once per day is too infrequent. Consider an hourly transfer. This should be frequent enough, even during the final day of payroll close. Assure that you have a manual ability to transfer batches into PayMIX. Determine when other network traffic (unrelated to payroll) occurs. Carefully coordinate the scheduled time for payroll transfers. If there is typically a peak of network traffic at the top of each hour, then establish your automatic transfer at 15 minutes after each hour.

In order to organize PayMIX batches, determine who will review batches and establish a batch naming convention for the Reference Name of a batch. Consider grouping batches weekly and by location in order to facilitate sorting for user access and for reporting. Here's an example naming convention: YMMDDLRRRRRR where:

Item	Description
Y	Year, expressed as only one digit (7 for 1997, 8 for 1998, etc.) to allow more text to be visible in the Reference field on the PayMIX Batch Information window.
MM	Month
DD	Day
LL	Location
RR	Reference number identifying the original source of the data from the legacy system.

Other issues for grouping of batches should be considered. PayMIX is designed such that separate batches exist for flat amount (Earnings) batches versus rate times time (Time) batches. You may want to create separate batches that are dedicated to exceptions (such as reduction in pay to recurring elements using Special Inputs elements).

The integration into PayMIX will involve insertion into these primary tables:

- PAY\_PDT\_BATCH\_HEADERS
- PAY\_PDT\_BATCH\_LINES
- PAY\_PDT\_BATCH\_CHECKS

If dynamic costing information is included in this integration, the integration will access these tables:

- FND\_ID\_FLEX\_SEGMENTS
- FND\_FLEX\_VALUES
- PAY\_COST\_ALLOCATIONS\_F
- PAY\_COST\_ALLOCATION\_KEYFLEX

First, identify which cost segments, such as location and cost center; can dynamically change from one payment period to the next. There will be other segments, such as a cost account, which will probably not change. In this example, assume that cost center and location are segments 1 and 2 respectively. Also assume that these are the only segments that can dynamically change. For each time-entry record, verify that the values actually exist in the value set for segments 1 and 2 by accessing the tables:

- FND\_ID\_FLEX\_SEGMENTS
- FND\_FLEX\_VALUES

Second, now you know that each of the segment values is a valid value but now need to determine if the code combination is valid. Determine if the code combination exists for the business group within the following tables:

- PAY\_COST\_ALLOCATIONS\_F
- PAY\_COST\_ALLOCATION\_KEYFLEX.

If the code combination exists, then retrieve the cost\_allocation\_keyflex\_id. If it does not exist, then insert into these tables using the PYCSKFLI package and retrieve the newly created cost\_allocation\_keyflex\_id.

One final note on the integration from T&A to PayMIX. Keep in mind that Oracle Payroll is not Fair Labor Standards Act (FLSA) compliant. The integration can be written to accommodate FLSA rules such that each overtime record within PayMIX can have an adjusted overtime rate.

## Overtime FLSA Compliance

Oracle Payroll is not Fair Labor Standards Act (FLSA) compliant in the calculation of overtime. The seeded Oracle Payroll earning for Overtime does not properly calculate the adjusted overtime base rate of pay when the payroll frequency is greater than one week. The seeded Overtime earning calculates an average rate based on the entire payroll period not on a separate week by week basis.

Example: Biweekly payroll period from 01-DEC – 14-DEC for employee making \$10/hour. Note in this example that the earning “Sales Commission” was created with Overtime Base checked on the Earnings Form.

Element Entries				
Date	Earning Element	Hrs	Multiple	Amt
Week 1				
07-DEC	Time Entry Wages	40	1	\$400
07-DEC	Sales Commission			\$200
07-DEC	Overtime	10	1.5	
Week 2				

14-DEC	Time Entry Wages	40	1	\$400
14-DEC	Overtime	1	1.5	

FLSA expects this overtime calculation:  
 Week 1 (\$400 + \$200) / 40 hours = \$15/hour  
 \$15/hour \* 10 hours OT \* 1.5 multiple = \$225

Week 2 (\$400) / 40 hours = \$10/hour  
 \$10/hour \* 1 hour OT \* 1.5 multiple = \$15

**FLSA Overtime = \$225 + \$15 = \$240**

Oracle Payroll calculates:  
 (\$400 + \$200 + \$400) / (40 hours + 40 hours) =  
 \$1000 / 80 hours = \$12.50/hour

\$12.50/hour \* 11 hrs OT \* 1.5 multiple =  
**Oracle Payroll Overtime = \$206.25**

Thus: Employee is underpaid for the current payroll period by \$33.75.

There are two different approaches to solving FLSA overtime compliance calculations. Both involve the use of Oracle PayMIX. Both involve the entry of an adjusted rate of pay into PayMIX for each overtime record. Approach #1 involves calculating the adjusted overtime rate before a record is inserted into PayMIX. Approach #2 involves calculating the adjusted overtime rate after records already exist in PayMIX. Approach #1 can only be used if you have an external Time & Attendance system. The integration program from Time & Attendance to PayMIX can include all of the overtime rules. If you are using Oracle Time Management, you will need to use Approach #2. If Approach #2 is used, it is vitally important the PayMIX Transfer process, which transfers PayMIX batches to the Element Entries window, is not run before the overtime adjustment process is run.

Questions you must ask in implementing an adjusted overtime calculation. Which earnings (Commission, Bonus, etc.) can influence an adjusted overtime rate of pay? Where do these adjustment earnings reside?

Does the integration program have access to all earnings for the current pay period at the time that the integration is being run?

Some earnings may be setup in Oracle as recurring earnings. If so, the overtime calculation program must read an employee's Element Entries to determine if the specific recurring earning exists. Other earnings, such as commissions, may have been previously fed through a different process into PayMIX. Thus, the overtime calculation program must read PayMIX to determine if the specific non-recurring earning exists. The integration gets complicated and is very rules based, but the FLSA deficiency can be solved.

If the Oracle Payroll product is ever upgraded to become FLSA compliant using the seeded Overtime element, take these steps to be prepared. All other earnings elements (Commission, Bonus, etc.) must be created carefully. On the Earnings window, select the appropriate check boxes for FLSA Hours and Overtime Base. Remember that if you fail to check these box appropriately, you cannot change the earnings' definitions later. In the meantime, your custom process to calculate an adjusted overtime-base rate of pay can use this information in establishing its rules for calculation.

## Union Pay Scales/Pay Progression

The use of Pay Scales is a common need when addressing union contracts. It is common to have a HR policy where various jobs have pre-negotiated rates of pay based on duration of employee service. For example, after 6 months, the employee is paid \$X, after 12 months, \$Y, etc.

There are two main obstacles to overcome in solving this problem via Oracle HR. First, Pay Scales are not tied to Salary Administration. Second, Oracle's implementation of automatic progression (how an employee's rate of pay is advanced from one step of the pay scale to the next) is not flexible. Automatic progression is not based on each individual employee's hire date nor is it based on the amount of time an employee has occupied a specific step on a pay scale.

The following has not been implemented but is a proposed solution for these two issues. Oracle Pay Scales will be used for data storage only. The automatic processes within Oracle HR will

not be used to allow employee advancement to a new progression step. Custom processes will be written to handle these situations. These custom processes will, in turn, use the supported Salary Administration API to generate new salary proposals.

We recommend a two-part solution. One solution will solve the issue when an employee becomes eligible to advance to the next step on a pay scale. The second solution will address the issue when there is an annual pay increase for all steps on a given pay scale.

The first solution addresses the advancement of an employee from one step to the next in a pay scale. The custom process will perform the following activities.

- a. Read employee's assignment and underlying grade step point placement records. Determine employee's current pay scale, step, and current date at that step. Note: A descriptive flexfield segment on the assignment form should be created to contain a "Grade Step Override Date". This override date exists to handle issues such as unpaid leave. If a date value exists in this field, then the program should ignore the effective date in the Grade Step form and use this override date (regardless of whether the override date is before or after the Grade Step's effective date).
- b. Based on the employee's pay scale, read the table corresponding to the Grade Step Placement window to determine if the employee is already at the ceiling. If so, stop
- c. Read the table corresponding to the Pay Scale window to determine the frequency that an employee should progress to the next step.
- d. Based on the date from step a and the frequency from step c, if the employee is not eligible to advance to the next step, stop. If the employee is eligible to advance to the next step, then write a SQL statement to update the table (no API currently available) to place the employee at the next available step for the Progression Point Placement window. *Warning, There is significant risk in writing directly to database tables. We do not recommend this as a general practice. Since Oracle does not offer an API for this item, the only other alternative*

*involves use of a report followed by a manual process. It is recommended that extensive testing occur on this step.*

- e. Clear the segment "Grade Step Override Date" from the employee assignment descriptive flexfield.
- f. Generate a salary proposal using the Salary Admin API.
- g. Payroll Manager must approve the proposed salary changes through the Oracle HR application.

The second solution addresses the issue when there is a pay increase (usually annual) for all steps on a given pay scale. The custom process will perform the following activities.

- a. Determine whether a new date tracked record exists in the progression point values table.
- b. If a new entry is identified, search the employee assignment Grade Step Placement database table to find all employees who are at that specific progression point step on that specific pay scale.
- c. Write SQL statement to update the table (no API) to reflect that the employee's current progression point step has a new rate of pay. This step might not be required. The fact that they are already assigned to the step should automatically point them to the correct new rate of pay.
- d. Repeat steps f-g above.

On another issue, consider this suggestion. When naming the points to a pay scale, be descriptive. Instead of referring to pay scale points generically as 1, 2, 3, 4, etc., use names such as 6 months, 12 months, 18 months, Temp Supervisor, Temp Maintenance, Temp Foreman, Temp Mechanic, etc.

One last thing to help make things more confusing. If you begin looking at the database tables through SQL, you'll discover the terminology of "spines". Progression points on the application windows are called spinal points in the database. Pay scales on the application windows are called Parent Spines.

## Costing of Earnings

Oracle Payroll is extremely flexible when it comes to costing of earnings. Cost information can be obtained at several levels within the Cost Allocation Key Flexfield: Payroll, Organization, Element Link, Assignment, Element Entry, and PayMIX. The valid cost levels are established when defining Flexfield Qualifiers for each segment of the Cost Allocation Key Flexfield.

How do you know which costing levels are appropriate for your implementation? Anytime you are in doubt, open up the segment (via Flexfield Qualifiers) at a costing level. Just because the segment is opened doesn't imply that you ever have to fill in data at that level. Usually, General Ledger (GL) Accounts are only opened at the Element Link level and GL Cost Centers are opened at all levels.

When preparing for the implementation of HR Organizations, it is recommended that you keep the GL requirements in mind. Costing defaults for earnings (and deductions) can be established based on HR Organizations. This aids in the future maintenance of employee information. The HR department does not have to concern itself with the GL cost segment values. The HR department merely needs to assign an employee to the correct organization. These costing defaults are particularly beneficial when an employee's earnings are costed the same way every pay period. Keep in mind that these costing defaults for an organization are flexible and can be overwritten during payroll processing. If Oracle Payroll retrieves costing information at a lower level, such as Element Entries, it will override the defaults from the employee's organization.

Oracle Payroll is very flexible in dynamically changing a cost segment value from one payroll period to the next. Time card records can be entered into PayMIX using different cost segment values for each record. Consider a maintenance employee who may work four hours at a cost center, two hours at another cost center, and still two more hours at a third cost center. The next day, this same employee may work at other cost centers. For more details about the integration of cost data into PayMIX, please see the section of this paper titled "Time and Attendance Integration with Payroll".

What about cost defaults for groups of employees where the organizations of the employees do not apply? This is where the People Group Key Flexfield becomes particularly beneficial. Consider a situation where you want to separately obtain cost account information for employee Regular Salary using a distinction between executive and administration employees. In this example, executive and administration employees may be scattered throughout numerous organizations. You could set up a People Group Key Flexfield segment to distinguish these two groups and then establish two Element Links for Regular Salary based on the People Group segment value. *Warning, At the time of the writing of this paper, a bug exists when salary changes are made in the middle of a payroll period. If Regular Salary has separate links based on People Group, you will receive a payroll error that salary is not valid for the entire length of the period.*

Here are a few thoughts that influence your transfer of costing information for balancing/offsetting entries to the GL. We have observed that balancing segments behave as if they are fixed costed even if the earnings element is defined as Costed. Consider setting up the Element Link with a fully qualified balancing segment. This will work if you are balancing to only one account for cash. If you have multiple cash accounts, you will need another solution. If you are too far into your implementation and cannot change your Element Links, be prepared for another solution. You will need a customized process that is invoked after you have run the Transfer to GL process. You will need to create a pre-process program that alters the data before the GL performs its import.

## **Benefit Eligibility Dates**

Most organizations administer benefits using eligibility criteria based on employee length of service using a special date that is other than the employee's most recent hire date. The Oracle Payroll Element Link window allows elements to be linked using employee qualifying conditions such as employee age and length of service following hire date. Here are a few examples of special dates:

- Adjusted hire date for rehiring of previously employed employees to give credit for previous periods of service
- Date that employee completed training or certification
- Adjusted service date to account for company buyouts
- Plant, unit or other seniority dates
- Job seniority dates

Oracle HR allows for special user defined dates to be stored using flexfields. The most common placement of these dates is using either the Person Descriptive Flexfield or a Special Information Type. What are the advantages of using one versus the other? Your decision should be based primarily on how you want to retrieve this data.

The advantage of a Special Information Type is that there are an unlimited number of segments at your disposal. If you establish a Special Information Type Key Flexfield for special dates, there are 30 segments for data storage. Should you need more than 30, you can merely create a second Special Information Type Key Flexfield to store additional special dates. The disadvantage of using a Special Information Type is that these segments cannot be accessed using Oracle Fast Formula.

The primary advantage in using descriptive flexfield segments on the Person window is that these segments can be accessed using Fast Formula. A negative to using the Person descriptive flexfield is that there are a limited number of segments. Additionally, these segments can be defined as required segments that may or may not be a good idea. When a new employee is added, you would be forced to enter this information which would prevent you from accidentally overlooking this information. However, this required entry will have a negative side effect when entering contacts or applicants since the Person window is also used for entry of people other than employees. Every person definition would require a value in these descriptive flexfield segments, which makes little sense.

Our conclusion: We prefer using the Person window Descriptive Flexfield as long as there are a sufficient number of segments available. If you intend to access these dates using Oracle Fast

Formula, then you should use the Descriptive Flexfield. If you intend to access these dates through a custom process (for example, a process that feeds time cards and other earnings to Oracle PayMIX), then your placement of these dates is only a matter of personal preference.

Our wish: Wouldn't it be nice if a future release of the Oracle HR/Payroll product included a reserved flexfield on the Person window that was dedicated to special dates? Each segment would be a date format that allowed for a user-definable name. Clearly, there is no way that Oracle or anyone else could create a hard-coded definition of every possible benefit eligibility date that a client needs. Every client has unique, and sometimes non-standard, eligibility criteria. If these dates existed, the Element Link window could be slightly modified. Eligibility based on length of service could be modified to add a drop-down list containing hire date plus all user eligibility dates. Now, your length of service (6 months, 3 years, etc.) would be based on the value from the selected date segment (or hire date). This functionality would enhance the product and not interfere with the goals of the future release of Oracle Advanced Benefits.

## 401k Calculations / Company Match

One of the most common benefit plans that companies offer employees is a 401k savings plan that usually includes a company match. This section will outline how we used standard Oracle features to implement a typical 401k plan. The features we used include:

- Balance Definitions
- Global Values
- Fast Formulas
- Formula Results

Our sample 401k plan has the following rules. Employees can elect to contribute up to 15% of their eligible earnings to the plan. The company will match 50 cents to every dollar up to 10% of the eligible earnings. The deductions should stop when employees reach the IRS maximum amount allowed for 401k plans.

Our first task is to setup a balance that will define eligible earnings. Use the balance window to

define this new balance (Navigation path: Compensation and Benefits™ Balance). In our example, we will name our balance 401K\_Earnings. After completing the initial window, click on the Feeds button and select from the pick list all of the earnings that should be used when calculating the pre-tax deductions. Click on the Dimensions button and select from the pick list the desired dimensions. We will use Assignment within GRE Run in our formula. This balance will be used to calculate both the actual deduction and the company match.

Next, we will define a Global Value that will store the maximum year-to-date deduction allowed under the IRS rules. Navigate to the Global Values window (Navigation Path: Compensation and Benefits™ Global Values) and define this amount. The 1998 amount is \$10,000. Each year you can update this value using this window. In our example, we will name this value 401K\_MAX.

We are now ready to define our Pre-Tax deduction. Navigate to the Deduction window (Navigation Path: Compensation and Benefits™ Deductions) to setup this deduction. Use Pre-Tax Deduction as the classification and Deferred 401k as the category. Our example will use Percent of Earnings as the Calculation Rule. This will allow Oracle to generate a default formula that we can modify for our use.

Next, we will define the element that will store the company match portion of the 401K plan. Navigate to the Element Description (Navigation path: Compensation and Benefits™ Element Description) and setup this element as an Employer Liability. Our formula will pass the calculation results to this element.

We are now ready to make changes to the generated formula to calculate both the deduction and the company match. Navigate to the Write Formula window (Navigation Path: Compensation and Benefits™ Write Formulas), and query the formula that Oracle created for us. Normally, the name of the formula is XXXXX\_PERCENT\_OF\_EARNINGS, where XXXXX is the name of the deduction we defined above. Click on the Edit button to bring up the Edit window. Scroll down to the section where the actual deduction is calculated. Oracle inserts a comment right before the logic that instructs users where to put their custom modifications.

The default formula uses the Regular Earnings balance. As the formula comments suggest, we want to replace the Regular Earnings balance with the one we created above. We suggest that you create a separate validation formula to validate the maximum 15% rule. You can also use the Maximum field on the Input Value to control the maximum allowed for the percentage.

The key modification to this formula is to include logic that calculates the company match. The results of this calculation should be stored in a variable that is included on the RETURN statement. This will allow us to pass this result to the Employer Liability element we created above. To do this, navigate to the Formula Results window (Navigation Path: Compensation and Benefits™ Formula Results) and select from the pick list the Pre-Tax deduction for our 401K plan. With the cursor down in the Formula Results region of the window, click on the New Record icon to open up a new line. Select from the pick list the name of the variable that stored the result of the company match. In our example, we used EMPL\_CONTR. This will automatically pass the results of our formula to the Employer Liability element.

After keying in the necessary links for each of our elements, we are now ready to enroll people into our plan. We only have to add one element, the Pre-Tax deduction element, to Element Entries of participating employees. We enter the desired percentage in the corresponding input value. The formula will handle the rest.

By using standard Oracle features, we were able to administer a typical 401K plan that can enforce IRS rules, calculate the company match and enforce any special plan rules.

## **Laser Printing of Payroll Checks**

The standard delivered solution for printing payroll checks is to spool the results of the Check Writer process to a printer that is mounted with preprinted forms. The user is responsible for providing the mechanism for getting this done. The easiest solution is to purchase the preprinted forms from an Oracle supported vendor, such as Evergreen. One of our clients did use this approach and spooled the results

from Check Writer to a laser printer with minimal effort. This solution, however, requires the user to match up the check numbers preprinted on the form with the check numbers generated by Check Writer. If the printer “eats” one of the forms, the check numbers can very easily get out of sync. Since it is undesirable to preprint signatures on the check form, you also have the issue on how to turn these unsigned checks into valid cashable documents.

We have solved this problem two different ways. Both of these solutions did **not** attempt to modify the results from Check Writer. Instead, we introduced processes that took the standard results from Check Writer and inserted the needed information.

The first solution used the macro features of Microsoft Excel. We made a bit-mapped image of the check signature and stored in on a controlled PC. Next, we created an FTP script that prompted the user to key in the concurrent request ID of the Check Writer process. This script then copied the output file from the UNIX server to the controlled PC. The user then started up Microsoft Excel and executed the specially designed macro. This macro stepped through the copied file and inserted the signature into the appropriate spot. The final step was to spool the resultant spreadsheet to the laser printer where the Evergreen forms were already loaded. We still had the problem of keeping the check numbers in sync. However, we did not have to run the checks through a check signer.

This solution is easy to implement and uses tools that probably already exist. It does have its drawbacks. First, Excel can only handle approximately 450 checks in one spreadsheet at a time (memory constraint). Our FTP process had to recognize this limit and break up the Check Writer file into multiple import files. Each file had to be processed through the macro and printed separately. Obviously, this solution is unthinkable for check runs that number into the thousands. Additionally, it does not address the check number syncing issue.

Our second solution provides a much cleaner approach to the problem. Once again, we did not attempt to modify the standard Check Writer process. This time we used a third party product called FormsXpress from Optio Software. This

product is designed to receive output files and allows users to manipulate the results. By using this product, we totally redesigned the look and feel of the check document. Normally, the actual check prints at the top of the form followed by the Statement of Earnings. Our client wanted to flip this around and print the actual check on the bottom of the form. We also inserted the company logo and signature. The biggest gain was that we were able to print the check number in the MICR format at the bottom of check document. This allowed us to use standard printer stock and not preprinted forms.

We were surprised with the ease of use of this product. It does require the use of a laser printer with the special MICR cartridge. There are also obvious control issues, since this solution is very easy to implement. Most printers provide some hardware locking features that allow you to physically control the use of the MICR cartridge. All in all, we feel this second solution offers the most flexibility without making any software changes to Check Writer.

## **Paid Time Off Accruals**

Many companies require their employees to earn their vacation and sick time through accruals. Oracle provides a standard mechanism that allows employees to accrue vacation and sick time on a pay period basis. We have found, however, that many companies accrue paid time off in ways that do not easily fit into Oracle’s standard mechanism. For example, a company’s accrual rate might be based on hire date, but carryovers occur on their anniversary date instead of January 1. Oracle’s standard solution performs carryovers on January 1. Other business rules may require companies to develop their own accrual solutions.

One of our clients had a very complex vacation and sick accrual process. Our solution used standard Oracle functionality but still allowed the company to continue to support their unique business rules. The tools we used included Balance Creation, Table Structures, Fast Formulas, Formula Results and Balance Feeds. We could present an entire paper on this topic alone. Instead, we will present a simplified

example of the one we implemented and will only address the vacation time portion.

For our example company, let's say that they accrue vacation hours per pay period based on the table below:

Years of Service	Vacation Rate
0 to 4 years	.33 days
5 to 10 years	.66 days

Next, employees can carryover up to 320 hours on their anniversary date. Any hours over this amount are lost. Finally, employees can "sell" vacation back to the employer. The goal of the accrual process is to properly accrue at the rate describe above, carryover the allowed amount at their anniversary date, and properly decrement the balance when vacation is either taken or sold.

The first step is to create the balance that will hold the total hours available for vacation. Navigate to the Balance window (Navigation path: Compensation and Benefits™ Balance) to define our balance. Make sure one of the dimensions includes Lifetime To Date (LTD). This will be the balance dimension that will track the total accrued hours, since any year-to-date balances would be automatically zeroed out at the beginning of a new calendar year.

Our next step is to define the table above using the Table Structure window (Navigation Path: Compensation and Benefits™ Table Structure). The rows will contain the range of years of service, and the columns will have the corresponding accrual rates. Next, we navigate to the Table Values window to log in the actual accrual rates.

Next, we need to define the element that will be assigned to each of the eligible employees. We defined this element as an Information element and added any needed input values necessary. In our example, we will input a vacation date that the accruals should be based. Generally, this is the same as the hire date but could be different if there was a break in service.

Now we create our Fast Formula that will be attached to this element. It will use the vacation date and look up the accrual rate in our table. It

will pass the results to an output variable on the RETURN statement.

Our next element is the key to the solution, which we will call the Hold Element, which is another Information Element. We need an element that does nothing more than process the balance feeds. The accrual formula will pass its output as an indirect result to an input value on this Hold Element called Accrued Hours. The Balance Feeds for the Hold Element will use the Accrued Hours input value to add to our Vacation Hours Balance created above.

We use the same approach for any vacation taken. The number of hours taken must be passed to another input value on the Hold Element called Taken Hours. Once again, the Balance Feeds for the Hold Element will use the Taken Hours input value to subtract from our Vacation Hours Balance.

We will use the Formula Results screen to create this link between the formula and the Hold Element. Navigate to the Formula Results window (Navigation Path: Compensation and Benefits™ Formula Results) and select from the pick list the Accrual Element created above. Assign the accrual formula created above as a Standard formula. In the Formula Results region, define the output variable that has the accrual rate as an indirect result to the input value Accrued Hours in our Hold Element. Use the same approach for the formula that calculates the vacation. The output variable that has the total hours taken must be defined as an indirect result to the input value Taken Hours in our Hold Element.

The logic for the carryover process must be handled in the accrual formula. First, we must determine if the anniversary date occurs during the period being processed. If so, then we must determine if the life-to-date vacation balance will exceed the allowed limit of 320 hours. If so, we must adjust the Accrued Rate for this run so that the resultant balance will equal 320 hours.

This discussion has outlined the approach we took to solve this complex issue. We also developed several custom reports to document the results of the accrual process. **It is important to note that we met the client's requirements for**

**their complex vacation accrual process without customization to standard Oracle functionality.**

## Fast Formulas

Throughout this paper, we have demonstrated the power of Fast Formulas and its ability to solve our special challenges. This section will summarize some recommendations for Fast Formulas.

The biggest advantage of Fast Formulas is the ability to access “Database Items” when performing payroll calculations. Database Items are pieces of information that Oracle automatically maintains for you. Examples of useful database items include:

- Year-to-date 401k deduction amount
- Total Regular Earnings for current pay run
- An input value on another element
- Global Values
- Segments from a Key Flexfield

Our first recommendation uses the last example from the preceding list. Many times Oracle HR/Payroll users must decide where to store needed information. Do you use the People Group Flexfield, or should you use Special Information Types? If you need the information in a Fast Formula, then use the People Group Flexfield. At the time of this writing, Oracle HR does not create a database item for information stored in a Special Information Type.

Several of our solutions use the power of Indirect Results. This is where the results of one formula can be passed on to another element. When using this powerful feature, make sure that the receiving element will process **after** the generating element. You accomplish this task by assigning the receiving element a higher priority number than the generating element. We have used this feature to calculate imputed income and deduction amounts in the same formula and pass the results to the corresponding element. When doing this, do not forget to handle the situation when there is insufficient pay to cover a deduction.

Many times, we have created Table Structures to store necessary rates for payroll calculations. The Fast Formula statement GET\_TABLE\_VALUE can be used to retrieve these rates. We have used this feature when rates are different based on either a location or a value from one of the People Group Flexfield segments.

## Year to Date Balance Conversion

During initial conversion and implementation, companies are faced with the decision on whether to implement in the middle of a calendar year, or to wait until January 1. Clearly, the job is easier when you can wait until the start of a new calendar year. However, this is generally the busiest time of the year for a payroll department. Also, many companies want to phase in different locations into the new system throughout the year. For many reasons, some companies will have to transfer period balances from the current payroll system into Oracle.

One of the most unique features of Oracle Payroll is that it does not explicitly store year-to-date, quarter-to-date, and month-to-date balances. These numbers are automatically derived by calculating the balances as they would have appeared at the current session date. This date can be arbitrarily set by using a feature called date-tracking. You can literally see year-to-date, quarter-to-date and month-to-date results for any date in the past, as long as the Oracle Payroll system was active on that date.

Although this feature is very powerful, it complicates mid year conversions. Oracle needs to have a starting point to begin these calculations. Therefore, you must perform an initial balance load to establish this starting point. Our first recommendation is to coincide the implementation date with the start of a quarter. This will allow you to only need to provide the year-to-date balances as of the starting date. If the implementation date is not at the start of a quarter, you will have to load quarter-to-date values additionally. Moreover, if your implementation date is not at the beginning of a month, you will need to load month-to-date values.

Balance loads can be broken down into two parts. The first part is the easiest. This is loading a balance for each earning and deduction element you defined in Oracle. Strangely enough, this part is not required. Oracle does not use any these balances when calculating taxes or preparing the end of the year W2s.

The second and most difficult part, the tax balances must be addressed next. Oracle Payroll maintains many tax-related balances with its tight integration to Vertex. Since Vertex has no knowledge on how taxes were calculated in the legacy system, you must manually provide the numbers to populate the Vertex required balances. Below is a partial list of all of the balances required just to calculate State Unemployment Insurance (SUI)

- SUI EE Gross
- SUI ER Gross
- SUI ER Subj Whable
- SUI ER Taxable
- SUI ER Liability

There are many other tax bodies that must be loaded in a similar fashion. For companies with sites in different states, there may be different rules on what earnings should go in each balance.

How do you go about determining what should go in each balance? Oracle has published a white paper titled "Oracle HRMS TRM Supplement: US Legislative Balance Initialization" that lists the required balances. You must determine what numbers should go in each balance. We start with the Taxability Rules window to drive this process. This window (Navigation Path: Compensation and Benefits <sup>TM</sup> Taxability Rules) outlines which earnings are taxable for each tax body. It also outlines which pre-tax deductions should be subject to each tax body. Use these rules to determine which earnings and deductions make up each balance. We generally like to produce a spreadsheet that explicitly provides a roadmap that a legacy programmer can use to prepare a file of these balances.

The next step is to load these numbers into the Oracle-provided interface tables. These tables are described in the Oracle HRMS Implementation Guide. Once baded, you must first run the Initial Balance Structure Creation

process from the application. This process will create the necessary initial balance feed elements and links. Once completed, the next step is to validate the table load by running the Initial Balance Load Process using the Validate option. This will verify that the interface table was loaded properly. Once verified, rerun the Initial Balance Load Process with the Transfer option. This will actually perform the balance load. You have the option to undo the entire process if you need to start from scratch.

Once the load is completed, you can review the balances on-line using the Employee Balance window. (This functionality actually did not work for us at our last client site. Oracle has stated that this has since been fixed). The final task is to do a test parallel payroll run. We recommend doing this since many of the tax calculations are self-adjusting. This allows problems with the balance load to "stick out like a sore thumb". For example, if we understated the subject wages for SUI, Vertex will try to self-adjust the SUI liability down using negative numbers. Likewise, if we did not properly load the Medicare Withheld balance, Vertex will try to catch up all of the deductions with one large deduction. Both of these deductions really do stick out.

Because of the complex nature of Oracle tax balances, and the possibility of each state having unique subject tax rules, the balance load process is very complicated. It is also a very iterative process requiring many attempts before all of the problems are ironed out. Do not underestimate the task involved.

## Conclusions

Despite the many challenges for a successful Human Resources or Payroll implementation, we have found the Oracle HRMS product family to be quite credible. No product exists that can do all things for all people. The flexibility and processing power of the Oracle HRMS products have left a number of clients with an excellent long-term solution. Because of this product flexibility, we have solved customer-specific needs while minimizing customization needs. Our wide-variety of customer-specific solutions

(except for pay scales) that have been described in this paper have been implemented without customizing the Oracle products themselves. When you solve customer-specific needs without customization, you minimize future maintenance costs. This, we conclude, is a significant reason why the Oracle HRMS product family offers clients an excellent long-term solution.

### **About the Authors**

Ken Conway and Bill Stratton are consultants who are exclusively dedicated to Oracle HRMS implementations for BOSS Corporation (Better Organization Service Solutions). Ken and Bill have contributed in both a management and hands-on role in the implementation of several of the first implementations of Oracle Human Resources with Oracle Payroll that have gone into production in the United States. They have been involved in the Oracle BETA program during the early releases of the Oracle HR/Payroll products. As members of the BOSS Corporation HR/Payroll Systems Division, Ken and Bill join an experienced team, which provides consulting services to clients throughout North America.