# Oracle® Human Resources Management Systems

FastFormula User Guide

Release 12.2

**Part No. E59058-08**

May 2022

ORACLE

Oracle Human Resources Management Systems FastFormula User Guide, Release 12.2

Part No. E59058-08



Primary Author: Gowri Arur

# Contents

# Send Us Your Comments

**Oracle Human Resources Management Systems FastFormula User Guide, Release 12.2**

**Part No. E59058-08**

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Oracle E-Business Suite Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: appsdoc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

# Preface

## Intended Audience

Welcome to Release 12.2 of the *Oracle Human Resources Management Systems FastFormula User Guide*.

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area
- Oracle Human Resources Management Systems (HRMS)

  Oracle HRMS is a major component of the Oracle E-Business Suite of applications. If you are unfamiliar with Oracle HRMS, then Oracle suggests that you attend one or more of the Oracle HRMS training classes available through Oracle University.
- The Oracle Applications graphical user interface

  To learn more about the Oracle Applications graphical user interface, read the *Oracle E-Business Suite User's Guide*.

See Related Information Sources on page x for more Oracle E-Business Suite product information.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.

com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

## Structure



## Related Information Sources

Oracle HRMS shares business and setup information with other Oracle Applications products. Therefore, it is advised that you reference other user guides and implementation guides when you set up and use Oracle HRMS.

**Guides Related to All Products**

*Oracle E-Business Suite User's Guide*

This guide explains how to navigate, enter and query data, and run concurrent requests using the user interface (UI) of Oracle E-Business Suite. It includes information on setting preferences and customizing the UI. In addition, this guide describes accessibility features and keyboard shortcuts for Oracle E-Business Suite.

*Oracle Application Framework Personalization Guide*

This guide covers the design-time and run-time aspects of personalizing applications built with Oracle Application Framework.

*Oracle E-Business Suite Maintenance Guide*

This guide contains information about the strategies, tasks, and troubleshooting activities that can be used to help ensure an Oracle E-Business Suite system keeps running smoothly, together with a comprehensive description of the relevant tools and utilities. It also describes how to patch a system, with recommendations for optimizing typical patching operations and reducing downtime.

*Oracle E-Business Suite Security Guide*

This guide contains information on a comprehensive range of security-related topics, including access control, user management, function security, data security, and auditing. It also describes how Oracle E-Business Suite can be integrated into a single sign-on environment.

*Oracle E-Business Suite Setup Guide*

This guide contains information on system configuration tasks that are carried out either after installation or whenever there is a significant change to the system. The activities described include defining concurrent programs and managers, enabling Oracle Applications Manager features, and setting up printers and online help.

*Oracle E-Business Suite Flexfields Guide*

This guide provides flexfields planning, setup, and reference information for the Oracle

E-Business Suite implementation team, as well as for users responsible for the ongoing maintenance of Oracle E-Business Suite product data. This guide also provides information on creating custom reports on flexfields data.

**Guides Related to This Product**

*Oracle Human Resources Management Systems Implementation Guide*

Learn about the setup procedures you need to carry out to implement Oracle HRMS successfully in your enterprise.

*Oracle Human Resources Management Systems Configuring, Reporting, and System Administration Guide*

Learn about extending and configuring Oracle HRMS, managing security, auditing, and information access.

*Oracle Human Resources Management Systems Enterprise and Workforce Management Guide*

Learn how to use Oracle HRMS to represent your enterprise. This includes setting up your organization hierarchy, recording details about jobs and positions within your enterprise, defining person types to represent your workforce, and also how to manage your budgets and costs.

*Oracle Human Resources Management Systems Workforce Sourcing, Deployment, and Talent Management Guide*

Learn how to use Oracle HRMS to represent your workforce. This includes recruiting new workers, developing their careers, managing contingent workers, and reporting on your workforce.

*Oracle Human Resources Management Systems Payroll Processing Management Guide*

Learn about wage attachments, taxes and social insurance, the payroll run, and other processes.

*Oracle Human Resources Management Systems Compensation and Benefits Management Guide*

Learn how to use Oracle HRMS to manage your total compensation package. For example, read how to administer salaries and benefits, set up automated grade/step progression, and allocate salary budgets. You can also learn about setting up earnings and deductions for payroll processing, managing leave and absences, and reporting on compensation across your enterprise.

*Oracle Human Resources Management Systems FastFormula User Guide*

Learn about the different uses of Oracle FastFormula, and understand the rules and techniques you should employ when defining and amending formulas for use with Oracle applications.

*Oracle Self-Service Human Resources Deploy Self-Service Capability Guide*

Set up and use self-service human resources (SSHR) functions for managers, HR Professionals, and employees.

*Oracle Human Resources Management Systems Window Navigation and Reports Guide*

This guide lists the default navigation paths for all windows and the default reports and processes as they are supplied in Oracle HRMS.

*Oracle Performance Management Implementation and User Guide*

Learn how to set up and use performance management functions. This includes setting objectives, defining performance management plans, managing appraisals, and administering questionnaires.

*Oracle Succession Planning Implementation and User Guide*

Learn how to set up and use Succession Planning functions. This includes identifying succession-planning requirements, using talent profile, organization chart, suitability analyzer, and performance matrices.

*Oracle Human Resources Management Systems Approvals Management Implementation Guide*

Use Oracle Approvals Management (AME) to define the approval rules that determine the approval processes for Oracle applications.

*Oracle iRecruitment Implementation and User Guide*

Set up and use Oracle *i*Recruitment to manage all of your enterprise's recruitment needs.

*Oracle Learning Management Implementation Guide*

Learn how to implement and configure Oracle Learning Management (OLM).

*Oracle Learning Management User Guide*

Use Oracle Learning Management to accomplish your online and offline learning goals.

*Oracle Time and Labor Implementation and User Guide*

Learn how to capture work patterns, such as shift hours, so that this information can be used by other applications, such as General Ledger.

*Oracle Labor Distribution User Guide*

Learn how to maintain employee labor distribution schedules, distribute pay amounts, encumber (commit) labor expenses, distribute labor costs, adjust posted labor distribution, route distribution adjustment for approval, and manage error recovery processes. You also learn how to set up effort reporting for Office of Management and Budget (OMB) compliance.

## Integration Repository

The Oracle Integration Repository is a compilation of information about the service endpoints exposed by the Oracle E-Business Suite of applications. It provides a complete catalog of Oracle E-Business Suite's business service interfaces. The tool lets users easily discover and deploy the appropriate business service interface for integration with any system, application, or business partner.

The Oracle Integration Repository is shipped as part of the Oracle E-Business Suite. As your instance is patched, the repository is automatically updated with content appropriate for the precise revisions of interfaces in your environment.

## Do Not Use Database Tools to Modify Oracle E-Business Suite Data

Oracle STRONGLY RECOMMENDS that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle E-Business Suite data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL*Plus to modify Oracle E-Business Suite data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle E-Business Suite tables are interrelated, any change you make using an Oracle E-Business Suite form can update many tables at once. But when you modify Oracle E-Business Suite data using anything other than Oracle E-Business Suite, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle E-Business Suite.

When you use Oracle E-Business Suite to modify your data, Oracle E-Business Suite automatically checks that your changes are valid. Oracle E-Business Suite also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

# 1

# FastFormula

## Oracle FastFormula Overview

Formulas are generic expressions of calculations or comparisons you want to repeat with different input values. Formulas take input from a window, a database, or a process, such as a payroll run and they return values or messages.

Oracle FastFormula is a simple way to write formulas using English words and basic mathematical functions. You can use information from your database in formulas without learning the database structure or a programming language.

### Uses of Oracle FastFormula

In HRMS, Oracle FastFormula is used for validation, to perform calculations, and to specify rules. Here are some examples.

In Payroll, you use formulas to:

- Validate element inputs
- Calculate element pay values and run results during payroll processing
- Specify the rules for skipping an element during payroll processing
- Perform legislative checks during a payroll run

In Compensation and Benefits Management, you use formulas to:

- Specify the rules for Paid Time Off accrual plans, such as how much time is accrued and over what period, when new starters are eligible to begin accruing time, and how much time can be carried over to the next accrual term
- Define custom calculations for benefits administration
- Calculate the duration of an absence given the start and end dates and times

- Create rules for benefits administration such as eligibility determination

In People Management, you use formulas to:

- Check that element entry values are valid for an assignment
- Specify the criteria for including an assignment in an assignment set and to edit assignment sets
- Configure the people management templates in a number of ways such as supplying additional information to be available from fields on the template and validating field entries
- Define collective agreements
- Generate custom global person number sequences for employees, applicants, and contingent workers

You also use formulas to define Oracle Business Intelligence Systems reports, to select the database information you want to display in a QuickPaint report, and to perform calculations for the report.

You can easily create and maintain business rules by calling formulas from the Custom Library. For example, you can use formulas to validate data entry in forms by calling formulas from the Custom Library, and to check that entries made to a user table are valid. You can then call these business rules from other PL/SQL applications.

When you write a formula, you specify for which of these purposes you will use it.

## Components of Formulas

Formulas are made up of a number of different components. These can include assignment statements, different types of input including database items, functions, nested expressions, and conditions. See: Introduction to Formula Components, page 1-10

When writing formulas, there are a number of techniques you should use to ensure your formulas are easy to use and understand. See: Formula Writing Techniques, page 1-16. There are also rules for using each type of component that you need to follow. See: Formula Reference, page 1-23

## Oracle FastFormula Functions

Oracle FastFormula provides functions that manipulate data in various ways. These include:

- Text functions such as the GREATEST function that compares the values of all the text string operands and returns the value of the operand that is alphabetically last

- Numeric functions such as the ROUND function that rounds off a numeric value to the specified number of decimal places
- Date functions such as the ADD_DAYS function that adds a number of days to a date
- Data conversion functions such as the CONVERT function that converts a character string from one character set to another

Other types of functions provided by Oracle FastFormula include functions:

- To get values from tables such as the GET_LOOKUP_MEANING function that enables Oracle FastFormula to translate a lookup code into a meaning
- For accrual type formulas such as the GET_START_DATE function that returns the date at which the accrual formula should start calculating accruals
- That allow you to call another formula, either once or in a loop such as the CALL_FORMULA function that runs a named formula with no inputs and no outputs
- To set and use globals in SQL*Plus from within your formulas such as the ISNULL set of three functions that test whether a text, numeric, or date value is NULL

See: Functions, page 1-33

## Supplied Formulas

Formulas are predefined for all the tax calculations required for Oracle Payroll. You should not edit these formulas. When tax rules change, you will automatically receive updates.

There are also a number of formulas predefined for accrual plans. You can use these formulas as supplied, or customize them to match the rules of your own plans.

Depending on the legislation, the system may automatically generate some payroll formulas to define earnings and deductions elements, and you can create as many other formulas as you require to process the elements you define.

Formulas for QuickPaint reports and assignment sets can be generated from criteria you enter in windows. You can edit these generated formulas to add more functionality.

## Database Items

There are two types of database items available to you in Oracle HRMS for writing formulas and defining QuickPaint reports. Static database items come as part of the system and you cannot modify them. Dynamic database items are created by Oracle HRMS processes whenever you define new elements or other related entities. See: Database Items, page 1-62

### FastFormula Transaction Manager Concurrent Manager

To use the FastFormula Assistant, you must ensure the FastFormula Transaction Manager concurrent manager is running on your environment. A system administrator can setup the menus, add the functions and activate the concurrent manager. See: Defining Concurrent Managers, *Oracle E-Business Suite Setup Guide*

## Oracle FastFormula

### What is Oracle FastFormula?

Oracle FastFormula is a simple way to write formulas using English words and basic mathematical functions. You can use information from your database in formulas without learning the database structure or a programming language.

### How can you use formulas?

You can use formulas to:

- Calculate element pay values and run results during payroll processing.
- Check that element entry values are valid for an assignment.
- Check that entries made to a user table are valid.
- Specify the criteria for including an assignment in an assignment set.
- Create rules for benefits administration, such as eligibility determination
- Select the database information you want to display in a QuickPaint report, and perform calculations for the report.
- Specify the rules for skipping an element during payroll processing.
- Prepare records in the format you require for the magnetic tape writer. For most countries, the Oracle localization team has written the required formulas.
- Perform legislative checks during a payroll run.
- Specify the rules for Paid Time Off accrual plans, such as how much time is accrued and over what period, when new starters are eligible to begin accruing time, and how much time can be carried over to the next accrual term.
- Calculate the duration of an absence, given the start and end dates and times.
- Generate custom global person number sequences for employees, applicants, and

contingent workers.

- Configure the people management templates in a number of ways, including supplying additional information to be available from fields on the template, and validating field entries.

- Validate forms by calling formulas from the Custom Library. Calling formulas from the Custom Library also allows you to easily create and maintain business rules.

  For more information about calling formulas from PL/SQL, refer to the technical essay Calling FastFormula from PL/SQL, *Oracle HRMS Implementation Guide*.

When you write a formula, you specify for which of these purposes you will use it.

## Are any formulas supplied?

Yes, formulas are predefined for all the tax calculations required for Oracle Payroll. You should not edit these formulas; you will automatically receive updates when tax rules change.

There are also a number of formulas predefined for accrual plans. You can use these formulas as supplied, or customize them to match the rules of your own plans.

When US and Canadian Payroll users define earnings and deductions, the system automatically generates the formulas required to process these elements. When necessary, you can edit these generated payroll formulas. In other legislations, some formulas may be predefined, and you can create as many other formulas as you require to process the elements you define.

Formulas for QuickPaint reports and assignment sets can be generated from criteria you enter in windows. You can edit these generated formulas to add more functionality.

# Uses for Oracle FastFormula

## Uses for Oracle FastFormula

You can use Oracle FastFormula to:

- Calculate your payrolls
- Define the rules for PTO accrual plans
- Define custom calculations for benefits administration
- Define QuickPaint reports
- Validate element inputs or user tables
- Edit assignment sets
- Calculate absence duration
- Configure people management templates
- Set up business rules and call them from other PL/SQL applications
- Define your Oracle Business Intelligence Systems reports
- Define collective agreements
- Define custom global person number sequences
- Define employment categories for EEO reports (US only)
- Calculate ratings for individual competencies and objectives, and calculate a total score for an appraisal

### Payroll Calculations

You can use predefined payroll formulas. When you receive Oracle Payroll, some formulas are predefined in your system. You cannot make any changes to these formulas, but you can copy them and modify the copies.

US and Canadian Payroll users can use Oracle FastFormula to edit the formulas the system generates for the earnings types, other payments and deductions you initiate in Oracle Payroll. You make these edits directly to the generated formula (not to a copy) using the Formula window.

All Oracle Payroll users can use FastFormula to write payroll calculations and skip rules for elements you define yourself to represent earnings and deductions. You can associate more than one formula with each element, to perform different processing for employee assignments with different statuses. US and Canadian Payroll users need only define their own elements and formulas for earnings and deductions with highly complex calculations requiring a number of different calls to the database.

You can write Payroll Run Proration formulas to run after the usual payroll formula and handle proration when employees start work or terminate mid-pay period, or when rates, grades, or other values change, requiring an element to be prorated.

## PTO Accrual Plans

You can use Oracle FastFormula to edit the seeded Accrual type formulas, or to write your own. Each accrual plan needs two formulas: one to calculate gross accrual, and the other to return information to the PTO Carry Over process. You can optionally create a third formula if you want to use the Batch Element Entry (BEE) to make absence entries against accrual plans. This formula is called by BEE to check whether an employee is eligible to use accrued PTO.

See: Accrual Formulas, *Oracle HRMS Compensation and Benefits Management Guide*

## Benefits Administration

You can use Oracle FastFormula to augment the system's benefits administration features. Use formulas to configure your plan design to the requirements of your enterprise. For example, you can write a formula to calculate benefits eligibility for those cases where eligibility determination is most complex.

## QuickPaint Reports

In the Define QuickPaint Report window, you can paste database items from a list into the Report Definition area and enter free text to label the data. When you save your QuickPaint Report definition, a formula is generated automatically. Formulas generated from QuickPaint do not include conditional logic or calculations. You may want to add these features, for example to sum up compensation elements or to report different data for different assignments.

## Validation of Element Inputs or User Tables

You can use Oracle FastFormula to validate user entries into element input values using lookups or maximum and minimum values. However, if you need more complex validation, you can write a formula to check the entry.

You can also use a formula to validate entries into user tables that you define.

## Assignment Sets

When you define assignment sets in the Assignment Set window, Oracle FastFormula generates a formula to define an assignment set from the criteria entered. However, you may want to change the sequence in which the set criteria are checked for each assignment.

## Absence Duration

You can write a formula to calculate the duration of an absence from the start and end dates and times. Your localization team may have created a default formula, but you can override this with your own version.

## Configuring People Management Templates

There are several ways you can use formulas to configure the people management templates:

- A Template Validation formula can check values entered in a field.
- A Template Information formula can specify information to be displayed from the right mouse menu when a user right-clicks in a field.
- A QuickPaint formula can return a text string to display in the Assignment field on the Maintenance window and in the Data Organizer.
- A QuickPaint formula can return message tokens that you can use in a notification message issued from template forms.

## Calling FastFormula from PL/SQL

You can call formulas from PL/SQL applications. This enables direct access to data items and makes it possible to develop custom code for localized business rules.

More detailed information about calling FastFormula from PL/SQL can be found in the technical essay Calling FastFormula from PL/SQL, *Oracle HRMS Implementation Guide.*

## Oracle Business Intelligence Systems (BIS) Reports

Using formulas you can configure your HRMS BIS reports so that they answer the business questions which are important to your enterprise. You can:

- Define how workforce should be counted within your enterprise for the Workforce reports.
- Set up information about how you want to record and report on regular and

overtime hours for the Hours Worked Analysis report.

- Set up the standard hours for your enterprise for the Absence Analysis report.

## Collective Agreements

Using formulas you can calculate whether a person is eligible to receive a collective agreement entitlement. This can be used when defining an eligibility profile to be used in conjunction with a collective agreement, instead of selecting criteria elements. You select the formula as a rule when defining the eligibility profile.

## Global Person Numbering

When you select automatic local numbering for employees, applicants, or contingent workers, Oracle HRMS allocates person numbers from a sequence that is specific to the business group. When you select global numbering, Oracle HRMS allocates person numbers from a single sequence to workers of the relevant type throughout the enterprise.

You can replace the default local or global person number sequence with a custom global sequence by defining a formula for each person type, as appropriate. You may want to use a formula to provide an alphanumeric numbering scheme, for example, or some other variation of the default scheme for any or all person types. For example, you could use a custom global sequence for employees but use the default local or global sequence for applicants and contingent workers. Alternatively, you could use a custom sequence for all three person types by defining three formulas.

## Employment Categories for EEO Reports (US only)

For the EEO4 and EEO5 reports you can use Oracle FastFormula to create a formula of employment categories. You can define a formula for each business group and the EEO reports will pick the list of employment categories from the formula.

## Competency, Objective, and Appraisal Ratings

You can use Oracle FastFormula to:

- Calculate a rating for each competency in a worker's appraisal.
- Calculate a rating for each objective in a worker's appraisal.
- Calculate a total score for the appraisal based on the final scores for competencies and objectives.

See: Writing Formulas for Rating Competencies and Objectives, page 1-143

# Introduction to Formula Components

The following information uses a series of examples to help you understand how to use different components to build your Oracle formulas.

To start with a simple example, suppose you wanted to calculate the pay value for the element Wage by multiplying the number of hours an employee works each week by hourly rate. You could write this formula:

```
wage = hours_worked * hourly_rate

RETURN wage
```

> **Note:** If you want to know the rules which govern the use of a specific component, refer to *Formula Reference*.

## Assignment and Return Statements

The first line is an *Assignment statement* that simply assigns a value to the element Wage. The second line is a *Return statement* that passes back the Wage value to the payroll run.

## Constants and Variables

In this example, the Wage value is calculated, but it could be a *constant* value, such as: `wage = 200`. To calculate the Wage value, Oracle FastFormula needs to get values for the *variables* hours_worked and hourly_rate. They are called variables because they can contain different values depending, in this example, on which assignment the payroll run is processing.

## Data Types

Both variables and constants can be one of three data types:

- numeric
- text
- date

The variables in the Wage example are numeric.

## Types of Input

We have said that Oracle FastFormula needs to get values for the variables hours_worked and hourly_rate. There are three ways it can do this:

- Receiving them as input when the formula is called.
- Finding the values in the database from *database items.*
- Using *global values*, which you enter in the Globals window.

To use a database item or global value in your formula, you simply refer to it by name. You can browse through lists of database items in the Formulas window. To use a value passed to the formula at run time, you must write an *Inputs statement*.

### Input Statements

In our Wage example, suppose that hours_worked is an input value to the element Wage. To pass the element input values to the formula during a payroll run, you define an Inputs statement, as follows:

```
INPUTS ARE hours_worked

wage = hours_worked * hourly_rate

RETURN wage
```

The name you use in the Inputs statement must be the same as the name of the element input value, and multiple words must be joined by underscores. In this example, the input value hours_worked is numeric. If the input value is not numeric, you must tell Oracle FastFormula whether it is text or date. For example:

```
INPUTS ARE start_date (date)
```

### Database Items

Suppose that hourly_rate is a standard rate taken from the Grade Rates table. This is an example of a database item. A database item has a label, or special piece of code, telling Oracle FastFormula the path to take to access the data. Oracle HRMS produces most of the database items you need without you taking any special action. These items include both information unique to your enterprise, which you hold in flexfield segments, and standard information such as assignment numbers and grades.

In the Formulas window, you pick database items from a list. You will see that the name of the database item for a grade rate called hourly_rate is actually grade_hourly_rate_value. This is the name you must use in your formula.

By this naming convention, Oracle FastFormula knows that hourly_rate is a database item from the Grade Rate table. But how does it know which hourly_rate to use from this table? It works this out from the *context* the payroll run provides for each element entry. The context identifies:

- the Business Group
- the element and element link

- the payroll and payroll run
- the employee and employee assignment.

> **Important:** You should use an Inputs statement in preference to database items where possible because this is more efficient.
>
> See: Writing Efficient Payroll Calculation Formulas, page 1-19.

### Global Variables

Use global values to store information that does not change often, but you refer to frequently, such as Company Name, or company-wide percentages used to calculate certain types of bonus. You define the global value and change its value using the Globals window.

See: Defining Global Values, page 1-153

### Local Variables

Local variables exist in one formula only. You can change the value of a local variable by assigning it a value in an Assignment statement. In the Wage example, the variable wage itself is a local variable. It receives a value within the formula by the Assignment statement:

```
wage = hours_worked * hourly_rate
```

> **Note:** You cannot change the value of input values, database items, or global values within a formula.

## Functions

The Assignment statement in the wages example above uses a simple multiplication to calculate the value of the wages element. You can also use addition, subtraction, division, and a number of *functions*. For example:

```
bonus = GREATEST(days_at_work,163) + bonus_rate
```

Here the function GREATEST tells Oracle FastFormula to use the value of the variable days_at_work, if it is greater than 163, and otherwise to use the constant 163.

The data type of variables and constants determines how operators and functions act on the values. For example, the addition operator (+) can be used with numeric or text data, while division can be used with numeric data only.

There are special functions that convert variables from:

- numbers to text (TO_TEXT)

- dates to text (TO_TEXT)
- text to date (TO_DATE)
- text to number (TO_NUM)

See: Functions, page 1-33

## Nested Expressions

The Assignment statement can use as many arithmetic operators and functions as you require. Each function or calculation is one *expression*, and you can nest expressions to create more complex calculations. You must use brackets to make clear to Oracle FastFormula the order in which the calculations are performed. For example:

```
ANNUAL_BONUS = trunc(((((salary_amount/100)*

bonus_percentage)/183)*(days_between(end_period_date,

start_date) + 1)), 2)
```

Oracle FastFormula begins calculating inside the brackets and from left to right, in the following steps:

1. `salary_amount/100`
2. `1. * bonus_percentage`
3. `2. / 183`
4. `days_between (end_period_date, start_date)`
5. `4. + 1`
6. `3. * 5.`
7. `TRUNC(6.,2)`

Notice that TRUNC is another function. It rounds a numeric value down to the number of decimal places specified after the comma (two in this case).

## Incorporating Conditions

In our simple Wage element example, only one value is returned, and it is calculated in the same way for every assignment. However you may need to perform different calculations depending on the particular group of employee assignments, or the time of the year, or some other factors. You can do this by incorporating *conditions* in your formula.

### Simple Conditions

For example:

```
IF age < 20 THEN

  training_allowance = 30

ELSE

  training_allowance = 0
```

The formula checks whether the condition (age < 20) is true or false. If it is true, the formula processes the statement that follows the word THEN. If the condition is not true, the formula ignores this statement and processes any statement that follows the word ELSE. The ELSE clause is optional.

### Complex Conditions

In the example above, the condition compares a variable (age) to a constant (20). The condition can be more complex, comparing expressions that contain functions and arithmetic operators. For example:

```
IF (DAYS_BETWEEN(end_period_date, start_date)+1) >= threshold
```

DAYS_BETWEEN is another function.

We have seen two *comparators*: less than (<) and greater than or equal to (>=). A full list of the comparators you can use appears in the Reference section.

See: Formula Reference, page 1-23

### WAS DEFAULTED

There is a special type of condition called WAS DEFAULTED. Use this to test whether a default value has been placed in an input value or database item. Default values are placed using the Default statement. For example:

```
DEFAULT FOR hourly_rate IS 3.00

X = hours_worked * hourly_rate

IF hourly_rate WAS DEFAULTED

  THEN

    MESG = 'Warning: hourly rate defaulted'
```

In this example, if the database item hourly_rate is empty (NULL), the formula uses the default value of 3.00 and issues a warning message.

### Combined Conditions

You can combine conditions using the *logical operators* AND, OR, NOT.

- Use AND if you want an action to occur when more than one condition is true. For example:

```
IF  (days_between(end_period_date, start_date) + 1) >= 183

AND  employee_status = 'FULL TIME'

THEN  . . .
```

- Use OR if you want an action to occur when any one of two or more conditions is true. For example:

```
IF  stock_level < 10000

OR  order_size >= 1500

THEN  . . .
```

- Use NOT if you want an action to occur when a condition is *not* true. For example:

```
IF NOT (months_between(purchase_date, system_date) => 60

THEN  . . .
```

As with Assignment statements, you may need to use brackets to tell Oracle FastFormula in which order to test conditions. By default, NOT has the highest precedence, followed by AND then OR. So the following condition:

```
IF   X = 6 AND NOT Y = 7 OR P >= 6
```

is interpreted as:

```
IF   ((X = 6) AND (NOT (Y = 7))) OR (P >= 6)
```

How you use brackets can change dramatically the meaning of a formula.

> **Tip:** Use brackets whenever you create multiple conditions, so that the meaning of the formula is clear to other readers.

### Multiple Actions Based On Conditions

We have seen how to make conditions more complex. You can also make the actions performed as complex as you like. In our simple example above, the result of the condition was the assignment of a value to the variable training_allowance. As well as assigning values, you can perform calculations and return values.

For example, suppose you must check whether there are sufficient funds in a bank account before processing a withdrawal:

```
INPUTS ARE acct_balance,

           acct (text),

           debit_amt

IF   acct_balance >= debit_amt

  THEN

  (

     new_balance = acct_balance - debit_amt

     RETURN new_balance

  )

  ELSE

  (

     message = 'Account No. ' + acct + ' - Insufficient Funds'

     message2 = 'Account Balance is ' + TO_TEXT(acct_balance)

     RETURN message, message2

  )
```

Notice that you can return two variables in the same Return statement.

> **Important:** The brackets following THEN and ELSE are essential when you have multiple actions based on a condition. Without them, Oracle FastFormula processes the first statement conditionally and the other statements unconditionally.

# Formula Writing Techniques

When writing formulas there are a number of techniques you should follow to ensure your formulas are easy to read, use and understand.

## Commenting Formula

It is good practice to include comments in your formulas to explain to other people

what the formula does.

So, for example, you can name your formula:

```
/*  Formula:  Attendance Bonus  */
```

and write an explanation of your formula:

```
/*  Use this formula to calculate the annual bonus for
    clerical staff.  Employees receive either a percentage of
    their salary (if they have been present for 183 or more
    days in the last six months), or a pro rata bonus (if they
    have been in attendance for less than 183 days in the last
    six months).  */
```

Oracle FastFormula ignores everything between the comment delimiters: /* and */. You can place comments anywhere in a formula without affecting the formula's performance.

> **Caution:** Do not put a comment within a comment. This causes Oracle FastFormula to return a syntax error.

You can use a comment to explain what part of your formula does. So, for example, you might want a comment explaining who decides the bonus percentage:

```
INPUTS ARE  salary_amount,
            start_date (date),
            end_period_date (date),
            bonus_percentage /*  decided at board level */
```

You can also use comments to `comment out' parts of the formula you do not currently want to use. So, for example, you can include a fifth input of employee_status, ensuring that employees with a status of full time are awarded a bonus. However, as you do not yet have a range of statuses, you do not currently need the fifth input.

```
INPUTS ARE  salary_amount,
            start_date (date),
            end_period_date (date),
            bonus_percentage /*  decided at board level  */
        /*  employee_status (text)  */
```

Use comments and white space freely when entering formulas. This makes the formulas easier to read and understand, and has no effect on performance or memory usage. Use indentation for the same reason, especially when you are using brackets to control the order of processing.

It is good practice to include the following information in a comment at the beginning of a formula:

- Formula title and short statement of its purpose
- Description of formula inputs
- List of variables and constants that may require updating
- Description of the input values of the element that receives the formula's direct result
- Explanation of the formula's calculations
- Administrative information such as the name, address and telephone number of an office administering the earnings, deduction, or charge the formula affects
- The dates of formula modifications, the names of those entering the edits, and possibly the reasons for change

## Alias Statements

Database items are named by the system when it creates them, and sometimes these names are too long to conveniently use in a formula. You cannot shorten the name of a database item (or a global value) itself, but you can set up an alternative shorter name to use within the formula. For example:

```
ALIAS  as_overtime_qualifying_length_of_service AS ot_qls
```

In the rest of the formula, you can use the alias (in this example, ot_qls) as if it were the actual variable.

> **Important:** Using an Alias is more efficient than assigning the database item to a local variable with a short name.

## Default Statements

Use the Default statement to set a default value for an input value or a database item. The formula uses the default value if the database item is empty or no input value is provided when you run the formula. For example:

```
DEFAULT FOR hourly_rate IS 3.00
```

```
X = hours_worked * hourly_rate

IF hourly_rate WAS DEFAULTED

  THEN

    MESG = 'Warning: hourly rate defaulted'
```

This example sets a default of 3.00 for the database item hourly_rate. If hourly_rate is empty (NULL) in the database, the formula uses the default value of 3.00. The formula uses the 'WAS DEFAULTED' test to detect when a default value is used, in which case it issues a warning message.

> **Important:** You must use the Default statement for database items that can be empty. The Database Items window includes a check box labelled Default Required. This check box is checked for database items that can be empty. The Database Items window appears when you choose the Show Items button on the Formulas window.

## Writing Efficient Payroll Calculation Formulas

The following guidelines are generally true for typical payroll runs:

- The longer an element's formula, the longer its processing time.
- The more elements entered for an assignment, the longer its processing time.
- One element associated with a lengthy formula usually processes faster than two related elements each associated with a short formula.
- The overall number of elements and formulas in the system has little effect on processing efficiency. It is the number of elements per assignment that affects processing time.

### Variable Names and Aliases

To improve readability use names that are brief yet meaningful. Name length has no effect on performance or memory usage. Use Aliases if the names of database items or global values are long.

### Input Statements

Use Input statements rather than database items whenever possible. This improves formula processing by as much as a factor of ten. It speeds up the running of your payroll by eliminating the need to access the database for the input values.

*Inefficient:*

```
Salary = Salary_annual_salary / 12

         RETURN Salary
```

*Efficient:*

```
INPUTS ARE Annual_salary

Salary = Annual_salary / 12

         RETURN Salary
```

### Date Literals

Use the TO_DATE function only when the operand is a variable.

*Inefficient:*

```
Start_date = TO_DATE ( '1992-01-12 00:00:00' )
```

*Efficient:*

```
Start_date = '1992-01-12 00:00:00' (date)
```

### Single Expressions

Use a single expression in straightforward formulas where this does not lead to confusion.

*Inefficient:*

```
Temp = Salary / Annualizing_factor

Tax = Temp * 3
```

*Efficient:*

```
Tax = (Salary / Annualizing_factor) * 3
```

### Database Items

Do not refer to database items until you need them. People sometimes list at the top of a formula all the database items the formula might need, thinking this helps Oracle FastFormula process more quickly. However, this in fact slows processing by causing unnecessary database calls.

*Inefficient:*

```
S = Salary

A = Age
```

```
IF S < 20000 THEN

 IF A < 20 THEN

 Training_allowance = 30

ELSE

 Training_allowance = 0
```

*Efficient:*

```
IF Salary < 20000 THEN

 IF Age < 20 THEN

  Training_allowance = 30

 ELSE

  Training_allowance = 0
```

The first example always causes a database fetch for Age whereas the second only fetches Age if Salary is less than 20000.

### Balance Dimensions for UK HRMS

Wherever possible, use balance dimensions for single assignments only in formulas. Multiple assignments require more calculation, leading to slower processing time. The number of genuine multiple assignments in a payroll is not normally high, and the presence of a small number does not lead to any significant increase in overall processing time. There could be a problem, however, if you unnecessarily link balance dimensions for multiple assignments into general formulas.

### Proration Formulas for UK HRMS

You set up proration formulas to enable element values to be calculated accurately if they change during a payroll period, for example, if an employee leaves the company or if their pay rate changes.

For more detailed information on proration, see the Technical Essay entitled Proration available on My Oracle Support.

## FastFormula Assistant Overview

The FastFormula Assistant provides a fast and easy way to create, edit, and copy formulas. This wizard uses an intuitive OA Framework-based interface that guides you through the various tasks of writing a formula, such as including generic formula text, functions, and database items in your formula.

Use the FastFormula Assistant to quickly:

- Generate specific FastFormula templates based on formula types
- Include the required functions and database items in the formula text
- Download formula text into a file for making changes using an external editor and upload it back
- Generate line numbers for formula text so that you can easily locate errors
- Choose and compile multiple formulas

If you are new to formulas or want to make changes without keeping date-tracked records of your formulas, you can use FastFormula Assistant. You can also use the Formula window to create and update formulas. The application keeps the date-tracked history of all the formulas you create and update using the Formula window.

Writing or Editing a Formula, page 1-146

You can access FastFormula Assistant using one of the following options in the HRMS Navigator menu. Your system administrator may restrict your access to either option depending on the security requirements of your enterprise:

- Global FastFormula Assistant, which provides access to formulas across all business groups, allowing you to copy a formula from one business group to another and view formulas from all business groups. However, you can update only those formulas that are within your own business group.
- FastFormula Assistant, which provides access to all formulas in the current business group only.

# Formula Reference

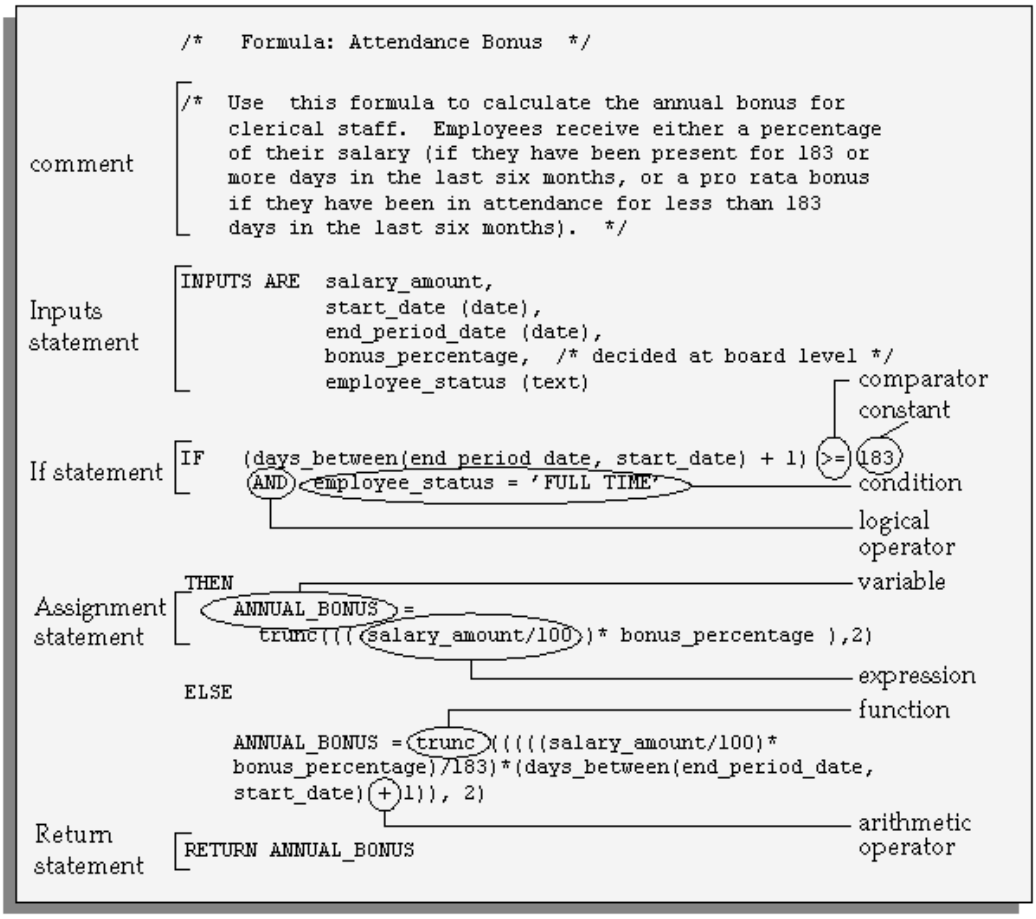
## Formula Reference

Formulas comprise *statements* and *comments*. Statements are instructions telling Oracle FastFormula how to process *constants* and *variables*, which are the basic information units in a formula. The two key types of statement, which describe the formula's calculations, are the Assignment statement and the If statement. These statements can include *expressions*,which manipulate constants and variables using *arithmetic operators* and *functions*. The operators and functions you can use and the results they give depend upon the data type of the constants and variables. In If statements, one expression can be compared to another using *comparators* to create a *condition*. Conditions can be combined using *logical operators*.

There are rules about how you use each of these components. Refer to the component description for more detailed information.

An example of each of these formula components is identified in the sample formula.

***Components in a Sample Formula***




## Input Values in Payroll Formulas

In many formulas for calculating pay, some or all of the information you need comes from the input values of the element.

For example, suppose you pay some salaried employees using a recurring element called Salary. The Salary element has an input value called annual_salary. You need a formula that divides the input value into twelve parts:

```
INPUTS ARE annual_salary

Salary = annual_salary/12

RETURN Salary
```

**Important:** When you use an Inputs statement, you need to make sure

that none of the input values can have a value of null because this causes the formula to fail. You can avoid this problem by using the Default statement.

Using an Inputs statement is the most efficient way to access the input values of the element with which the formula is associated. However, if the formula uses the input values of other elements, it must access the database items for them.

For example, if you want to use the input value annual_salary in the formula to calculate the element Bonus, you use the database item as follows:

```
IF Salary_annual_salary > 20000

THEN

Bonus = bonus_rate * (sales_achieved - sales_threshold)
```

Notice that the database item name is in two parts: the input value (annual_salary) name prefixed by the element name (Salary). This is the naming convention for the database items of element input values.

## Multiple Entries of Element Input Values

When you define an element, you can enable multiple entries of the element within a payroll period.

For example, suppose you use the element Wages to pay some weekly-paid employees. The Wages element has the input value hours_worked. Each week, you regularly make five entries for the input value hours_worked.

To calculate Wages, you can multiply the hours worked each day by the employee's standard rate from the grade rates table, so that your formula looks like this:

```
INPUTS ARE hours_worked

Wages =  hours_worked * standard_rate

RETURN Wages
```

During the payroll run, the formula processes five times, creating five separate pay values, one for each entry.

Now consider using the database item Wages_hours_worked instead of an Inputs statement. The database item for an entry value **sums up** all the entries made in the payroll period.

This is a great convenience when referring to input value totals for a payroll period. However, you must be sure that it is the totals that you want to use. In this example, using the database item produces the wrong result.

Wages_hours_worked gives a single value that is the **sum** of the five entries in each

weekly payroll period. When the payroll runs, the formula processes five times, each time calculating wages using the total hours worked in the week.

> **Important:** If multiple entries are enabled for an element, be careful when using database items for the element's entry values. These database items hold the sum of all the entries. This includes entries made as indirect formula results during the payroll run.

### Database Items for Numeric Values Only

Entry value database items are **not** created for input values with units of character, date, or time when multiple entries are enabled for the element. This is because entry values of these types cannot be summed over the payroll period. Only numeric values can be summed. Numeric input values are those with units of:

- Hours
- Integer
- Money
- Number

Notice that other database items, such as default, minimum, or maximum, may be created for non-numeric input values.

## Constants

Every piece of information that you can manipulate or use in a formula is a constant or a variable.

The data type of a constant or variable describes what kind of data the constant or variable holds. Generally, you use constant and variables of the same data type in an expression.

*Constants* are actual values you use in a formula. Oracle FastFormula uses constants directly rather than holding them in a variable.

There are three types of constant:

- numeric
- text
- date

## Numeric Constants

Enter numeric constants without quotes. Precede negative numbers with a minus sign (-). Numbers may have a decimal component after a decimal point. Do not use exponents and floating point (scientific) notations. So, for example, you cannot use $2^2$ or $10e^{1.24}$ as numeric constants. Do not use commas or spaces in a number. So, for example, you cannot use 10,000 or 10 000.00 numeric constants.

Examples of valid numeric constants are:

- 63
- 3.55
- -2.3
- - 0.33
- - .2
- 10000

## Text Constants

Enclose text constants in single quotes. They may contain spaces. You can represent the single quote character in a text constant by writing it twice ('). Note that this is not the same as the double quote ("). Examples of valid text constants are:

- `J. Smith'
- `P O''Donnell'
- `1234'
- `Manager'
- `12 Union Road'
- `The Bonus this year is 23%'

## Date Constants

Date constants contain a date. Enclose dates in single quotes and follow immediately with the word date, in brackets. Use the format YYYY-MON-DD HH24:MI:SS or DD-MON-YYYY. It is recommended that you use the first format if you want to compile the formula under different language settings.

Examples of valid date constants are:

- `1989-03-12 00:00:00' (date)
- `12-MAR-1989' (date)

## Variables

You use *variables* in a formula to access information. Variables can have frequently changing values.

The data type of a variable determines the type of information the variable holds:

- numeric
- text
- date

You do not have to specify what type you want a variable to be. Oracle FastFormula works out the type from how you use the variable. For example, if you set a variable to `J. Smith', this is interpreted as a text variable.

The system also warns you if you try to perform any inconsistent operations, such as trying to add a number to a text string.

There are three classes of variable in Oracle FastFormula:

- **Local Variable -** Variables that occur in a single formula only.
- **Global Value -** Values that can occur in any formula.
- **Database Item -** Items that exist in the application's database.

The variable class determines how a formula uses it.

### Local Variables

*Local variables* occur in a single formula only. You can change a local variable within the formula by assigning a value to it using an Assignment statement.

You can use local variables to store data in a formula. You might want to hold data temporarily while you perform some other calculations, or to pass data back to the application.

Below is an example showing the use of a local variable, annual_leave.

```
/* Formula: Annual Leave Formula */

IF years_service >= 10

THEN
```

```
 annual_leave = 25

ELSE

 annual_leave = 20 + FLOOR (years_service/2)

RETURN annual_leave
```

## Global Values

*Global values* are visible from within any formula. Use global values to store information that does not change often, but you refer to frequently, such as company name, or a location allowance that applies to many employees. Global values are datetracked so you can make date effective changes ahead of time.

You can never change a global value using a formula. You alter global values using the Globals window. The global value is the same across all formulas within a Business Group.

See: Defining Global Values, page 1-153

Below is an example using a global value.

```
/* Formula: HAZARD ALLOWANCE FORMULA */

IF basic_hours > hazard_limit

THEN

     hazard_allowance = 2.30

ELSE

     hazard_allowance = 2.00

RETURN hazard_allowance
```

In this example, hazard_limit is a global value, which has been preset to reflect the point at which workers' hazard payment increases.

## Database Items

*Database items* exist in the application database and have a label, hidden from users, that the system uses to find the data.

There are two types of database item:

- *Static* database items are predefined. They include standard types of information, such as the sex, birth date, and work location of an employee, or the start and end dates of a payroll period.

- *Dynamic* database items are generated from your definitions of:
  - elements
  - balances
  - absence types
  - grade rates and pay scale rates
  - flexfield segments

The name of your element, pay and input values, balance, absence type, grade rate, pay scale rate, or flexfield segment becomes part of the name of the generated database item. This helps you identify the database item you need when you display a list of all available items in the Formulas window.

Definitions of flexfield segments only become database items when you initiate the Declare Descriptive Flexfield process from the Submit Requests window. The other definitions become database items immediately when you save them to the database.

See:

Static Database Items, page 1-63

Dynamic Database Items, page 1-87.

Database items are specific to the *context* in which you use them. For example, using the database item hourly_rate gives the appropriate hourly rate for the specific assignment being processed.

Like global values, database item values cannot be changed within a formula.

## Rules for Determining Variable Class and Data Type

The rules that determine the data type and class of variables in a formula are:

1. The variable can be an input you name in the Inputs statement. For example:

```
INPUTS ARE  salary_amount,

  start_date (date)

  frequency (text)
```

   If you do not specify the variable type in the statement, Oracle FastFormula assumes it is numeric.

2. If the variable is not an input, Oracle FastFormula looks in the list of global values the first time the variable occurs. If the variable is in the list, Oracle FastFormula determines the data type from there.

3. If the variable is not in the list, Oracle FastFormula searches the list of database items. Again, if it is in the list, Oracle FastFormula knows the data type.

4. Finally, if Oracle FastFormula does not find the variable in either the global values or the database items, then it treats the variable as a local variable. It determines the data type from the way you use the variable.

Notice that if the variable is either a global value or a database item, then any attempt in your formula to alter the value of the variable causes an error.

If the variable is a local variable, it does not contain a value when it is first used in the formula. Therefore you must assign a value to the variable before you try to use it in a condition or expression. If you fail to assign a value, Oracle FastFormula fails when you attempt to verify or run the formula.

> **Tip:** To avoid a failure, assign values to your local variables when they first appear in your formula.

## Naming Variables

Variables have names comprising one or more words. The words must be joined by underscores. The words must each start with an alphabetic character (A-Z) and can be followed by alphanumeric characters (A-Z, 0-9). The maximum size of a variable name is 80 characters.

Oracle FastFormula is not sensitive to case. So, for example, the variable called EMPLOYEE_NAME is the same as the variable Employee_name.

The following reserved words cannot be used as the names of variables:

| | | | |
|---|---|---|---|
| ALIAS | AND | ARE | AS |
| DEFAULT | DEFAULTED | ELSE | EXECUTE |
| FOR | IF | INPUTS | IS |
| NOT | OR | RETURN | THEN |
| USING | WAS | | |

Also, any word consisting only of digits, as these could be mistaken for numbers.

You may find that the name of a database item or global value is too long to use conveniently in your formula. You can set up an alternative, shorter name for use within a formula. You set this up using the Alias statement.

See: Statements, page 1-53

# Expressions

Expressions combine constants and variables with arithmetic operators (+, -, *, /) and functions to return a value of a certain data type. For example, the expression (3 + 2) returns a value of 5, and is of numeric data type.

The format of an expression is:

```
SUBEXPRESSION [operator SUBEXPRESSION ...]
```

This means that a number of 'subexpressions' can combine in a single expression. For example, the subexpressions (3 + 2) and MONTHS_BETWEEN(start_date, end_date) can combine in a single expression as follows:

```
(3 + 2) + MONTHS_BETWEEN(start_date, end_date)
```

Expressions can also be used inside functions, as in the following example:

```
salary = GREATEST(minimum_wage, (hourly_rate * hours_worked))
```

## Data Type of Expressions

The rules for determining the data type of an expression are simple. Operands in an expression are normally of the same data type, and this is normally the data type of the expression as a whole. For example, in the following expression all the operands are numeric and the expression itself is numeric:

```
GREATEST(minimum_wage, (hourly_rate * hours_worked))
```

There are some exceptions to this. For example:

```
DAYS_BETWEEN(date1, date2)

MONTHS_BETWEEN(date1, date2)
```

These have date operands, but return a numeric value.

So the expression:

```
4 + days_between(start_date, todays_date)
```

returns a numeric result.

# Arithmetic Operators

An expression may contain arithmetic operators, which determine how variables and constants are manipulated. For example, the operator "+" indicates that two items should be added together.

The division, subtraction, and multiplication operators can only be used with numeric operands. The operands can be variables, constants, or subexpressions. A formula error occurs if:

- the result of subtraction is too large a negative number
- the result of multiplication is too large
- the second operand of a division evaluates to zero

In both cases, 'too large' here is determined by the normal limits in the ORACLE database.

The addition operator can be used with numeric or text operands. If the result is greater than 240 characters, a formula error occurs.

Notice that you enclose text constants in single forward quotes only ('), not double quotes ("). For example, the statements:

```
string1 = 'will '

result_string = 'Pigs ' + string1 + 'fly'
```

set the local variable `result_string` to 'Pigs will fly'.

## Functions

Oracle FastFormula provides functions that manipulate data in different ways. Some functions work on only one type of data, some can work on two, others work on all three data types.

The functions are described below, separated into the three data types and functions that convert between data types. Where a function returns a different data type result than the data type of its operands, the description explains this.

Some functions retrieve data from Oracle Applications tables. These are described in the sections: Functions To Get Values from Tables, and Functions for Accrual Type Formulas. Some functions enable you to set and use globals in SQL*Plus. They are described in the section: Functions to Set and Get Globals.

The general form of a function is:

```
NAME OF FUNCTION(operand, operand, . .)
```

Notice that, as with the operators, the operands of a function can be variables, constants, or complete expressions. If the operand is a text string, you must enclose it in quote marks.

## Text Functions

### CHR

CHR(n)

The CHR function returns the character having the binary equivalent to number operand *n* in the database character set.

Example

```
/* CHR (10) used to add a newline to the end of REPORT_TEXT2. */

 REPORT_TEXT2 = 'Warning the Transaction Limit has been exceeded'
```

### DEBUG

DEBUG(*expr*)

This function accepts a string and uses a DBMS_OUTPUT statement to output the string to the console. Use this function when you are testing a new formula to track its processing and identify where it is failing.

### GREATEST

GREATEST(*expr*, *expr* [, *expr*] . . .)

GREATEST_OF(*expr*, *expr* [, *expr*] . . .)

The GREATEST function compares the values of all the text string operands. It returns the value of the operand that is alphabetically last. If there are two or more operands that meet the criteria, Oracle FastFormula returns the first.

### INITCAP

INITCAP(*expr*)

The INITCAP function returns the expression *expr* with the first letter of each word in uppercase, all other letters in lowercase. Words are delimited by white space or characters that are not alphanumeric.

### INSTR

INSTR(*expr1*,*expr2*[,n[,m]])

The INSTR searches *expr1* beginning with its *n*th character for the *n*th occurrence of *expr2* and returns the position of the character in *expr1* that is the first character of this occurrence. If *n* is negative, Oracle FastFormula counts and searches backward from the end of *expr1*. The value of *m* must be positive. The default values of both *n* and *m* are 1, meaning Oracle FastFormula begins searching at the first character of *expr1* for the first occurrence of *expr2*. The return value is relative to the beginning of *expr1*, regardless of

the value of *n*, and is expressed in characters. If the search is unsuccessful (if *expr2* does not appear *m* times after the *n*th character of *expr1*) the return value is 0.

### INSTRB

INSTRB(*expr1*,*expr2*[,n[,m]])

The same as INSTR, except that *n* and the return value are expressed in bytes, rather than in characters. For a single-byte database character set, INSTRB is equivalent to INSTR.

### LEAST

LEAST(*expr*, *expr* [, *expr*] . . .)

LEAST_OF(*expr*, *expr* [, *expr*] . . .)

The LEAST function compares the values of all the text string operands. It returns the value of the operand that is alphabetically first. If there are two or more operands that meet the criteria, Oracle FastFormula returns the first.

### LENGTH

LENGTH(*expr*)

The LENGTH function returns the number of characters in the text string operand *expr*.

**Note:** The data type of the result of this function is numeric.

### LENGTHB

LENGTHB(*char*)

The LENGTHB function returns the length of *char* in characters. If *char* has datatype CHAR, the length includes all trailing blanks. If *char* is null, this function returns null.

### LOWER

LOWER(*expr*)

The LOWER function returns the string operand *expr* with all letters lowercase. The return value has the same datatype as the argument *expr*.

### LPAD

(*expr*, *n* [,*pad*])

The LPAD function returns the text string operand *expr* left-padded to length *n* with the sequence of characters in *pad*. The default for *pad* is a blank. If *expr* is longer than *n*, then LPADreturns the portion of *expr* that fits in *n*.

Examples:

```
/* A is set to 'XYXYXhello' */

A = LPAD ('hello, 10, 'XY')

/* A is set to 'hell' */

A = LPAD ('hello', 4 )
```

### LTRIM

(*expr* [,*set*])

The LTRIM function returns the text string operand *expr* with all the leftmost characters that appear in *set* removed. The default for *set* is a blank. If none of the leftmost characters of *expr* appear in *set* then *expr* is returned

Examples:

```
/* A is set to 'def' */

A = LTRIM ('abcdef','abc')

/* A is set to 'abcdef' */

A = LTRIM ('abcdef','bc')
```

### REPLACE

(*expr, search_string* [,*replacement_string*])

The REPLACE function returns the text string operand *expr* with every occurrence of *search_string* replaced with *replacement_string*. If *replacement_string* is omitted or null, all occurrences of *search_string* are removed. If *search_string* is NULL, *expr* is returned. REPLACE allows you to substitute one string for another as well as to remove character strings.

Example:

```
SELECT REPLACE ('JACK and JUE','J','BL') "Changes"

FROM DUAL

Changes

-------

BLACK and BLUE
```

### RPAD

(*expr, n* [,*pad*])

The RPAD function returns the text string operand *expr* right-padded to length *n* with the sequence of characters in *pad*. The default for *pad* is a blank. If *expr* is longer than *n,* then RPADreturns the portion of *expr* that fits in *n.*

Examples:

```
/* A is set to 'helloXYXYX' */

A = RPAD ('hello, 10, 'XY')

/* A is set to 'hell' */

A = RPAD ('hello', 4 )
```

### RTRIM

(*expr* [,*set*])

The RTRIM function returns the text string operand *expr* with all the rightmost characters that appear in *set* removed. The default for *set* is a blank. If none of the rightmost characters of *expr* appear in *set* then *expr* is returned

Examples:

```
/* A is set to 'abc' */

A = RTRIM ('abcdef','def')

/* A is set to 'abcdef' */

A = RTRIM ('abcdef','de')
```

### SUBSTRING

SUBSTR(*expr, m* [,*n*])

SUBSTRING(*expr, m* [,*n*])

The SUBSTRING function returns a substring of the text string operand *expr* of length *n* characters beginning at the *m*th character. If you omit the third operand, the substring starts from *m* and finishes at the end of *expr*.

> **Note:** The first operand is a text operand. The second and third operands are numeric operands. The resulting data type of this function is text.

> **Tip:** Always check string length before you start to substring. For example:

```
/* Check that the tax code starts with GG */

IF length(Tax_code) <= 2

THEN

 (message = 'Tax code is too short'

 RETURN message

 )

IF substr( Tax_code, 1, 2) = 'GG' THEN ...
```

Or, to check if Tax_code is a string of at least two characters starting with 'GG':

```
IF Tax_code LIKE 'GG%' THEN ...
```

### SUBSTRB

(*expr*, m [,n])

The same as SUBSTR, except that the arguments *m* and *n* are expressed in bytes, rather than in characters. For a single-byte database character set, SUBSTRB is equivalent to SUBSTR.

### TRANSLATE

(*expr*, from, to)

The TRANSLATE function returns the text string operand *expr* with all occurrences of each character in *from* replaced by its corresponding character in *to*. Characters in *expr* that are not in *from* are not replaced. The argument *from* can contain more characters than *to*. In this case, the extra characters at the end of *from* have no corresponding characters in *to*. If these extra characters appear in *expr*, they are removed from the return value. Oracle FastFormula interprets the empty string as null, and if this function has a null argument, it returns null.

### TRIM

TRIM(*trim_character* FROM *trim_source*)

The TRIM function allows you to trim heading or trailing characters (or both) from a character string. If *trim_character* or *trim_source* is a character literal, you must enclose it in single quotes. You can specify LEADING or TRAILING to remove leading or trailing characters. If you specify none of these, both leading and trailing characters are removed equal to *trim_character*.

### UPPER

UPPER(*expr*)

The UPPER function converts a text string to upper case.

## Numeric Functions

### ABS

ABS(*n*)

The ABS function returns the magnitude of a numeric operand *n* as a positive numeric value.

If the value of the operand is positive, its value returns unchanged. If the operand is negative then the value's sign inverts, and the value returns as a positive number.

Example:

ABS (-17) returns 17

### CALCULATE_HOURS_WORKED

CALCULATE_HOURS_WORKED(*n, date1, date2, standard_frequency*)

The CALCULATE_HOURS_WORKED function returns the total number of hours worked in a given date range.

The function works by calculating the total number of hours worked for an employee between *date1* and *date2*, taking into account that the employee works *n* hours in the standard working period *standard_frequency*. This parameter gives the unit of measurement for the standard working period. It can be one of:

- W (weekly)
- M (monthly)
- Y (yearly)

Example:

```
CALCULATE_HOURS_WORKED (40, 01-FEB-2003, 28-FEB-2003, W) returns 160
```

This indicates that the employee has worked 160 hours in the month of February 2003, based on a 40-hour week and taking into account the number of working days in that month.

### FLOOR

FLOOR(*n*)

The FLOOR function returns the integer part of a numeric operand *n*.

If the value of the operand contains information after the decimal point, Oracle FastFormula discards that information and returns a whole number.

Example:

FLOOR(35.455) returns 35

### GREATEST

GREATEST(*n*, *n* [, *n*] . . .)

GREATEST_OF(*n*, *n* [, *n*] . . .)

The GREATEST function compares all the operands and returns the largest value.

### LEAST

LEAST(*n*, *n* [, *n*] . . .)

LEAST_OF(*n*, *n* [, *n*] . . .)

The LEAST function compares all the operands and returns the smallest value.

### POWER

POWER(*m*,*n*)

Returns *m* raised to the *n*th power. The base *m* and the exponent *n* can be any numbers, but if *m* is negative, *n* must be an integer.

### ROUND

ROUND(*n* [, *m*])

The ROUND function rounds off a numeric value *n* to *m* decimal places and a date depending on the format of *m*. For numeric values, the first operand is the value Oracle FastFormula rounds off, the second the number of places Oracle FastFormula rounds off to. For dates, ROUND returns *n* rounded to the unit specified by the format model of *m* such as Year or Day. Refer to the *SQL Language Reference Manual* for details of the valid formats you can specify.

Examples:

ROUND(2.3401, 2) returns 2.34

ROUND (2.3461, 2) returns 2.35

ROUND (TO_DATE('27-OCT-1992', 'DD-MON-YYYY'), 'YEAR') returns 01-JAN-1993

### ROUNDUP

ROUNDUP(*n* [, *m*])

ROUND_UP(*n* [, *m*])

The ROUNDUP function rounds a numeric value *n* up to *m* decimal places. The first operand is the value to be rounded up, the second the number of places to round to. If the digits after the rounding point are zero, the value is unchanged. If the digits are not zero, the value is incremented at the rounding point.

Examples:

ROUND_UP(2.3401, 2) returns 2.35

ROUND_UP(2.3400, 2) returns 2.34.

### TRUNC

TRUNC(*n* [, *m*])

TRUNCATE(*n* [, *m*])

The TRUNC function rounds a numeric value *n* down to *m* decimal places. The first operand is the value to be rounded down, the second the number of places to round to. TRUNC also returns n with the time portion of the day truncated to the unit specified by the format model of m. If you omit m, d is truncated to the nearest day The default model, 'DD', returns the date rounded or truncated to the day with a time of midnight.

Oracle FastFormula drops all digits (if any) after the specified truncation point.

Examples:

TRUNC(2.3401, 2) returns 2.34.

TRUNC(TO_DATE('27-OCT-1992', 'DD-MON-YYYY'), 'YEAR') returns 01-JAN-1992

## Date Functions

### ADD_DAYS

ADD_DAYS(*date*, *n*)

The ADD_DAYS function adds a number of days to a date. The resulting date accords with the calendar.

**Note:** Oracle FastFormula ignores any fractional part of the number *n*.

Example:

ADD_DAYS ('30-DEC-1990' (date), 6) returns 5 JAN 1991

### ADD_MONTHS

ADD_MONTHS(*date*, *n*)

The ADD_MONTHS function adds a number of months to a date. The resulting date accords with the calendar.

**Note:** Oracle FastFormula ignores any fractional part of the number *n*.

### ADD_YEARS

ADD_YEARS(*date*, *n*)

The ADD_YEARS function adds a number of years to a date. The resulting date accords with the calendar.

**Note:** Oracle FastFormula ignores any fractional part of the number *n*.

### GREATEST

GREATEST(*date1*, *date2*[, *date3]* . . .)

The GREATEST function compares all the operands and returns the latest date.

### LAST_DAY

LAST_DAY(*d*)

The LAST_DAY function returns the date of the last day of the month that contains *d*. You might use this function to determine how many days are left in the current month.

### LEAST

LEAST(*date1*, *date2* [, *date3*] . . .)

The LEAST function compares all the operands and returns the earliest date.

### DAYS_BETWEEN

DAYS_BETWEEN(*date1*, *date2*)

The DAYS_BETWEEN function returns the number of days between two dates. If the later date is first, the result is a positive number. If the earlier date is first, the result is a negative number. The number returned is also based on the real calendar.

**Note:** The result is a numeric data type.

Example:

DAYS_BETWEEN('1995/06/27 00:00:00' (date), '1995/07/03 00:00:00' (date)) returns -5

### MONTHS_BETWEEN

MONTHS_BETWEEN(*date1*, *date2*)

The MONTHS_BETWEEN function returns the number of months between two dates.

If the later date is first, the result is a positive number. If the earlier date is first, the result is a negative number. The number returned is also based on the real calendar.

If the result is not a whole number of months (that is, there are some days as well), the days part is shown as a decimal.

**Note:** The result is a numeric data type.

### NEW_TIME

NEW_TIME(*d*, *zl*, *z2*)

Returns the date and time in zone *z2* when the date and time in zone *z1* are *d*. The arguments *z1* and *z2* can be any one of these text strings:

| | |
|---|---|
| AST or ADT | Atlantic Standard or Daylight Time |
| BST or BDT | Bering Standard or Daylight Time |
| CST or CDT | Central Standard or Daylight Time |
| EST or EDT | Eastern Standard or Daylght Time |
| GMT | Grenwich Mean Time |
| HST or HDT | Alaska-Hawaii Standard Time or Daylight Time |
| MST or MDT | Mountain Standard or Daylight Time |
| NST | Newfoundland Standard Time |
| PST or PDT | Pacific Standard or Daylight Time |
| YST or YDT | Yukon Standard or Daylight Time |

### NEXT_DAY

NEXT_DAY(*d*, *expr*)

The NEXT_DAY function returns the date of the first weekday named by *expr* that is later than the date *d*. The argument *expr* must be a day of the week in your session's date language. The return value has the same hours, minutes, and seconds component as the argument *d*.

## Data Conversion Functions

Use data conversion functions to convert from one data type to another data type. For example, you could have an expression returning a number value for salary, which you want to include in a printed message (that is, a character value). To print the number as part of the message, you need to convert the value of salary from a number to a character value, using the TO_TEXT function.

### CONVERT

(*expr*, *dest_char_set* [,*source_char_set*])

The CONVERT function converts a character string from one character set to another. The *expr* argument is the value to be converted. The *dest_char_set* argument is the name of the character set to which *expr* is converted. The *source_char_set* argument is the name of the character set in which *expr* is stored in the database. The default value is the database character set.

### INSTR

(*expr1*,*expr2*[,*n*[,*m*]])

The INSTR function searches *expr1* beginning with its *n*th character for the *m*th occurrence of *expr2* and returns the position of the character in *expr1* that is the first character of this occurrence. If *n* is negative, Oracle FastFormula counts and searches backwards.

### NUM_TO_CHAR

NUM_TO_CHAR(*n, format*)

Converts the number *n* from number data type to text data type using the specified format. This function is equivalent to the SQL TO_CHAR function. For example:

```
NUM_TO_CHAR(amount, '$9,990.99')
```

This returns the amount with a leading dollar sign, commas every three digits, and two decimal places. Refer to the *SQL Language Reference Manual* for a full list of the valid number formats you can specify.

### TO_DATE

TO_DATE (*expr* [, *format*])

Converts the expression *expr* of text data type to a date data type. The text expression must be of the form 'YYYY/MM/DD HH24:MI:SS' if no format is provided. The day and year must be in numeric form. For example:

```
 /* legal */
```

```
date_1 = TO_DATE ('12 January 89', 'DD Month YY')

/* illegal */

date_1 = TO_DATE ('12 January Nineteen-Eighty-Nine',

     'DD Month Year')
```

**Note:** When assigning date variables from constants it is much more efficient to say:

```
date_1 = '1989/01/12 00:00:00'(date)
```

**Note:** The text expression must be in the format of either YYYY/MM/DD HH24:MI:SS or DD-MON-YYYY if no format is provided.

### TO_NUMBER

TO_NUM(*expr*)

TO_NUMBER(*expr*)

Converts the expression *expr* of text data type to a number data type. The expression must represent a valid number. So for example, you cannot convert an expression such as `Type 24' but you can convert the text expression `1234'. For decimal values, you must always use a period as a decinal point, for example '4.5'.

### TO_TEXT

TO_TEXT(*n)* TO_TEXT (*date1 [, format]*)

TO_CHAR(*n*) TO_CHAR(*date1 [, format]*)

DATE_TO_TEXT(*n*) (*date1 [, format]*)

The TO_TEXT function converts:

- the number *n* from number data type to text data type. The default number format has the decinal point as a period, for example '4.5'.
- the date *date1* from date data type to text data type. The optional *format* should be a text string like 'DD/MM/YYYY'. The default format is 'YYYY/MM/DD HH24:MI:SS'.

For example:

```
birthdate = '21-JAN-1960' (date)

     mesg = 'Birthdate is: ' + TO_CHAR  (birthdate)
```

```
/* sets mesg to 'Birthdate is: 1960/01/21 00:00:00' */

     mesg = 'Birthdate is: ' + TO_CHAR (birthdate,

            'DD-MON-YY')

/* sets mesg to 'Birthdate is: 21-JAN-60' */

     mesg = 'Birthdate is: ' + TO_CHAR (birthdate,

            'DD Month Year')

/* sets mesg to 'Birthdate is: 21 January Nineteen-Sixty' */
```

## Functions to Get Values From Tables

### GET_LOOKUP_ MEANING

GET_LOOKUP_MEANING(*lookup_type , lookup_code)*

The GET_LOOKUP_MEANING function enables Oracle FastFormula to translate a lookup code into a meaning. This can be used for any descriptive flexfield items or developer flexfield items that are based on lookups.

Example:

```
GET_LOOKUP_MEANING ('ETH_TYPE', PEOPLE_GB_ETHNIC_ORIGIN)
```

### GET_TABLE_VALUE

GET_TABLE_VALUE(*table_name, column_name, row_value* [,*effective date*])

The GET_TABLE_VALUE function returns the value of a cell in a user-defined table. The three text operands, which identify the cell (table_name, column_name, and row_value), are mandatory. The date operand is optional. If it is not supplied, the function returns the cell value as of the effective date.

You cannot use this function in formulas for user table validation or QuickPaint reports.

Example:

```
GET_TABLE_VALUE('WAGE RATES', 'Wage Rate', Rate_Code)
```

### RAISE_ERROR

RAISE_ERROR(*application_ID, message name*)

This function allows you to raise a functional error message from within a formula. It accepts an Application ID and the message_name of an Oracle Applications error message to raise.

Example:

```
ERROR = RAISE_ERROR(800, 'error_name')
```

### RATES_HISTORY

RATES_HISTORY(*element or rate type name, date, element or rate type indicator, time dimension*)

This function uses information stored in the UK Element Attribution Information EIT and information about the assignment's contract type to calculate a payment rate as of the given date and expressed for the selected time dimension (such as hourly or annual). If the element or rate type indicator is R, the function sums the rates for all elements classified with the given rate type (which is stored against the element in the Rate Type Information EIT).

The time dimension parameter must be A (annual), D (daily), H (hourly), or P (periodic). The element or rate type parameter must be R (rate type) or E (element).

The function can also adjust the returned rate for FTE and length of service, if these factors are set to Yes in the Element Attribution Information.

## Functions for Accrual Type Formulas

In addition to the standard FastFormula functions, you may find the following functions useful for your Accrual and Carry Over formulas.

### CALCULATE_PAYROLL_PERIODS

This function takes no parameters; it uses the payroll id context. It calculates the number of payroll periods in one year for that payroll, and sets the global variable PAYROLL_YEAR_NUMBER_OF_PERIODS to that value. For example, the function would set the global variable to 12 for a calendar month payroll.

Example:

```
E = CALCULATE_PAYROLL_PERIODS
```

### GET_ABSENCE

GET_ABSENCE(*calculation date, start date*)

This function calculates the total amount of absence contributing to an accrual plan between two dates. It counts the whole of any absence that:

- has a start date and an end date, and
- starts on or between the two dates given as inputs

Example:

```
TOTAL_ABSENCE = GET_ABSENCE('01-JAN-1999'(date), '31-DEC-1999'(date))
```

### GET_CARRY_OVER

GET_CARRY_OVER(*calculation date, start date*)

This function returns the number of days or hours recorded on the Carry Over element entry with an effective date on or between the two input dates. If more than one element entry is effective between these dates, the function sums the hours or days.

Carry Over element entries may also have an expiry date, after which any unused carry over is lost. If the calculation date is after the expiry date, the function checks the absences taken between start and calculation date. If the person took absences totaling the carry over, the function returns total carry over because it was all used before it expired. If absences total less than the carry over, the function returns total absence time; the rest of the carryover is lost.

For example, if the carry over is 10 days and 6 days absence were taken up to the expiry date, the function returns 6. The other four days of carry over have expired and been lost.

### GET_NET_ACCRUAL

GET_NET_ACCRUAL(*calculation date, plan id, accrual start date, accrual latest balance*)

This function calls the accrual formula defined in the accrual plan to return the net accrual at the calculation date. The following contexts must be available to a formula calling this function: assignment id, payroll id, business group id, and assignment action id.

### GET_OTHER_NET_CONTRIBUTION

GET_OTHER_NET_CONTRIBUTION(*calculation date, start date*)

This function calculates the total amount of net contribution other than absences or carry over between two dates. It looks for element entries for all elements that have been added in the Net Calculation Rules window. It sums the days or hours from all entries with an effective date on or between the two input dates.

### GET_PAYROLL_PERIOD

GET_PAYROLL_PERIOD(*date*)

This function determines the payroll period spanning the input date and sets global variables containing the start and end date and the period number. It returns 0 if successful, and otherwise error.

This example shows how to use this function then use the GET_DATE and GET_NUMBER functions to retrieve the values it sets in the global variables:

```
E = GET_PAYROLL_PERIOD(Calculation_date)

Calculation_Period_SD = GET_DATE('PAYROLL_PERIOD_START_DATE')

Calculation_Period_ED = GET_DATE('PAYROLL_PERIOD_END_DATE')

Calculation_Period_PNUM = GET_NUMBER('PAYROLL_PERIOD_NUMBER')
```

### GET_ACCRUAL_BAND

GET_ACCRUAL_BAND(*number*)

This function determines the appropriate accrual band for the specified length of service. It sets global variables containing the ANNUAL_RATE, UPPER_LIMIT and CEILING values for the band. ANNUAL_RATE is the amount that should accrue this accrual term. UPPER_LIMIT is the length of service that must be completed for the employee to go to the next accrual band. CEILING is the maximum number of hours or days the employee can accrue at any time. The function returns 0 if successful, and otherwise error.

This example shows how to use this function then use the GET_NUMBER function to retrieve the values it sets in the global variables:

```
IF (GET_ACCRUAL_BAND(Years_Service) = 0 THEN

(

Annual_Rate = GET_NUMBER('ANNUAL_RATE')

Upper_Limit = GET_NUMBER('UPPER_LIMIT')

Ceiling = GET_NUMBER('CEILING')

ELSE

( ... [processing for error] ....)

)
```

### GET_ASSIGNMENT_STATUS

GET_ASSIGNMENT_STATUS(*date*)

This function determine the assignment status at a given date. It populates the globals ASSIGNMENT_EFFECTIVE_SD, ASSIGNMENT_EFFECTIVE_ED and ASSIGNMENT_SYSTEM_STATUS. It returns 0 if successful, and otherwise error.

Example:

```
ERROR = GET_ASSIGNMENT_STATUS('01-JAN-1999'(date))
```

### GET_ASG_INACTIVE_DAYS

GET_ASG_INACTIVE_DAYS*(period start date, period end date)*

This function checks the assignment status on each day from period start date to period

end date. It calls the function GET_WORKING_DAYS to calculate the total number of working days in the period (Mondays to Fridays) and subtracts the number of working days when the assignment was inactive. It returns the number of inactive working days.

### GET_PERIOD_DATES

GET_PERIOD_DATES(*date1, unit, date2, number*)

This function determines the start and end dates of a period of time with the duration specified by the unit input and the number (such as 2 months). Valid units are D (days), M, (months), and W (weeks). The period spans date1 and starts on a date that is a multiple of the unit duration on from date2 (or backwards from date2).

The function populates the globals PERIOD_START_DATE and PERIOD_END_DATE. It returns 0 if successful, and otherwise error.

Example:

```
Error = GET_PERIOD_DATES('1-FEB-1999'(date), 'M', '15-DEC-1998'(date),
1)
```

This example populates PERIOD_START_DATE with 15-JAN-1999 and PERIOD_END_DATE with 14-FEB-1999.

An example where the period starts before date2:

```
Error = GET_PERIOD_DATES('1-FEB-1999'(date), 'M', '15-APR-1999'(date),
2)
```

This example populates PERIOD_START_DATE with 15-JAN-1999 and PERIOD_END_DATE with 14-MAR-1999.

### GET_START_DATE

GET_START_DATE(*accrual start date, start of accrual term*)

This function returns the date at which the accrual formula should start calculating accruals.

- If there is no payroll balance holding gross accruals, the date is always the start of the accrual term.
- If there is a payroll balance and there are retrospective absence entries that have not already been used in an accrual calculation, the function returns the start date of the earliest of these entries.
- If there is a payroll balance and there are no unprocessed retrospective absence entries, the function returns accrual start date.

  This date, which is passed into the accrual formula, is the day after either the Date Earned or the Date Paid of the last payroll period in which the assignment was processed--depending on the PTO Balance Type set for the business group.

**Note:** Although GET_START_DATE returns the start date of the earliest of any unprocessed retrospective element entries, this date is not currently used in the seeded accrual formulas. If GET_START_DATE finds any unprocessed retrospective element entries, the formula always calculates accruals from the beginning of the accrual term.

#### GET_WORKING_DAYS

GET_WORKING_DAYS(*start date, end date)*

This function returns the number of working days (Mondays to Fridays) in the period from start date to end date.

#### PUT_MESSAGE

PUT_MESSAGE(*expr*)

This function adds a message to the stack to be output at the end of the formula by the Accruals form.

Example:

```
E = PUT_MESSAGE('The assignment is not yet eligible for accrual')
```

### Functions to Call a Formula

These functions allow you to call another formula, either once or in a loop. They require all the contexts available to the Accruals formula type.

#### CALL_FORMULA

CALL_FORMULA(*formula name*)

This function runs a named formula with no inputs and no outputs.

#### LOOP_CONTROL

LOOP_CONTROL(*formula name*)

This function repeatedly calls another formula, which must have the return parameter 'CONTINUE_LOOP'. The loop continues until the function detects a value other than 'Y' in CONTINUE_LOOP. If it detects 'N', the function returns 0 (success); if it detects another value, the function returns 1 (error).

### Functions to Set and Get Globals

Using the following functions, you can set and use globals in SQL*Plus from within

your formulas.

### SET_TEXT, SET_NUMBER, SET_DATE

SET_TEXT(*variable name, value*)

SET_NUMBER(*variable name, value*)

SET_DATE(*variable name, value*)

These functions accept the name of a global variable and the value to be set. They determine whether the global exists and, if not, create a new global. They return 0 if successful and 1 if not successful.

Examples:

```
E = SET_NUMBER('UPPER_LIMT', 0)

E = SET_DATE('CONTINUOUS_SERVICE_DATE', service_start_date)
```

### GET_TEXT, GET_NUMBER, GET_DATE

GET_TEXT(*variable name)*

GET_NUMBER(*variable name*)

GET_DATE(*variable name*)

These functions accept the name of a global variable and return its value. If they cannot find the global, they return NULL.

Example:

```
Calculation_Period_SD = GET_DATE('PAYROLL_PERIOD_START_DATE')
```

### CLEAR_GLOBALS

This function sets to NULL the value of all global variables that were set using SET_TEXT, SET_NUMBER, and SET_DATE. There are no inputs. It returns 0 if successful and 1 if not successful.

### REMOVE_GLOBALS

This function removes all global variables. There are no inputs. It returns 0 if successful and 1 if not successful.

### ISNULL

ISNULL(*variable name*)

A set of three functions that test whether a text, numeric, or date value is NULL. Returns Y if the value is NULL and N otherwise.

Example:

```
IF IS_NULL(VARIABLE_NAME) = 'Y' THEN
ERROR = SET_NUMBER(VARIABLE_NAME, 0)
```

### Rate By Criteria Function

Use the Rate By Criteria (RBC) function within the Payroll formula attached to the element to return the RBC rate for which the employee is eligible. The element must be associated with a criteria rate definition.

**RBC_Rate_Retrieval**

The function evaluates the eligible rate for an employee by processing the rate matrixes in the employee's business group.

To call the seeded RBC_Rate_Retrieval function from a formula, declare a local variable within the Element formula text:

<*local variable1*> = RBC_Rate_Retrieval()

The function includes predefined contexts, so you include no additional parameters.

## Comments

A formula may contain any number of comments, which can be placed anywhere in the formula.

Comments start with the sequence /* (slash asterisk), and finish with */ (asterisk slash). Oracle FastFormula ignores all text within these comment delimiters.

> **Caution:** Do not put a comment within a comment. This causes Oracle FastFormula to return a syntax error.

## Statements

Statements are instructions that Oracle FastFormula carries out. There are six types of statement you can use:

- Alias statement
- Assignment statement
- Default statement
- If statement
- Inputs statement

- Return statement

An If statement can have Assignment, Return, and other If statements nested within it, enabling you to build up powerful formulas.

## Order of Statements

1. Alias statements (if any)
2. Default statements (if any)
3. Input statement (if any)
4. Other statements

## Alias Statement

The format of the Alias statement is:

```
ALIAS varname1 AS varname2
```

where varname1 is the name of an existing database item or global value, and varname2 is a unique name not currently known to the system nor used previously in your formula.

Use the Alias statement to define another name, or alias, for existing variables in the system. You can declare aliases for database items and global values.

Alias statements must appear before any other statements in a formula.

## Default Statement

The format of the Default statement is:

```
DEFAULT FOR <varname> IS <constant>
```

where varname is an input value or database item, and constant is a constant value matching the data type of varname.

Use the Default statement to set a default value for an input value or database item. The formula uses the default value if the database item is empty or the input value is not provided when you run the formula.

You can use the Default statement with the 'WAS DEFAULTED' test to detect when a default value has been used. For example:

```
DEFAULT FOR hourly_rate IS 3.00

X = hours_worked * hourly_rate

IF hourly_rate WAS DEFAULTED
```

```
THEN

  MESG = 'Warning: hourly rate defaulted'
```

This example sets a default of 3.00 for the database item hourly_rate. If hourly_rate is empty (NULL) in the database, the formula uses the default value of 3.00 and issues a warning message.

> **Important:** You must use the Default statement for database items that can be empty. The Database Items window includes a check box labelled Default Required. This check box is checked for database items that can be empty. The Database Items window appears when you click the Show Items button on the Formulas window.

## Inputs Statement

The format of the Inputs statement is:

```
INPUTS ARE varname1(data type)[, varname2 (data type)] ...
```

Use the Inputs statement to pass input values from an element into a formula.

For example,

```
INPUTS ARE bonus (number),

           start_date (date)
```

You do not need to declare the type of number variables because this is the default data type. You can define up to 15 input values for an element.

TheInputs statement must appear before the other formula statements except:

- any Alias statements, which must always be at the top of the formula
- any Default statements that provide default values for input values

### Input Variables or Database Items

Always use the Inputs statement to retrieve the input values of the element associated with the formula. Using a database item forces the formula to execute the code and work out the path to retrieve the database item.

For example, the formula below:

```
INPUTS ARE wage_rate,

           hours_worked

wage = wage_rate * hours_worked
```

```
RETURN wage
```

is more efficient than the second formula:

```
wage = wage_wage_rate * wage_hours_worked

RETURN wage
```

## Assignment Statement

Use the Assignment statement to set a value for a local variable. The format of the Assignment statement is:

```
varname = expression
```

For example:

```
rate = hourly_rate + 14

wage = hours_worked * rate
```

Oracle FastFormula evaluates the expression on the right hand side of the statement, and places its result in the variable you name on the left hand side. The left side of an Assignment statement must always be a local variable because a formula can only change the value of local variables.

## IF Statement

Use the If statement to check a condition that controls whether a sequence of statements is executed.

There are two *clauses* in the If statement: the THEN and ELSE clauses.

- The THEN clause lets you define what to do if the conditions are true.
- The ELSE clause lets you define what to do if the conditions are not true.

The If statement is the only statement that can have other statements *nested* within it, including other IF statements.

### Format of Statement

The format of the If statement is:

```
IF [NOT] condition

[logical operator] [NOT] condition

THEN
```

```
    statement [statement ..]

ELSE

    statement [statement ..]
```

The If statement can consist of a single condition, or two or more conditions combined with logical operators. The logical operators are AND, OR and NOT. The first two combine the conditions logically, and the third negates a condition:

- The AND operator means that if both conditions are true, then their combination is true.
- The OR operator means that if either condition is true, then their combination is true.
- If the NOT operator precedes a condition, this inverts the truth of the condition. That is, if condition X is true, then NOT X is false.

### Format of Conditions

A condition itself has a valid format. This is:

```
expression comparator expression
```

The values represented by each expression are compared together in the way described by the comparator. The two expressions must both return the same data type. There are eight comparators, and the way they work depends upon the data type of the values they are comparing.

| Comparator | Symbols | Data Types | Meaning |
|---|---|---|---|
| Equals | = | All | The condition is true if both expressions have exactly the same value. For text, the case of the expression must be the same. So, for example, 'Smith' is not equal to 'SMITH'. |
| Not Equal to | != <> >< | All | The condition is true if the result of the first expression does NOT have the same value as the result of the second expression. |

| Comparator | Symbols | Data Types | Meaning |
|---|---|---|---|
| Greater than | > | All | The condition is true if the first expression is alphabetically after, or has a numerically greater value, or a later date than the second expression. |
| Less than | < | All | The condition is true if the first expression is alphabetically before, or has a numerically smaller value, or an earlier date than the second expression. |
| Greater than or equal to | >=<br>=> | All | The condition is true if either the greater than OR the equal to comparator returns a true result. |
| Less than or equal to | <=<br>=< | All | The condition is true if either the less than OR the equal to comparator returns a true result. |

| Comparator | Symbols | Data Types | Meaning |
|---|---|---|---|
| Like | **LIKE** | Text | The condition is true if the two text expressions match according to the rules of the LIKE syntax. You can include Wildcards in the text to allow searching for text that matches a pattern, or words that begin with a certain sequence of letters.<br>- percent sign (%) matches any number of characters in that position<br>- underscore (_) matches a single character occurrence in that position. |
| Not Like | NOT LIKE | Text | The condition is true if the two text expressions do NOT match according to the rules of the LIKE syntax. |

There is a special comparator called WAS DEFAULTED that you can use to test database items and input values. If there is no value available for an input value or database item, the formula uses a default value. The condition containing the WAS DEFAULTED comparator is True if a default value was used. For example:

```
DEFAULT FOR employee_middle_name IS ' '

  IF  employee_middle_name WAS DEFAULTED

    THEN

     /* special processing */
```

### Correct Use of Brackets

If you group more than one statement under the THEN or ELSE clauses, enclose the

group of statements within brackets, that is ( and ). In the absence of brackets, Oracle FastFormula conditionally executes *only* the statement that immediately follows the THEN or ELSE clause. Any other statements are executed unconditionally. For example, when the following formula runs, High_salary is always set to Y:

```
High_salary = 'N'

IF Salary > 20000

  THEN Tax = Salary * .25

       High_salary = 'Y'
```

To prevent this, use brackets as follows:

```
High_salary = 'N'

IF Salary > 20000

  THEN

 (

  Tax = Salary * .25

  High_salary = 'Y'

  )
```

## Return Statement

Use the Return statement to return values in local variables to the application. Oracle FastFormula can pass back any number of variables. The variable does not need to contain a value.

Example:

```
/* Formula: LONDON ALLOWANCE FORMULA */

INPUTS ARE this_months_extra (number)

London_allowance = (grade_pay/20 + this_months_extra)

RETURN London_allowance
```

Notice that you do not have to declare the data type of local variables in the Return statement (as the formula already knows the data type).

Oracle FastFormula stops executing the formula when it reaches the Return statement. Any statements after the Return statement are ignored.

## Formula Compilation

When you have created or edited a formula in the Formula window, you choose the Verify button to compile it.

If you need to compile many formulas at the same time, you can run the Bulk Compile Formulas process in the Submit Requests window. For example, you run this process when you upgrade your legislative information, which includes formulas.

> **Note:** If you make any changes to a function after you have compiled a formula that uses it, you need to recompile the formula for the changes to take effect.

## Formula Errors

There are two types of error that can occur when using Oracle FastFormula:

- Verify-time errors occur in the Formulas window when you run the formula verification. An error message explains the nature of the error.

  Common verify-time errors are syntax errors resulting from typing mistakes.

- Run-time errors occur when a problem arises while a formula is running. The usual cause is a data problem, either in the formula or in the application database.

The basic Oracle FastFormula errors that can occur at run-time are:

- **Uninitialized Variables**: An uninitialized local variable is one that has no value when the formula runs. The term 'uninitialized' means you have not assigned any value to the variable before you try to use it. This causes an error in all statements except the Return statement. For example:

```
IF   (tax_band < 2000)

  THEN tax = salary / 8

IF   (tax_band > 2000)

  THEN tax = salary / 10

IF tax > 1000

  THEN  ...
```

  This formula fails with an 'Uninitialized variable' message (for the variable tax) if the tax band is set to 2000.

- **Divide by Zero**: Dividing a number by zero is an operation that provides no logical result. If this situation ever arises, Oracle FastFormula passes a code back to the application indicating an error (the application then takes the appropriate action).

  Always check for the possibility of a divide by zero error if there is any chance it could occur. For example, the formula:

  ```
  x = salary/contribution_proportion
  ```

  produces an error if the contribution proportion is set to zero. In this formula, check for the divide by zero condition as follows:

  ```
  IF   contribution_proportion = 0

  THEN (message = 'The contribution proportion is not

  valid.' RETURN message)

  ELSE x = salary/contribution_proportion
  ```

- **No Data Found**: A database item supposed to be in the database was not found. This represents an error in the application data.

- **Too Many Rows**: The database item definition within the application caused more than one value to be fetched from the database.

- **Value Exceeded Allowable Range**: This can occur for a variety of reasons such as:
  - exceeding the maximum allowable length of a string (which is 240 characters)
  - rounding up a number to an excessive number of places, for example, round (1,100)
  - using an invalid date, for example, 39-DEC-1990.

- **Invalid Number**: This occurs only when a database item contains an item that does not make sense as a number.

- **Null Data Found**:A database item was found to have a null value when it should have had a non-null value. Use the Default statement for database items marked as Default Required in the Database Items window.

## Database Items

This topic lists the database items available to you in Oracle HRMS for writing formulas and defining QuickPaint reports. The database items are grouped into two listings:

- *Static Database Items*, page 1-63

- *Dynamic Database Items*, page 1-87

Static database items are shipped with the system and you cannot modify them. Dynamic database items are created by Oracle HRMS processes whenever you define new elements or other related entities.

## Static Database Items

Static database items are shipped with the system and you cannot modify them.

***Accrual Plan Information***

| Database item | Description |
|---|---|
| **ACP_CARRIED_OVER_DATE** | The effective date stored in the latest Carry Over element entry for the assignment and accrual plan |
| **ACP_CARRIED_OVER_PTO** | The amount of PTO stored for an assignment in the latest Carry Over element entry |
| **ACP_CATEGORY** | The category of accrual plan |
| **ACP_CONTINUOUS_SERVICE_DATE** | An employee's adjusted service date |
| **ACP_ENROLLMENT_END_DATE** | The end date of an employee's enrollment in the accrual plan |
| **ACP_ENROLLMENT_START_DATE** | The start date of an employee's enrollment in the accrual plan |
| **ACP_INELIGIBLE_PERIOD_LENGTH** | The length of the plan's ineligibility period (a number) |
| **ACP_INELIGIBLE_PERIOD_TYPE** | The units (e.g. months) for measuring the length of the plan's ineligibility period |
| **ACP_NAME** | The name of the accrual plan |
| **ACP_SERVICE_START_DATE** | The start date of an employee's period of service |

| Database item | Description |
|---|---|
| **ACP_START** | The rule for determining the start date for new hires in the plan |
| **ACP_TERMINATION_DATE** | The end date of an employee's period of service |
| **ACP_UOM** | The units (hours or days) for accumulating PTO |
| **PTO_ACCRUAL_PLAN_ID** | The id of the accrual plan. |
| **PTO_DATE_PAID_CALCULATION_DATE** | The last day of the period for calculating accruals in the payroll run (when the PTO Balance Type for the business group is Date Paid) |
| **PTO_DATE_EARNED_CALCULATION_DATE** | The last day of the period for calculating accruals in the payroll run (when the PTO Balance Type for the business group is Date Earned) |
| **PTO_DATE_PAID_START_DATE** | The first day of the period for calculating accruals in the payroll run (when the PTO Balance Type for the business group is Date Paid) |
| **PTO_DATE_EARNED_START_DATE** | The first day of the period for calculating accruals in the payroll run (when the PTO Balance Type for the business group is Date Earned) |

***Applicant Information***

| Database item | Description |
|---|---|
| **APL_DATE_END** | The date the application ended |
| **APL_DATE_RECEIVED** | The date the application was received |

***Assignment Address Detail (US/UK only)***

| Database item | Description |
|---|---|
| **PER_ADR_UK_COUNTY** | The assignment's home county (UK only) |
| **PER_ADR_US_COUNTY** | The assignment's county (US only) |
| **PER_ADR_US_STATE** | The assignment's state (US only) |
| **PER_ADR_US_STATE_CODE** | The assignment |

***Contact Addresses***

| Database item | Description |
|---|---|
| **CON_ADR_CITY** | The name of the contact's town or city |
| **CON_ADR_COUNTRY** | The name of the contact's country |
| **CON_ADR_DATE_FROM** | The first date on which the contact can be contacted |
| **CON_ADR_DATE_TO** | The last date on which the contact can be contacted |
| **CON_ADR_LINE_1** | The first line of the contact's address |
| **CON_ADR_LINE_2** | The second line of the contact's address |
| **CON_ADR_LINE_3** | The third line of the contact's address |
| **CON_ADR_PHONE_1** | The contact's first telephone number |
| **CON_ADR_PHONE_2** | The contact's second telephone number |
| **CON_ADR_PHONE_3** | The contact's third telephone number |
| **CON_ADR_POSTAL_CODE** | The contact's postal code |

| Database item | Description |
|---|---|
| **CON_ADR_REGION_1** | The first line of the contact's region |
| **CON_ADR_REGION_2** | The second line of the contact's region |
| **CON_ADR_REGION_3** | The third line of the contact's region |

***Contact Information***

| Database item | Description |
|---|---|
| **CON_AGE** | The contact's age |
| **CON_APP_NUMBER** | The contact's applicant number |
| **CON_BENEFICIARY_FLAG** | The contact's beneficiary flag |
| **CON_CURRENT_APP** | Whether the contact is a current applicant (yes/no) |
| **CON_CURRENT_EMP** | Whether the contact is a current employee (yes/no) |
| **CON_CURRENT_CWK** | Whether the contact is a current contingent worker (yes/no) |
| **CON_CWK_NUMBER** | The contact's contingent worker number |
| **CON_DATE_END** | The end date of the contact's relationship |
| **CON_DATE_START** | The start date of the contact's relationship |
| **CON_DATE_OF_BIRTH** | The contact's date of birth |
| **CON_DEPENDENT_FLAG** | The contact's dependent flag |
| **CON_DISABLED** | Whether the contact is disabled (yes/no) |
| **CON_EMP_NUMBER** | The contact's employee number |

| Database item | Description |
|---|---|
| **CON_END_DATE** | The date to which this contact information is effective |
| **CON_END_LIFE_REASON_ID** | ID for the reason for the end of the relationship |
| **CON_FIRST_NAME** | The contact's first name |
| **CON_FULL_NAME** | The contact's full name |
| **CON_KNOWN_AS** | The contact's preferred name |
| **CON_LAST_NAME** | The contact's last name |
| **CON_MARITAL_STATUS** | The contact's marital status |
| **CON_MIDDLE_NAMES** | The contact's middle names |
| **CON_NATIONAL_IDENTIFIER** | The contact's national identifier |
| **CON_NATIONALITY** | The contact's nationality |
| **CON_PERSONAL_FLAG** | Personal relationship flag |
| **CON_PERSON_TYPE** | The contact's person type - employee or applicant, for example |
| **CON_RLTD_PER_RSDS_W_DSGNTR** | Whether the contact shares the same residence as the employee. |
| **CON_RELATIONSHIP** | The relationship of the contact to the employee |
| **CON_SEQUENCE_NUMBER** | Contact's sequence number |
| **CON_SEX** | The contact's sex |
| **CON_START_DATE** | The date from which this contact information is effective |
| **CON_START_LIFE_REASON_ID** | ID for reason for the start of the relationship |

| Database item | Description |
|---|---|
| **CON_THIRD_PARTY_PAY_FLAG** | Third party payments relationship flag |
| **CON_TITLE** | The contact's title |
| **CON_WORK_PHONE** | The contact's work telephone number |

***Contingent Worker***

| Database item | Description |
|---|---|
| **CWK_START_DATE** | The contingent workers's start date |
| **CWK_END_DATE** | The contingent worker's end date |
| **CWK_PROJ_END_DATE** | The contingent worker's projected end date |
| **CWK_LEAVING_REASON** | The reason the contingent worker left |
| **CWK_LEAVING_REASON_CODE** | The code of the reason the contingent worker left |

***Contracts Information***

| Database item | Description |
|---|---|
| **CTR_STATUS_MEANING** | Contract status meaning |
| **CTR_TYPE_MEANING** | Contract type meaning |
| **CTR_STATUS** | Contract status code |
| **CTR_TYPE** | Contract type code |

***Date Information***

| Database item | Description |
|---|---|
| **SESSION_DATE** | The effective date from FND_SESSIONS |
| SYSDATE | The system date |

***Element Type Details***

| Database item | Description |
|---|---|
| **CURRENT_ELEMENT_TYPE_ID** | The type ID of the element being processed |
| **ELEMENT_NAME** | The name of the element being processed |
| **ENTRY_END_DATE** | The end date of the original entry |
| **ENTRY_START_DATE** | The start date of the original entry |

***Employee Assignment Information***

| Database item | Description |
|---|---|
| **ASG_ASSIGNMENT_CATEGORY** | The category for the assignment |
| **ASG_ASSIGNMENT_SEQUENCE** | This is used as a default for assignment number |
| **ASG_BARGAINING_UNIT_CODE** | The employee's bargaining unit code |
| **ASG_CHANGE_REASON** | The change reason for the assignment |
| **ASG_DATE_FROM** | The date from which this assignment information is effective |
| **ASG_DATE_TO** | The date to which this assignment information is effective |

| Database item | Description |
|---|---|
| **ASG_EMPLOYMENT_CATEGORY** | The employment category for the assignment |
| **ASG_EMPLOYMENT_CATEGORY_CODE** | The employment category code for the assignment |
| **ASG_END_TIME** | The standard end time for the assignment |
| **ASG_FREQ** | The frequency for which the assignment working hours are measured |
| **ASG_FREQ_CODE** | The working hours frequency code for the assignment |
| **ASG_FTE_VALUE** | The full-time equivalent budget actual value for the assignment |
| **ASG_FULL_TIME_FREQ** | The full-time frequency for the assignment |
| **ASG_FULL_TIME_HOURS** | The full-time working hours for the assignment |
| **ASG_GRADE** | The employee's grade |
| **ASG_GRADE_DATE_FROM** | The date from which this assignment grade information is effective |
| **ASG_GRADE_DATE_TO** | The date to which this assignment grade information is effective |
| **ASG_GROUP** | The employee's group |
| **ASG_HEAD_VALUE** | The head budget value for the assignment |
| **ASG_HOURS** | The standard number of working hours for the assignment |
| **ASG_INT_ADDR_LINE** | The internal address of the assignment |
| **ASG_JOB** | The employee's job |

| Database item | Description |
|---|---|
| **ASG_JOB_DATE_FROM** | The date from which this assignment job information is effective |
| **ASG_JOB_DATE_TO** | The date to which this assignment job information is effective |
| **ASG_LABOUR_UNION_MEMBER_FLAG** | Whether the assignment is a union member |
| **ASG_LAST_CHANGE_REASON** | The reason the salary was changed |
| **ASG_LAST_EARNED_PAYROLL_NAME** | The payroll name the assignment was last processed with as at the date earned |
| **ASG_LAST_EARNED_PERIOD_ID** | The time period ID the assignment was last processed with as at the date earned |
| **ASG_LAST_EARNED_PERIOD_NAME** | The time period name the assignment was last processed with as at the date earned |
| **ASG_LAST_EARNED_PERIOD_NUMBER** | The time period number the assignment was last processed with as at the date earned |
| **ASG_LAST_PERFORMANCE_DATE** | Last performance review date |
| **ASG_LAST_PERFORMANCE_LOCATION** | Last performance review location |
| **ASG_LAST_PERFORMANCE_RATING** | Last performance review rating |
| **ASG_LAST_PERFORMANCE_TYPE** | Last performance review type |
| **ASG_LAST_PROC_PAYROLL_NAME** | The payroll name the assignment was last processed |
| **ASG_LAST_PROC_PERIOD_ID** | The time period ID the assignment was last processed |
| **ASG_LAST_PROC_PERIOD_NAME** | The period name the assignment was last processed |
| **ASG_LAST_PROC_PERIOD_NUMBER** | The period number the assignment was last processed |

| Database item | Description |
|---|---|
| **ASG_LAST_PROPOSED_SALARY_CHANGE** | The proposed salary change |
| **ASG_LAST_PROPOSED_SALARY_PERCENT** | The proposed salary change as a percentage |
| **ASG_LAST_SALARY_CHANGE_APPROVED** | Whether the last proposed salary change has been approved |
| **ASG_LAST_SALARY_DATE** | The last salary change date |
| **ASG_LOCATION** | The employee's location |
| **ASG_LOC_INACTIVE_DATE** | The date to which the location information is effective |
| **ASG_MANAGER** | Whether the assignment is a managerial assignment (yes/no) |
| **ASG_MONEY_VALUE** | The assignment's money budget actual value |
| **ASG_NEXT_PERFORMANCE_DATE** | Next performance review date |
| **ASG_NEXT_SALARY_DATE** | The date of the next salary change |
| **ASG_NUMBER** | The assignment number |
| **ASG_ORG** | The employee's organization |
| **ASG_ORG_DATE_FROM** | The date from which assignment organization information is effective |
| **ASG_ORG_DATE_TO** | The date to which assignment organization information is effective |
| **ASG_ORG_INT_EXT** | The assignment organization internal external flag |
| **ASG_ORG_LOCATION** | The location of the assignment organization |
| **ASG_ORG_TYPE** | The assignment organization type |

| Database item | Description |
|---|---|
| **ASG_PAYROLL** | The employee's payroll |
| **ASG_PERFORMANCE_REVIEW_FREQUENCY** | The performance review period for the assignment |
| **ASG_PERFORMANCE_REVIEW_PERIOD** | The performance review frequency for the assignment |
| **ASG_PER_STATUS_DP** | Personal status for the assignment (as of Date Paid) |
| **ASG_PFT_VALUE** | The PFT budget value for the assignment |
| **ASG_POS_HOURS** | The standard number of working hours for the position |
| **ASG_POSITION** | The employee's position |
| **ASG_POS_DATE_FROM** | The date from which this assignment position information is effective |
| **ASG_POS_DATE_TO** | The date to which this assignment position information is effective |
| **ASG_POS_END_TIME** | The standard end time for the assignment position |
| **ASG_POS_FREQ** | The frequency for which the assignment position's hours are measured |
| **ASG_POS_FREQUENCY** | The frequency for which the assignment position's hours are measured |
| **ASG_POS_FTE** | The assignment position FTE |
| **ASG_POS_LOCATION** | The location of the assignment position |
| **ASG_POS_PROB_PERIOD** | The probation period for the assignment position (in numeric format) |

| Database item | Description |
|---|---|
| **ASG_POS_PROBATION_PERIOD** | The probation period for the assignment position (formatted as text) |
| **ASG_POS_PROBATION_PERIOD_UNITS** | The units used to measure the probation period |
| **ASG_POS_START_TIME** | The standard start time for the assignment position |
| **ASG_POS_WORKING_HOURS** | The standard number of working hours for the position |
| **ASG_PRIMARY** | Whether this is the employee's primary assignment (yes/no) |
| **ASG_PRIMARY_CODE** | The assignment's primary code |
| **ASG_PROB_END_DATE** | The probation period end date |
| **ASG_PROB_PERIOD** | The assignment's probation period |
| **ASG_PROB_UNITS** | The units of the assignment's probation period |
| **ASG_REC_FULL_NAME** | The full name for the recruiter |
| **ASG_RELIEF** | The relief position if the current position holder is absent |
| **ASG_SALARY** | The current salary for an employee |
| **ASG_SALARY_BASIS** | The payment basis (i.e. frequency) for the assignment, e.g. monthly |
| **ASG_SALARY_BASIS_ANNUALIZATION_FACTOR** | The payment basis pay annualization factor for the assignment |
| **ASG_SALARY_BASIS_CODE** | The payment basis lookup code for the assignment |
| **ASG_SALARY_BASIS_GRADE_ANNUALIZATION_FACTOR** | The payment grade basis pay annualization factor for the assignment |

| Database item | Description |
|---|---|
| **ASG_SALARY_BASIS_NAME** | The salary basis name for the assignment |
| **ASG_SALARY_ELEMENT** | The display element name |
| **ASG_SALARY_ELEMENT_VALUE_NAME** | The display input value name |
| **ASG_SALARY_GRADE_RATE** | The display rate name |
| **ASG_SALARY_RATE_BASIS** | The salary rate basis |
| **ASG_SALARY_REVIEW_FREQUENCY** | The salary review frequency for the assignment |
| **ASG_SALARY_REVIEW_PERIOD** | The salary review period for the assignment |
| **ASG_START_DATE** | The start date of the assignment |
| **ASG_START_TIME** | The standard start time for the assignment |
| **ASG_STATUS** | The primary status for the assignment |
| **ASG_SUCCESSOR** | The position name that will succeed into this position |
| **ASG_SUP_FULL_NAME** | The full name for the supervisor |
| **ASG_TYPE** | Whether this is an employee or applicant assignment |
| **ASG_VACANCY** | The name of the vacancy applied for |
| **ASG_WORK_AT_HOME** | The work at home code for an assignment |
| **ASSIGNMENT_ACTION_END_DATE** | The end date of the assignment action |
| **ASSIGNMENT_ACTION_START_DATE** | The start date of the assignment action |
| **CHEQUE_UK_NUMBER** | The cheque number for the assignment action (UK spelling) |

| Database item | Description |
|---|---|
| **CHECK_US_NUMBER** | The check number for the assignment action (US spelling) |
| **GROSSUP_AMOUNT** | The gross up amount to be added to the net amount |

***Employee Hire Information***

| Database item | Description |
|---|---|
| **EMP_HIRE_DATE** | The employee's hire date |
| **EMP_LAST_PROCESS_DATE** | The date the employee was last processed |
| **EMP_LEAVING_REASON** | The reason the employee left |
| **EMP_LEAVING_REASON_CODE** | The code for the reason the employee left |
| **EMP_PROJ_TERM_DATE** | The employee's projected termination date |
| **EMP_TERM_ACCEPTED_BY** | The person who accepted the employee's notice |
| **EMP_TERM_DATE** | The employee's termination date |

***Home Address Details (UK only)***

| Database item | Description |
|---|---|
| **PER_ADR_UK_COUNTY** | The person's home county (UK only) |

***Home Address Details (US only)***

| Database item | Description |
|---|---|
| **PER_ADR_US_COUNTY** | The person's county (US only) |
| **PER_ADR_US_STATE** | The person's state (US only) |
| **PER_ADR_US_STATE_CODE** | The person's state code (US only) |

***Human Resources Intelligence***

| Database item | Description |
|---|---|
| **HRI_ASG_EMPLOYMENT_CATEGORY_CODE** | The employment category code |
| **HRI_ASG_FREQ_CODE** | The assignment working hours frequency code |
| **HRI_ASG_FULL_TIME_FREQ** | The full-time frequency |
| **HRI_ASG_FULL_TIME_HOURS** | The full-time working hours |
| **HRI_ASG_HOURS** | The normal working hours of the assignment |
| **HRI_ASG_PER_EMP_PTU** | Person Assignment EMP User Person Type |
| **HRI_ASG_PER_APL_PTU** | Person Assignment APL User Person Type |
| **HRI_ASG_PER_CWK_PTU** | Person Assignment CWK User Person Type |
| **HRI_ASG_PRIMARY_CODE** | The primary assignment code |

***Location Details***

| Database item | Description |
|---|---|
| **LOC_ADR_LINE_1** | The first line of the assignment's work address |

| Database item | Description |
|---|---|
| **LOC_ADR_LINE_2** | The second line of the assignment's work address |
| **LOC_ADR_LINE_3** | The third line of the assignment's work address |
| **LOC_ADR_POSTAL_CODE** | The postal code for the assignment's work address |
| **LOC_ADR_REGION_1** | The first line of the assignment's region |
| **LOC_ADR_REGION_2** | The second line of the assignment's region |
| **LOC_ADR_REGION_3** | The third line of the assignment's region |
| **LOC_ADR_PHONE_1** | The assignment's first work telephone number |
| **LOC_ADR_PHONE_2** | The assignment's second work telephone number |
| **LOC_ADR_PHONE_3** | The assignment's third work telephone number |
| **LOC_ADR_CITY** | The town or city where the assignment works |
| **LOC_ADR_COUNTRY** | The country where the assignment works |

***Payroll Details***

| Database item | Description |
|---|---|
| **ACTION_TYPE** | Action type of target payroll action |
| **LAST_REG_PAYMENT_PERIOD** | The last regular payment period |
| **LAST_REG_PAYMENT_PERIOD_START_DATE** | The start date of the last regular payment period |
| **PAY_EARNED_CUT_OFF_DATE** | The cut-off date of the earned period |

| Database item | Description |
|---|---|
| **PAY_EARNED_DIRECT_DEPOSIT_DATE** | The direct deposit date of the earned period |
| **PAY_EARNED_END_DATE** | The end date of the earned period |
| **PAY_EARNED_PAY_ADVICE_DATE** | The pay advice date of the earned period |
| **PAY_EARNED_PERIOD_ID** | The ID of the time period of the earned period |
| **PAY_EARNED_PERIOD_NAME** | The period name for the earned period |
| **PAY_EARNED_PERIOD_NUMBER** | The period number for the earned period |
| **PAY_EARNED_START_DATE** | The start date of the earned period |
| **PAY_NO_OF_SCHEDULED_PAYMENTS** | The start date of the earned period |
| **PAY_PERIODS_PER_YEAR** | The number of payable periods in the year (as of date earned) |
| **PAY_PERIODS_PER_YEAR_DP** | The number of payable periods in the year (as of date paid) |
| **PAY_PROC_PERIOD_CUT_OFF_DATE** | The cut off date for the payroll period (as of date earned) |
| **PAY_PROC_PERIOD_CUT_OFF_DATE_DP** | The cut off date for the payroll period (as of date paid) |
| **PAY_PROC_PERIOD_DATE_PAID** | The date the payroll was paid |
| **PAY_PROC_PERIOD_DIRECT_DEPOSIT_DATE** | The direct deposit date for the payroll period (as of date earned) |
| **PAY_PROC_PERIOD_DIRECT_DEPOSIT_DATE_DP** | The direct deposit date for the payroll period (as of date paid) |
| **PAY_PROC_PERIOD_END_DATE** | The end date of the payroll period (as of date earned) |
| **PAY_PROC_PERIOD_END_DATE_DP** | The end date of the payroll period (as of date paid) |

| Database item | Description |
|---|---|
| **PAY_PROC_PERIOD_ID** | The ID of the time period for the payroll (as of date earned) |
| **PAY_PROC_PERIOD_ID_DP** | The id of the time period for the payroll (as of date paid) |
| **PAY_PROC_PERIOD_NAME** | The period name for the payroll (as of date earned) |
| **PAY_PROC_PERIOD_NAME_DP** | The period name for the payroll (as of date paid) |
| **PAY_PROC_PERIOD_NUMBER** | The current period number for the payroll (as of date earned) |
| **PAY_PROC_PERIOD_NUMBER_DP** | The current period number for the payroll (as of date paid) |
| **PAY_PROC_PERIOD_PAY_ADVICE_DATE** | The pay advice date for the payroll period (as of date earned) |
| **PAY_PROC_PERIOD_PAY_ADVICE_DATE_DP** | The pay advice date for the payroll period (as of date paid) |
| **PAY_PROC_PERIOD_START_DATE** | The start date of the payroll period (as of date earned) |
| **PAY_PROC_PERIOD_START_DATE_DP** | The start date of the payroll period (as of date paid) |
| **PAYROLL_ARREARS_FLAG** | Value of arrears flag for payrolls |

***People Addresses***

| Database item | Description |
|---|---|
| **PER_ADR_CITY** | The name of the person's town or city |
| **PER_ADR_COUNTRY** | The name of the person's country |

| Database item | Description |
|---|---|
| **PER_ADR_COUNTRY_CODE** | The person's country code |
| **PER_ADR_DATE_FROM** | The first date on which the person can be contacted at this address |
| **PER_ADR_DATE_TO** | The last date on which the person can be contacted at this address |
| **PER_ADR_LINE_1** | The first line of the person's address |
| **PER_ADR_LINE_2** | The second line of the person's address |
| **PER_ADR_LINE_3** | The third line of the person's address |
| **PER_ADR_PHONE_1** | The person's first contact number |
| **PER_ADR_PHONE_2** | The person's second contact number |
| **PER_ADR_PHONE_3** | The person's third contact number |
| **PER_ADR_POSTAL_CODE** | The person's postal code |
| **PER_ADR_REGION_1** | The first line of the person's region |
| **PER_ADR_REGION_2** | The second line of the person's region |
| **PER_ADR_REGION_3** | The third line of the person's region |

***People Information***

| Database item | Description |
|---|---|
| **PER_1099R_NAME** | Employee details for 1099R |
| **PER_AGE** | The person's age |
| **PER_APPLICANT_NUMBER** | The person's applicant number |

| Database item | Description |
|---|---|
| **PER_BENEFIT_GROUP_ID** | The ID of the person's benefit group |
| **PER_COORD_BEN_MED_PLN_NO** | The benefits medical plan number for the person |
| **PER_COORD_BEN_NO_CVG_FLAG** | Whether there is any other benefits coverage |
| **PER_CURRENT_APP** | Whether the person is a current applicant (yes/no) |
| **PER_CURRENT_CWK** | Whether the person is a current contingent worker (yes/no) |
| **PER_CURRENT_EMP** | Whether the person is a current employee (yes/no) |
| **PER_CWK_NUMBER** | The person's contingent worker number |
| **PER_DATE_OF_BIRTH** | The person's date of birth |
| **PER_DATE_OF_DEATH** | The person's date of death |
| **PER_DATE_VERIFIED** | The date the employee last verified his or her personal data |
| **PER_DISABLED** | Whether the person is disabled (yes/no) |
| **PER_DPNT_ADOPTION_DATE** | The person's dependent's adoption date |
| **PER_DPNT_VLNTRY_SVCE_FLAG** | Whether the dependent is on voluntary service |
| **PER_EMP_NUMBER** | The person's employee number |
| **PER_FIRST_NAME** | The person's first name |
| **PER_FULL_NAME** | The person's full name |
| **PER_KNOWN_AS** | The person's preferred name |
| **PER_LAST_NAME** | The person's last name |

| Database item | Description |
|---|---|
| **PER_MAIL_DESTINATION** | The person's mail destination |
| **PER_MARITAL_STATUS** | The person's marital status |
| **PER_MIDDLE_NAMES** | The person's middle names |
| **PER_NATIONALITY** | The person's nationality |
| **PER_NATIONAL_IDENTIFIER** | The person's national identifier |
| **PER_ORIGINAL_DATE_OF_HIRE** | Date the person was first hired |
| **PER_PERSON_TYPE** | Type of person (employee or applicant, for example) |
| **PER_PREFIX** | The person's name prefix |
| **PER_PREV_LAST_NAME** | The person's previous last name |
| **PER_RECEIPT_OF_DEATH_CERT_DATE** | Date of receipt of the person's death certificate |
| **PER_SEND_EXPENSES** | Where to send the person's expenses (home/office) |
| **PER_SEX** | The person's sex |
| **PER_SUFFIX** | The person's name suffix |
| **PER_TITLE** | The person's title |
| **PER_USES_TOBACCO_FLAG** | Whether the person uses tobacco |
| **PER_WORK_PHONE** | The person's work telephone number |

***Person Types***

| Database item | Description |
|---|---|
| **PTU_CON_PERSON_TYPE** | The contact's person type, for example employee, applicant |
| **PTU_PER_PERSON_TYPE** | The type of person, for example employee, applicant |
| **PTU_REC_PERSON_TYPE** | The recruiter's person type, for example employee |
| **PTU_SUP_PERSON_TYPE** | The supervisor's person type, for example employee |

***Recruiter Information***

| Database item | Description |
|---|---|
| **REC_CURRENT_APP** | Whether the recruiter is a current applicant (yes/no) |
| **REC_CURRENT_CWK** | Whether the recruiter is a current contingent worker (yes/no) |
| **REC_CWK_NUMBER** | The recruiter's contingent worker number |
| **REC_CURRENT_EMP** | Whether the recruiter is a current employee (yes/no) |
| **REC_EMP_NUMBER** | The recruiter's employee number |
| **REC_GRADE** | The recruiter's grade |
| **REC_INT_ADDR_LINE** | The recruiter's internal address |
| **REC_JOB** | The recruiter's job |
| **REC_LOCATION** | The recruiter's work location |

| Database item | Description |
|---|---|
| **REC_MANAGER** | Whether the assignment is a managerial assignment (yes/no) |
| **REC_ORG** | The name of the recruiter's organization |
| **REC_PERSON_TYPE** | The recruiter's person type (employee or applicant, for example) |
| **REC_POSITION** | The recruiter's position |
| **REC_WORK_PHONE** | The recruiter's work telephone number |

***Supervisor Information***

| Database item | Description |
|---|---|
| **SUP_CURRENT_CWK** | Whether the supervisor is a current contingent worker (yes/no) |
| **SUP_CURRENT_EMP** | Whether the supervisor is a current employee (yes/no) |
| **SUP_CWK_NUMBER** | The contingent worker number of the supervisor |
| **SUP_DATE_FROM** | The date from which this supervisor information is effective |
| **SUP_DATE_TO** | The date to which this supervisor information is effective |
| **SUP_EMAIL_ADDRESS** | The supervisor's email address |
| **SUP_EMP_NUMBER** | The supervisor's employee number |
| **SUP_GRADE** | The supervisor's grade |
| **SUP_INT_ADDR_LINE** | The supervisor's internal address |

| Database item | Description |
|---|---|
| **SUP_JOB** | The supervisor's job |
| **SUP_LOCATION** | The supervisor's work location |
| **SUP_MANAGER** | Whether the assignment is a managerial assignment (yes/no) |
| **SUP_ORG** | The supervisor's organization |
| **SUP_PERSON_TYPE** | The supervisor's person type |
| **SUP_POSITION** | The supervisor's position |
| **SUP_WORK_PHONE** | The supervisor's work telephone number |

***Work Address Details (UK only)***

| Database item | Description |
|---|---|
| **LOC_ADR_UK_COUNTY** | The assignment's work county (UK only) |

***Work Address Details (US only)***

| Database item | Description |
|---|---|
| **LOC_ADR_US_COUNTY** | The assignment's work county (US only) |
| **LOC_ADR_US_STATE** | The assignment's work state (US only) |
| **LOC_ADR_US_STATE_CODE** | The assignment's work state code (US only) |

## Static Database Items for Oracle US Federal HR

Static database items are shipped with the system and you cannot modify them. For a list of the standard static database items, refer to Static Database Items, page 1-63.

You can use the following Oracle Federal HR data items with Fast Formula Rules.

- Location Extra Information: Duty Station ID
- Assignment Extra Information: Step or Rate, Tenure, Annuitant Indicator, Pay Rate Determinant, Work Schedule, Part-Time Hours Biweekly, Duty Status
- Person Extra Information: Citizenship, Veterans Preference, Veterans Preference for RIF, Veterans Status, Appointment Type, Type of Employment, Race or National Origin, Date Last Promotion, SCD Leave, SCD Civilian, SCD RIF, SCD TSP, Date From Retained Grade, Date To Retained Grade, Retained Grade, Retained Step or Rate, Retained Pay Plan, Retained Pay Table ID, Retained Pay Basis, FERS Coverage, Previous Retirement Coverage, Frozen Service, NAF Retirement Indicator
- Position Extra Information: Valid Grade, Target Grade, Pay Table ID, Pay Basis, Employment Category Group, Occupation Category Code, FLSA Category, Bargaining Unit Status, Supervisory Status, Position Occupied, Intelligence Position Ind, LEO Position Indicator, Position Type

If you define agency-specific Extra Information types or segments, you can create database items for them.

See: Setting Up Extra Information Types (Excluding Organization EITs), *Oracle HRMS Configuring, Reporting, and System Administration Guide*

## Dynamic Database Items

Dynamic database items are created by Oracle HRMS processes whenever you define new elements or other related entities.

### Element Database Items

When you define a new element, Oracle HRMS runs a process to create a number of related database items for it. To ensure easy recognition of these items, the process adds the element name <ENAME> to each one. It also creates further database items for each pay and input value you use <INAME>.

Here is a list of database items created each time you define an element using the Element window:

| Database item | Description |
|---|---|
| **<ENAME>_BEN_CLASS** | The element's benefit classification |
| **<ENAME>_CLASSIFICATION** | The element's classification |

| Database item | Description |
|---|---|
| **<ENAME>_CLOSED_FOR_ENTRY** | Yes/no flag: translated into local language. If Yes, new element entries cannot be created but existing element entries can still be modified. |
| **<ENAME>_CLOSED_FOR_ENTRY_CODE** | Yes/no flag: If Yes, new element entries cannot be created but existing element entries can still be modified. |
| **<ENAME>_COSTABLE_TYPE** | The element's costable type (from lookup table) |
| **<ENAME>_COSTABLE_TYPE_CODE** | The element's costable type (code values) |
| **<ENAME>_COUNT** | The element entry count |
| **<ENAME>_END_DATE** | The date to which this element is effective |
| **<ENAME>_INPUT_CURRENCY_CODE** | The element's input currency code |
| **<ENAME>_LENGTH_OF_SERVICE** | The element's qualifying length of service |
| **<ENAME>_OUTPUT_CURRENCY_CODE** | The element's output currency code |
| **<ENAME>_PROCESSING_PRIORITY** | The element's processing priority |
| **<ENAME>_QUALIFYING_AGE** | The element's qualifying age |
| **<ENAME>_QUALIFYING_UNITS_CODE** | The qualifying length of service units (code values) |
| **<ENAME>_QUALIFYING_UNITS** | The qualifying length of service units (from lookup table) |
| **<ENAME>_REPORTING_NAME** | The element's reporting name |
| **<ENAME>_STANDARD_LINK** | Yes/no flag: yes = standard, no = discretionary |
| **<ENAME>_STANDARD_LINK_CODE** | Yes/no flag: yes = standard, no = discretionary |

| Database item | Description |
|---|---|
| **<ENAME>_<INAME>_UNIT_OF_MEASURE** | The element's unit of measure (from lookup table) |
| **<ENAME>_<INAME>_UNIT_OF_MEASURE_CODE** | The element's unit of measure (code values) |
| **<ENAME>_<INAME>_DEFAULT** | The element's default input value |
| **<ENAME>_<INAME>_MIN** | The element's minimum input value |
| **<ENAME>_<INAME>_MAX** | The element's maximum input value |

In addition to the items above, Oracle HRMS creates the following four items for elements defined with multiple entries *not* allowed:

| Database item | Description |
|---|---|
| **<ENAME>_<INAME>_ENTRY_VALUE** | The element value |
| **<ENAME>_<INAME>_USER_ENTERED_CODE** | Whether a value exists at the element entry level (yes/no) |
| **<ENAME>_<INAME>_START_DATE** | The element's start date |
| **<ENAME>_<INAME>_END_DATE** | The element's end date |

In addition to the common list above, Oracle HRMS creates the following item for elements defined with multiple entries allowed whose input values are numeric (that is, hours, integer, money or number).

| Database item | Description |
|---|---|
| **<ENAME>_<INAME>_ENTRY_VALUE** | The summed element values for the multiple entries |

The units for '<ENAME> <INAME> ENTRY VALUE' are generated for both recurring and nonrecurring elements and are user-definable. Oracle HRMS modifies the definition text to retrieve the entry value in the unit of measure as specified in the PAY_INPUT_VALUES_F table.

## Grade Rate Database Items

When you define a grade rate, Oracle HRMS runs a process to create a number of related database items for it. To ensure easy recognition of these items, the process adds the grade rate name <NAME> to each one.

Here is a list of database items created each time you define a grade rate using the Grade Rate window:

| Database item | Description |
|---|---|
| **GRADE_<NAME>_VALUE** | The grade rate's value |
| **GRADE_<NAME>_MINIMUM** | The grade rate's minimum value |
| **GRADE_<NAME>_MAXIMUM** | The grade rate's maximum value |

## Pay Scale Rate Database Items

When you define a pay scale rate, Oracle HRMS runs a process to create the following database item for it. To ensure easy recognition of this item, the process adds the rate name <NAME> to it.

| Database item | Description |
|---|---|
| **SPINE_<NAME>_VALUE** | The pay scale rates value |

## Descriptive Flexfield Database Items

When you define descriptive flexfield segments you make them available for use in QuickPaint by running the Create Descriptive Flexfield DB Items process from the Submit Requests window. This process creates database items for each of the descriptive flexfields listed below.

To ensure easy recognition of these items, the process adds the descriptive flexfield segment name <SEGMENT_NAME> to each one.

| Database item | Description |
|---|---|
| **PEOPLE_<SEGMENT_NAME>** | People descriptive flexfield database items |

| Database item | Description |
|---|---|
| **PAYROLLS_<SEGMENT_NAME>** | Payroll descriptive flexfield database items |
| **ASSIGNMENTS_<SEGMENT_NAME>** | Assignment descriptive flexfield database items |
| **GRADES_<SEGMENT_NAME>** | Grade descriptive flexfield database items |
| **ABSENCES_<SEGMENT_NAME>** | Absence descriptive flexfield database items |
| **ABSENCE_TYPES_<SEGMENT_NAME>** | Absence Type descriptive flexfield database items |
| **PERSON_ADDRESSES_<SEGMENT_NAME>** | Person Address descriptive flexfield database items |
| **EVENTS_<SEGMENT_NAME>** | Events descriptive flexfield database items |
| **JOBS_<SEGMENT_NAME>** | Jobs descriptive flexfield database items |
| **CONTACTS_<SEGMENT_NAME>** | Contacts descriptive flexfield database items |
| **PERIODS_OF_SERVICE_<SEGMENT_NAME>** | Periods of Service descriptive flexfield database items |
| **RECRUITMENT_ACTIVITIES_<SEGMENT_NAME>** | Recruitment Activities descriptive flexfield database items |
| **POSITION_<SEGMENT_NAME>** | Position descriptive flexfield database items |
| **APPLICATIONS_<SEGMENT_NAME** | Applications descriptive flexfield database items |
| **ORGANIZATION_<SEGMENT_NAME>** | Organization descriptive flexfield database items |

## Key Flexfield Database Items

When you define key flexfield segments you make them available for use in QuickPaint by running the Create Key Flexfield DB Items process from the Submit Requests window. This process creates database items for each of the key flexfields listed below.

To ensure easy recognition of these items, the process adds the key flexfield segment

name <SEGMENT_NAME> to each one.

Run this process for each of your Business Groups. If you define context-dependent key flexfield structures using BUSINESS_GROUP_ID as the reference field, the process creates database items for those flexfield segments as well. BUSINESS_GROUP_ID is the only reference field that the Create Key Flexfield DB Items process supports.

| Database item | Description |
|---|---|
| **COMP_KF_<SEGMENT_NAME>** | Competence key flexfield database items |
| **GRADE_KF_<SEGMENT_NAME>** | Grade key flexfield database items |
| **GROUP_KF_<SEGMENT_NAME>** | Group key flexfield database items |
| **JOB_KF_<SEGMENT_NAME>** | Job key flexfield database items |
| **POS_KF_<SEGMENT_NAME>** | Position key flexfield database items |
| **SCL_ASG_<LEGISLATION_CODE>_<ENAME>** | Assignment soft coded legislative flexfield database items |
| **SCL_ORG_<LEGISLATION_CODE>_<ENAME>** | Organization soft coded legislative flexfield database items |
| **SCL_PAY_<LEGISLATION_CODE>_<ENAME>** | Payroll soft coded legislative flexfield database items |

### Absence Database Items

When you define an absence type, Oracle HRMS runs a process to create the following database item for it. To ensure easy recognition of this item, the process adds the absence type name <ABSENCE_NAME> to it.

| Database item | Description |
|---|---|
| **<ABSENCE_NAME>_CUM_BALANCE** | The cumulative balance for an absence type |

## Formulas for Payroll Legislative Checks

Oracle FastFormula contains a formula type called Legislative Check that may have been set up by your localization team. This formula type can be set up to perform

certain checks during a payroll run. For example, you can catch errors such as negative gross pay and cause the payroll run to fail appropriately.

Sometimes you may not want to run the legislation check in the payroll run so it is possible that this functionality can be enabled or disabled using the HR:Execute Legislative Check Formula within Run user profile.

## Formulas for Benefits Administration

Oracle FastFormula contains many formula types that you can use for benefits administration. For example, grandfathered clauses and other special case scenarios may require you to write a FastFormula rule that defines special treatment for a subset of your benefits participants.

You can use Oracle FastFormula to calculate:

- The number of Hours Worked by a person in a given period
- A person's length of service
- The maximum coverage amount for a life insurance plan
- Participation Eligibility
- Other benefits related functions.

After you write a formula for use with Standard or Advanced Benefits, you link that formula to your plan design by selecting the formula in the Rule field of the appropriate window.

## Total Compensation Formula Types

The following table lists the formula types you can use in administering benefits with Oracle HRMS.

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Hours Worked Calculation | Used to determine an Hours Worked amount to be used for Eligibility, Coverage or Benefit, Premium, and Rate calculations | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | Hrs_Wkd_Fctr_Id; Value of the Hrs_Wkd_Fctr.Val, Min Val, Max Val | Amount | Determine total number of hours worked during the person's most recent pay period. |
| Age Calculation | Used to determine an Age value to be used for Eligibility, Coverage or Benefit, Premium, and Rate calculations | " | Value of the Age_Fctr.Val, Min Val, Max Val | Amount | |
| Length of Service Calculation | Used to determine a Service value to be used for Eligibility, Coverage or Benefit, Premium, and Rate calculations | " | LOS_Fctr_Id; Value of the LOS_Fctr. Val, Min Val, Max Val | Amount | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Compensation Calculation | Used to determine a Total Compensation amount to be used for Eligibility, Coverage or Benefit, Premium and Rate calculations | " | Comp_Lvl_Fctr_Id; Value of the Comp_Lvl_Fctr.Val, Min Val, Max Val | Amount | Sum the amounts for person's current Regular Salary defined balance as of the beginning of the year, Prior Year Total Commission benefits balance type, and Prior Year Total Bonus benefits balance type, and return the total amount. |
| Rate Value Calculation | Calculates a rate amount for a person. May be used to calculate a base or variable rate. | " | Acty_Base_Rt_Id or Vrbl_Rt_Id; Value of the Acty_Base_Rt.Val, Min Val, Max Val; Value of the Vrbl_Rt_Prfl. Val, Min Val, Max Val, | Amount | If a person is currently enrolled in this plan, or has been enrolled in this plan within the past 2 years, then the rate is equal to the rate person is currently paying. Otherwise, rate is equal to X. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Premium Value Calculation | Calculates a premium amount for a person. | " | Acty_Base_Rt_Id or Vrbl_Rt_Id or Actl_Prem_Id or Cvg_Amt_Calc_Mthd_Id; Value of the Acty_Base_Rt.Val, Min Val, Max Val | Amount | Premium amount is equal to .05 times Coverage amount less 50,000 |
| Matching Amount Calculation | Calculates the matching amount. | " | " | Amount | People that work at Division A get matching amount of 5%. All others get 4% matching amount. |
| Minimum Coverage Amount Calculation | Calculates a minimum coverage or benefit amount for a person. | " | " | Amount | If in Benefits Group A, minimum amount is $100. If in Benefits Group B, minimum amount is $200. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Maximum Coverage Amount Calculation | Calculates a maximum coverage or benefit amount for a person. | " | " | Amount | Coverage maximum amount is equal to coverage amount for current enrollment; if no current enrollment, then maximum is $100,000. |
| Period to Date Amount Calculation | Determines the maximum period to date amount a person may have for a particular activity rate. | " | | Amount | |
| Coverage Amount Calculation | Calculates a coverage or benefit amount for a person. | " | | Amount | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Age Determination Date | Determines the date from which age will be calculated (e.g., the first day of the next plan year). | " | | Date | If person is in an Annual Enrollment event then return 10/1/99. If person is becoming eligible as a result of a new hire, rehire, return from leave or a change from part-time to full-time then return the event occurred on date. |
| Hours Worked Determination Date | Determines the date from which hours worked will be calculated (e.g., the first day of the next plan year). | " | | Date | If person is in an Annual Enrollment event then return 10/1/99. If person is becoming eligible as a result of a new hire, rehire, return from leave or a change from part-time to full-time then return the event occurred on date. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Length of Service Date to Use | Determines the low date from which LOS will be calculated (e. g., original hire date or rehire date). | " | | Date | Return Start Date from the person's Absence Attendance row in order to determine the elapsed time that a person has been on a Leave of Absence. |
| Length of Service Determination Date | Determines the high date for which LOS will be calculated (e. g., the first day of the following month). | " | | Date | Return Start Date from the person's Absence Attendance row. |
| Compensation Determination Date | Determines the high date from which hours compensation will be calculated (e. g., the first day of the next plan year). | " | | Date | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Action Type Due Date | Determines the date on which the Action Type must be completed (e. g. 90 days from the Life Event Creation Date). | " | | Date | If person is currently enrolled, 60 days from life event creation date. If person is not currently enrolled, 30 days from life event creation date. |
| Participation Eligibility Start Date | Determines when eligibility for a person should start. | " | | Date | Add 6 months to the event date and return. |
| Participation Eligibility End Date | Determines when eligibility for a person should end. | " | | Date | If Organization on current assignment = A, return date equal to Event Date; otherwise, return date equal to Start of Following Month after event date. |
| Enrollment Coverage Start Date | Determines when enrollment coverage for a participant should start. | " | | Date | Coverage start date based on how long you have been absent from a plan. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Enrollment End | Determines when enrollment coverage for a participant should end. | " | | Date | If Calculated LOS is less than 5 years, then end of current month. If Calculated LOS is greater than or equal to 5 years, then end of 6 months after event date. |
| Dependent Coverage Start Date | Determines when coverage for a dependent should start. | " | | Date | If notified of birth within 31 days of event, Coverage Start Date is Date of Birth. Otherwise, coverage start date is date of notification. |
| Dependent Coverage End Date | Determines when coverage for a dependent should end. | " | | Date | If Contact Relationship Type = Spouse, coverage ends on date of event; if Contact Relationship Type = Child, coverage ends on end of the month following event. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Rate Start Date | Determines when a rate for an enrollment result for a participant should start. | " | | Date | Rate starts on the date after the rate end date of the current enrollment. |
| Rate End Date | Determines when a rate for an enrollment result for a participant should end. | " | | Date | Rate ends on the day before the person's next pay period. |
| Participation Eligibility | Determines whether a person is eligible for the associated compensation object. (This is the rule used in the eligibility profile rule entities.) | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id, Balance_Date | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Dependent Eligibility | Determines whether a person is eligible to be covered by a participant in a compensation object. | " | | Y/N | Dependent is eligible if Employee Work Location is 001 and Dependent is under age 21 or if Employee Work Location is not 001 and Dependent is under age 25. |
| Enrollment Opportunity | Determines whether the compensation object should be an electable choice for the person. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | One of the BEN_ENRT_MTHD values: A or E; or N | As a result of a transfer event, the participant can only enroll in an HMO if they were previously enrolled in an HMO and that HMO is no longer available to them in their new location. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| To Be Sent Date | Determines the date on which the communication should be sent to the person. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id, Cm_Typ_Id | | Date | If participant's organization is equal to {org1} or {org2} then return start of enrollment period - 14 days. If participant's organization is not equal to {org1} or {org2} then return start of enrollment period - 7 days. |
| Rounding | Rounds a number to the specified place or decimal. | None | | Amount | Amount to be rounded - $250 rounded to the next $500 |
| Percent Rounding | Rounds a percent to the specified place or decimal. | | | Amount | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Automatic Enrollment Method | Determines the conditions under which a person should be automatically enrolled in a compensation object. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | One of the BEN_ENRT_MTHD values: A or E; or N | Reinstate active benefits if rehired in the same plan year. |
| Deduction Schedule | Determines the deduction schedule to be used for this person. | " | | One of the BEN_DED_SCHED values | If bargaining unit code is not null return "Second Period In Month" else return "Every Pay Period". |
| Payment Schedule | Determines the payment schedule to be used for this person. | " | | One of the BEN_PYMT_SCHED values | If bargaining unit code is not null return "Second Period In Month" else return "Every Pay Period". |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Default to Assign Pending Action | Determines the applicable default option or benefit to assign to the person when his or her choice has been suspended. | " | | One of the BEN_DFLT_TO_ASN_PNDG_CTFN values | Reinstate the active benefits that a person had at the time that the person was previously active. (NOTE: cannot reinstate most recent benefits, as person could have some other elections, such as COBRA). |
| Enrollment Certification Required | Determines the conditions under which a person must provide certification in order to enroll or elect a particular plan or option in plan, or benefit. | " | | Y/N | Participant does not need to provide certification if they were enrolled in the compensation object and had provided certification within the past 12 months. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Dependent Certification Required | Determines the conditions under which a person must provide certification for his or her designated dependents. | " | | Y/N | |
| Beneficiary Certification Required | Determines the conditions under which a person must provide certification for his or her designated beneficiaries. | " | | Y/N | |
| Waive Certification Required | Determines the conditions under which a person must provide certification when he or she waives participation. | " | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Inspection Required | Determines whether inspection of the communication is required. If so, the "Inspection Flag" is set to 'yes' for this person's communication. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id, Cm_Typ_Id | | Y/N | If participant's division = "Corporate" or if participant's HCE indicator = "Y" return "YES". |
| Communication Appropriate | Determines for this communication and trigger, whether the communication should be sent; restricts to whom to send. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Ler_Id, Cm_Typ_Id | | Y/N | If person is in an Annual Enrollment event and today's date is within seven days of the enrollment period end date return "No". |
| Communication Type | Determines whether the communication should be sent. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Ler_Id, Cm_Typ_Id | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Mandatory Determination | Determines whether this option in plan should be assigned to a person (and not be optional) as part of the enrollment process. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | Y/N | |
| Postelection Edit | Performs edits on an enrollment result, e.g. is the spouse of the participant also enrolled; does the spouse of the participant work at the same company. | " | | Y/N and Text | If the participant elects a non-waive plan under the spouse life plan type, the participant must elect an equal or greater amount of participant life insurance. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Partial Month Proration Method | Determines which value to use: Date Earned, Pay Period End Date, or Payment Date. This tells the proration process which date to use when determining how many pay periods remain, and when the element entry should start. | " | | One of the Values of : BEN_PRTL_MO_DET_MTHD | If the participant is paid monthly and enrolls in a medical plan between the 8th and 15th of the month, then return 75% of the normal monthly price tag. |
| Partial Year Coverage Restriction | Determines the maximum coverage amount for partial years of coverage. Allows determination of values other than delivered values of: 1) Standard maximum as defined in the plan. 2) Prorate maximum based on the number of months remaining in the plan year. | " | | Amount | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Partial Month Effective Date Determination | Determines which value to use: Date Earned, Pay Period End Date, or Payment Date. This tells the proration process which date to use when determining how many pay periods remain, and when the element entry should start. | " | | One of the Values of : BEN_PRTL_MO_EFF_DT_DET | |
| Lack Certification Allow Reimbursement | Identifies cases where certification is waived. | " | | Y/N | |
| Compensation Object Selection | Determines which compensation objects are to be included for processing in a concurrent manager process. | " | | Y/N | Run the default process for the flex and nonflex programs only. Participation process selection mode: determine eligibility for all compensation objects that use derivable factors. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Person Selection | Determines which people are to be included for processing in a concurrent manager process. | " | | Y/N | Select all ex-participants who were working at a specific organization. |
| Verify Family Member | Determines whether the person has one or more contacts of a specific type or types, e.g. spouse, spouse and one child, more than one child. | " | | Y/N | |
| Five Percent Owner | Determines for this plan and regulation whether the person is a five percent owner as defined in the regulation. | " | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Highly Compensated | Determines for this plan and regulation whether the person is considered to be "highly compensated " as defined in the regulation. | " | | Y/N | |
| Key Employee | Determines for this plan and regulation whether the person is considered to be a "key employee" as defined in the regulation. | " | | Y/N | |
| Break in Service Value | Break in Service Value | " | | Amount | |
| Break in Service Determination | Determines whether a break in service has occurred and if this break should not be ignored. | " | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Contribution Nondiscriminatory | Compares individual employee pretax contribution amounts to total pretax contributions for highly and non-highly compensated persons. | " | | Y/N | |
| Coverage Nondiscriminatory | Compares the total number of persons eligible to participate in a plan minus the persons who are not eligible due to legislated allowable factors to the number actually participating. | " | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Extract Person Data Element | Specifies person or assignment information to be included as a data element item. This rule type can also return the results of a calculation performed on person or assignment information. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment_Id | | Text | |
| Extract Person Inclusion | Specifies person or assignment level inclusion or exclusion for system extract. | " | | Y/N | |
| Communication Usage | Determines whether a Communcation Usage requirement has been satisfied. If so, then a communication should be triggered for this usage. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Ler_Id, Cm_Typ_Id | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Default Enrollment | Determines whether this option in plan should be assigned to a person as part of the default enrollment process. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | | |
| Enrollment Period Start Date | Determines the date on which the enrollment period starts. | | | Date | |
| Enrollment Period End Date | Determines the date on which the enrollment period ends. | " | | Date | |
| Pop-Up Message | Determines whether a pop-up message displays and it what form. | " | | Y/N | For a participant who selects life insurance at 10x salary, display a warning message on the enrollment form if the participant's salary is less than 50k. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Collapse Life Event Resulting Occurred On Date | When life events are collapsed this rule determines the date to use for the resulting life event. | " | | Date | |
| Collapse Life Event Evaluation | Determines whether the life event should be collapsed and deleted, collapsed and voided, or neither. | " | | One of the values of: BEN_EVAL_DT: Collapse or Void; or leave as is | |
| Vested Value | Determine the vested percent for a person. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id, Balance_Date | | Percent | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Change Dependent Coverage | Used to determine if a dependent can be 'Added Only', 'Removed Only', Added and Removed', or 'Neither'. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | Y/N | |
| Evaluate Life Event | Determines if this life event is valid, or if the life event information needs to be changed, e.g. status, voided date, unprocessed date. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Ler Id | | Y/N, status code, unprocessed date, processed date. | |
| Maximum Waiting Period Date to Use | Determines the low date from which the maximum waiting period will be calculated (e.g., original hire date or rehire date). | " | | Date | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Waiting Period Value and UOM | Determines the waiting period and unit of measure to be applied to a person. | " | | Amount and UOM | |
| Maximum Period of Enrollment Value and UOM | Determines the maximum enrollment period and unit of measure for a compensation object. | " | | Amount and UOM | |
| Person Change Causes Life Event | Determines whether this life event is valid for a person based on the data that changed. | " | | Y/N | |
| Related Person Change Causes Life Event | Determines whether this life event is valid for a related person based on the data that changed. | " | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Participant Eligible to Rollover | Determines whether this person may roll over flex credits into a particular compensation object. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id, Balance_Date | | Y/N | |
| Payment Must Be Received | Determines whether a payment is missing or late. | " | | Y/N | |
| Life Event Reason Timeliness | Determines whether a life event has been reported in a timely manner. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | Y/N | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Required Period of Enrollment | Determines the earliest deenrollment date for a person's electable choice for a compensation object. | " | | Date | |
| Rate Lower Limit | Calculates an amount used to determine the lower limit value to which an activity rate or variable rate value is compared. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id, Balance_Date | | Amount | |
| Rate Upper Limit | Calculates an amount used to determine the upper limit value to which an activity rate or variable rate value is compared. | " | | Amount | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Coverage Lower Limit | Calculates an amount used to determine the lower limit for an activity or variable coverage or benefit. | " | | Amount | |
| Coverage Upper Limit | Calculates an amount used to determine the upper limit for an activity or variable coverage or benefit. | " | | Amount | |
| Premium Lower Limit | Calculates an amount used to determine the lower limit for an activity or variable premium. | " | | Amount | |
| Premium Upper Limit | Calculates an amount used to determine the upper limit for an activity or variable premium. | " | | Amount | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Maximum Period of Enrollment | Period of Enrollment Rule determines whether a person has been enrolled for the maximum length of time allowed for a plan or option in a plan. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | Y/N | |
| Maximum Period of Enrollment Determination Date | Determines the low date to be used when determining whether the person has been enrolled in a plan or option in plan for the maximum period of time. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | Date | |
| Partial Month Proration Value Calculation | Calculates a value for a partial month enrollment. | " | | Percent | |
| Variable Rate Add On Calculation | Calculates a new value when a variable rate result is used. | " | Result of the Variable Rate Calculation | Amount | Multiply the result by 102%. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Segment Costing Method | Determines how an amount is to be costed based on the segment in the COST KEY ALLOCATION KEY FLEXFIELD. | " | | Key Cost Allocation Flexfield Segment | |
| Extract Enrollment Data Element | Specifies enrollment information to be included as a data element item. This formula type can also return the results of a calculation performed on enrollment information. | Business_Group_Id, Effective_Date, Assignment Id, Pl_Id, Opt_Id, Ler_Id | | Text | |
| Maximum Credit Rollover Value | Determines the maximum amount a person may rollover to another plan or option in plan. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | Amount | |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Default Excess Credit Treatment | Determines how any excess credits are to be allocated. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | One of the values in BEN_DFLT_EXCS_TRTMT_CD | |
| Prorate Annual Election Value | Determines how a minimum or maximum annual election value is prorated. | Business_Group_Id, Effective_Date, Jurisdiction_Code, Organization_Id, Assignment Id, Pgm_Id, Pl_Typ_Id, Pl_Id, Opt_Id, Ler_Id | | Amount | |
| Extract Post Process | This formula type provides additional system extract processing and is called after all extraced records are written. | Business_Group_Id, Effective_Date | EXT_RSLT_ID (this is found on ben_ext_rslt and ben_ext_rslt_dtl tables). | Nothing. All processing should be via formula function. Commit will occur in calling program. | Allows you to insert intermittent totals, delete records, change sorting, format fields, update values, etc. |

| Formula Type | Description | Contexts | Input Values | Return Value | Sample Rule |
|---|---|---|---|---|---|
| Eligibility Access Calculation | Returns a person's value for a user-defined criteria so it can be evaluated in the eligibility determination process. | Business_Group_Id, Assignment Id, Date_Earned (life event occurred date or effective date), Organization_Id, pgm_id, pl_id, opt_id, ler_id, pl_typ_id | None | Criteria (and subcriteria) values | For a user-defined criteria of "Works more than 30 hours per week", rule returns Y or N according to the information stored on the person's work schedule. |
| Preferential Rate Calculation | For a given criteria rate definition, if an employee is found eligible for multiple rates by the Rate by Criteria API, the Preferential Rate Calculation formula can resolve which of these rates/ combination of rates must be paid to the employee. | BUSINESS_GROUP_ID, ASSIGNMENT_ID, DATE_EARNED, ORGANIZATION_ID, JURISDICTION_CODE | PQH_RBC_PERSON_ID, PQH_RBC_CRIT_RATE_DEFN_ID, MIN_1, MID_1, MAX_1, DFLT_1, MIN_2, MID_2, MAX_2, DFLT_2, MIN_3, MID_3, MAX_3, DFLT_3, MIN_4, MID_4, MAX_4, DFLT_4, MIN_5, MID_5, MAX_5, DFLT_5, | MIN, MID, MAX, DFLT | Preferred rate for people in the Sales organization is the maximum eligible rate; for people in the Marketing organization, preferred rate is the average of the eligible rates. |

## Benefit Uplift Formulas for Spain

Oracle HRMS enables you to override the benefit uplift amount for an employee set at the legal employer level while recording the employee's absence. You can use Oracle FastFormula to create the Benefit Uplift Daily Rate and the Benefit Uplift Duration Formula to calculate the amount based on employee's gross pay. See:Oracle FastFormula, page 1-1

**Benefit Uplift Daily Rate Formula**

Use the Benefit Uplift Daily Rate formula to calculate the benefit uplift amount for an employee depending on the daily rate of the gross pay. This formula returns a percentage rate of the gross pay applicable to the absence and the number of days paid at that percentage. For example, you can have an employer paying 100% of Gross Pay for the first 90 days of sickness and at 60% for 270 days. You attach this formula at the legal employer level to enable the application to calculate the benefit amount.

The formula must return the following value:

```
Return GROSS_PAY_PER_DAY
```

**Benefit Uplift Duration Formula**

Use the Benefit Duration formula to calculate the benefit uplift amount for an employee depending on the daily rate of gross pay. The formula returns the number of days paid at the percentage. This formula considers the absence start and end dates for an employee.

The formula must return the following values:

```
Return RATE1, VALUE1, RATE2, VALUE2, RATE3, VALUE3, RATE4, VALUE4,
RATE5, VALUE5, RATE6, VALUE6, RATE7, VALUE7, RATE8, VALUE8, RATE9,
VALUE9, RATE10, VALUE10
```

## Writing Formulas for Accrual Plans

Each accrual plan needs to be associated with two formulas: an accrual formula to calculate gross PTO entitlement to date and a Carry Over formula to be called by the carry over process at the end of the accrual term.

You can also associate a third formula to be called by BEE (Batch Element Entry) validation for entries to the absence element associated with the accrual plan. This Ineligibility formula checks whether an assignment is eligible to use accrued PTO. It must calculate the end of the ineligibility period in the same way as the Accrual formula for the plan. This formula is not required if you enter the ineligibility period for a plan in the Accrual Plan window.

Some formulas are seeded, see Seeded Accrual Type Formulas*, Oracle Time and Labor Implementation and User Guide*You can use these as supplied, edit them, or write your own formulas to provide the plan rules you require.

This topic explains:

- The formula types for formulas associated with accrual plans
- The required inputs and outputs for Accrual, Carry Over, and Ineligibility formulas
- Checks you must include in your Accrual formulas to avoid errors

For a sample Accrual formula and suggestions on how to change it to incorporate a whole range of plan rules, see: Sample Accrual Formula, page 1-157

### Formula Types

There are a number of formula types for formulas associated with accrual plans. You must define your formulas as the appropriate types or they will not be available for selection in the Accrual Plan window.

| Formula Type | Displays on list of values for . . . | Use for . . . |
|---|---|---|
| Accrual | Accrual Formula field | The top level formula that calculates PTO entitlement for a plan |
| Accrual Subformula | -- | Any formulas called by the top level formula, such as formulas for calculating the entitlement per period. |
| Accrual Carryover | Carryover Formula field | The formula to be called by the Carry Over process. |
| Accrual Ineligibility | Ineligibility Formula field | The formula to be called by BEE (if required) to specify whether an assignment is eligible to use accrued PTO. |

### Required Inputs and Outputs

If you write your own formulas for accrual plans, you **must** use the following INPUTS and RETURN statements. **Do not add extra lines to these statements.** You can use functions or database items to get extra inputs. The following values are available as contexts for all the accrual formula types (and therefore you do not need to retrieve them as inputs or database items):

- ASSIGNMENT_ID
- DATE_EARNED

- ACCRUAL_PLAN_ID
- BUSINESS_GROUP_ID
- PAYROLL_ID

Some formula functions have been defined specially for Accrual type formulas. They require some or all of these contexts and they return values need for the accrual calculation, such as total absences, accrual band, and period dates. See: Functions for Accrual Type Formulas, page 1-47. You can define and register any other functions you require.

### Inputs and Outputs for Accrual Formulas

```
INPUTS ARE
Calculation_Date (date)
ACCRUAL_START_DATE (date)
ACCRUAL_LATEST_BALANCE
/* Formula body */
RETURN total_accrued_pto, effective_start_date, effective_end_date,
accrual_end_date
```

| Input | Description |
|---|---|
| Calculation_Date | The date up to which accrual will be calculated |
| Accrual_Start_Date | The date to begin calculating accrual. If null, accruals are calculated from beginning of the accrual term. |
| Accrual_Latest_ Balance | The latest balance for the accrual term up to the day before Accrual_Start_Date. The latest balance is held in a payroll balance. |

**Note:** The Accrual_Start_Date and Accrual_Latest_Balance inputs are required only if your accrual plan uses a payroll balance to store gross accruals.

| Output | Description |
|---|---|
| total_accrued_pto | Gross accrued PTO this term |

| Output | Description |
|---|---|
| effective_start_date | Start date of accrual, which is normally the start of this accrual term, but may be plan enrollment date, hire date, adjusted service date, or other, depending on plan rules. |
| effective_end_date | Normally the calculation date, but should be the termination date if the employee has been terminated, or the end date of the plan element entry if the employee has left the plan. |
| accrual_end_date | This is an optional output. In the seeded formulas it is the end of the last full accrual period before the calculation date (because these formulas do not take account of partial accrual periods). |

### Inputs and Outputs for Carry Over Formula

```
INPUTS ARE
calculation_date (date),
accrual_term (text)
/* formula body */
RETURN max_carryover, effective_date, expiry_date, process
```

| Input | Description |
|---|---|
| calculation_date | Any date falling within an accrual term |
| accrual_term | 'PREVIOUS' or 'CURRENT' indicating whether to return the last date of the accrual term spanning calculation_date, or the accrual term previous to that spanning calculation date. |

| Output | Description |
|---|---|
| max_carryover | Maximum amount the employee can carry over, which may be dependent on an accrual band. |
| effective_date | The last date of an accrual term (either current or previous, as determined by the input). For example, this would be 31-DEC-YYYY for an accrual plan based on calendar years. |
| expiry_date | The date by which employees must use carried over PTO, otherwise they lose it. This output is optional. |
| process | Set to Yes by default. This means that the max_carryover amount is returned. If you set it to No, max_carryover is set to Null by the Carry Over process |

### Inputs and Outputs for Ineligibility Formula

```
INPUTS ARE
calculation_date (date),
/* formula body */
RETURN assignment_eligible
```

| Input | Description |
|---|---|
| calculation_date | The effective date of the element entry. |

| Output | Description |
|---|---|
| assignment_eligible | 'Y' or 'N'. If Y, BEE creates the entry to the absence element (assuming all other validation is successful). If N, BEE creates a warning on the batch line for the absence entry. |

#### Checks You Must Include In Your Accrual Formulas

You may notice that the seeded formulas contain statements to check a number of dates to see whether an employee is eligible to accrue any PTO. Be sure to include these checks in your formulas too.

### Termination Date

Check whether there is a termination date for the assignment. If the termination date is before the calculation date, calculate accrual as of the termination date. If your formula does not handle partial accrual periods, check whether the termination date is before the end of the first accrual period; if yes, set gross accrual to zero.

### Enrollment End Date

Check whether there is an end date for the assignment's enrollment in the plan. If the end date is before the calculation date, calculate accrual as of the end date. If your formula does not handle partial accrual periods, check whether the end date is before the end of the first accrual period; if yes, set gross accrual to zero.

### Calculation Date

If the calculation date is before the end of the first accrual period, set gross accrual to zero (unless your formula handles partial accrual periods).

### Hire Date

Check the employee's hire date or continuous service date. If your formula handles partial accrual periods, check that this date is before the calculation date, and if not, set the gross accrual to zero. If your formula does not handle partial periods, check that this date is before the start of the last full accrual period used in the current calculation. If the employee has not worked for a full accrual period before the calculation date, set the gross accrual to zero.

### Start Date for New Plan Participants

Check when the employee should start to accrue time. This is typically the date of enrollment in the plan or (if your formula does not handle partial accrual periods) the first period starting on or after the date of enrollment in the plan. If this date (or period) is after the calculation date (or period), set the accrual to zero.

> **Note:** The seeded and sample formulas also show how to incorporate other start dates in your plan, such as six months after hire date, or start of calendar year after hire date.

### Ineligibility Period

Check any ineligibility period (which is a time when a new participant accrues time but cannot use it, so it does not appear credited to him or her until the end of the period). If the eligibility period is still in force at the calculation date (or, if your formula does not handle partial accrual periods, on the end date of the last accrual period used in the calculation) set the gross accrual to zero.

### Inactive Assignment

Check whether the employee's assignment has been active throughout the period for which you are calculating accruals. Depending on your plan rules, your employees might not accrue time when their assignments are inactive, or they might accrue time at a reduced rate. You can use the function GET_ASG_INACTIVE_DAYS to check the assignment status on each day from period start date to period end date and return the number of inactive working days.

## Writing Formulas To Calculate Absence Duration

You can write a formula to calculate absence duration automatically when a user enters an absence start and end date, or time. Your localization team may have written a formula, which the system will use by default. However, if you need a configured formula to take account of special work hours or shift patterns, you can create a new formula that will override the supplied one. You can create one formula for each Business Group.

Your formula must be called: BG_ABSENCE_DURATION. You must select the formula type QuickPaint.

The formula inputs must be:

- days_or_hours (units for the duration: D or H)
- time_start, time_end
- date_start, date_end

The formula outputs must be:

- duration (the calculated value or FAILED)
- invalid_msg (optional - an error message name)

Use the supplied example formula (TEMPLATE_ABSENCE_DURATION) as the basis for your formulas.

## Writing Formulas to Calculate Eligibility for a Collective Agreement Entitlement

You can write a formula to be used to calculate whether a person is eligible to receive a collective agreement entitlement. This can be used when defining an eligibility profile to be used in conjunction with a collective agreement, instead of selecting criteria elements. You select the formula as a rule when defining the eligibility profile.

Your formula name can be whatever you like. You must select the formula type CAGR.

There are no formula inputs for this formula type.

The formula outputs depend on the category of the entitlement with which this formula is to be used. If the category is Absence, Payroll, or Assignment, then the output values are:

- Value
- Range From
- Range To

If the category is Pay Scale, then the output values are

- Parent_Spine_ID
- Step_ID
- From_Step_ID
- To_Step_ID
- Grade_Spine_ID

The contexts for this formula type are:

- BUSINESS_GROUP_ID
- PAYROLL_ID
- ASSIGNMENT_ID
- DATE_EARNED
- ORGANIZATION_ID
- TAX_UNIT_ID
- PERSON_ID

Use the supplied example formulas HR_CAGR_TEMPLATE (for Absence, Payroll, or Assignment categories) and HR_CAGR_PYS_TEMPLATE (for Pay Scale categories) as the basis for your formulas.

## Editing Assignment Set Formulas

Assignment set formulas do not normally need to be edited. If, however, you enter multiple criteria to define an assignment set, with conditions joined by AND or OR, you may want to edit the formula to change the brackets in the generated conditions. This changes the order in which the conditions are tested.

To view an assignment set formula, query it in the Formula window. The formula type is Assignment Set and the formula name is the same as the assignment set name. To edit an assignment set formula, make a copy, as for a QuickPaint formula.

## Writing Formulas for Templates

There are several ways you can use formulas to configure the people management templates:

- A Template Validation formula can check values entered in a field.
- A Template Information formula can specify information to be displayed from the right mouse menu when a user right-clicks in a field.
- A People Management Message formula can return a text string to display in the Assignment field on the Maintenance window and in the Data Organizer.
- A People Management Message formula can return message tokens that you can use in a notification message issued from template forms.

### Template Validation Formulas

If you use a formula to validate user entries in template fields, you must observe the following rules:

- Select the formula type Template Validation in the Formulas window.
- There can be up to five inputs, and they must be called item1, item2, item3, item4, and item5.
- The formula can return up to three outputs, which must be named as follows:
  - Status, which must have the value 's' if the validation was successful. Any other value is interpreted as an error.
  - Message, which is a text variable. The formula can return a message with

validation statuses of success, failure, or both.

- Item, which is the new value of the field that is being validated.

After creating the formula, you select it in the Validation Formula property for the field when you are setting up the template in the People Management Configurator. You can also enter up to five parameters to be passed to the formula, including the value you are validating. For example, you might enter the name of another field on the template if you want the formula to cross-validate the value in one field against another.

### Example

Here is a formula that validates the entry in the Organization field on the Maintenance window. It raises an error if the entry is not Corporate Finance.

```
/* Updateable Values Section */

/* Defaults Section */

/* Inputs Section */

INPUTS ARE item1 (text)
/* Main Body of Formula */

organization_name = item1

status = 's'

message = ' '

IF organization_name <> 'Corporate Finance' THEN

  (

message = 'Organization must be Corporate Finance'

  status = 'f'

   )

RETURN message, status
```

When you are setting up the template in the People Management Configurator, select the Organization (Maintain) item. Select the name of your formula as the Validation Formula property and select Organization (Maintain) as the Validation Formula Parameter 1 property. This passes and validates the value the user enters, not any codes or table identifiers to which it may be related.

### Template Information Formulas

If you write a formula to return additional information for a field, you must observe the following rules:

- Select the formula type Template Information in the Formulas window.
- There can be up to five inputs, and they must be called item1, item2, item3, item4, and item5.
- The formula can return one text output, which must be called Message. This is the information that is displayed when the user right clicks in the field and selects a prompt that you define in the right mouse menu.

After creating the formula, you select it in the Information Formula property for the field when you are setting up the template in the People Management Configurator. You can also enter up to five parameters to be passed to the formula. For example, if you were specifying an information formula for the job field, you might enter the name of the organization field so that the formula can return different additional job information depending on the organization.

### Example

Here is a formula that returns contact information to be displayed from the Supervisor field.

```
/* Updateable Values Section */

/* Defaults Section */

DEFAULT FOR sup_work_phone IS ' '

DEFAULT FOR sup_email_address IS ' '

/* Inputs Section */

/* Main Body of Formula */

message = 'Supervisor Contact Details||CHR(10)||'Telephone:

'||sup_work_phone||CHR(10)||'Email: '||sup_email_address

RETURN message
```

### People Management Message Formula for Assignment Field

Applicant and employee assignments are listed by name in the Data Organizer in the template Summary window and in the Assignment field on the Maintenance window. Since assignments do not have names, you can choose what assignment information is displayed as a name to help your users select the assignments they want to work with.

By default, assignments are identified as <job>.<organization>, such as Senior Manager. Engineering. However, you can choose any other database items to display.

If you want to override the default, write a formula (of type People Management Message) called ASSIGNMENT_NAME. This formula must return a text string. If there is no formula called ASSIGNMENT_NAME on the database, the system uses the predefined formula QH_ASSIGNMENT_NAME.

The QH_ASSIGNMENT_NAME formula is as follows:

```
/* Updateable Values Section */

/* Defaults Section */

DEFAULT FOR asg_job IS ' '

DEFAULT FOR asg_org IS ' '

/* Inputs Section */

/* Main Body of Formula */

assignment_name = asg_job||'.'||asg_org

RETURN assignment_name
```

#### People Management Message Formulas for Message Tokens

If you write a formula to return messge tokens, you must observe the following rules:

- Select the formula type People Management Message in the Formulas window.
- The formula must have the same name as the notification message that will use the tokens.
- The formula can return up to five text outputs, which must be named FF1, FF2, FF3, FF4 and FF5.

#### Example

Suppose you want to include an employee's preferred name in a New Hire notification. The New Hire notification is called NEW_STARTER, so you create a formula of the same name that returns this name in the variable FF1:

```
/* Updateable Values Section */

/* Defaults Section */

DEFAULT FOR per_known_as IS ' '

/* Inputs Section */
```

```
/* Main Body of Formula */

FF1 = per_known_as

RETURN FF1
```

Then you edit the New Hire notification to include the FF1 variable. For example:

Please note, that we have a new employee as of &HIRE_DATE.

They are &FULL_NAME (&EMPLOYEE_NUMBER).

Known as: &FF1.

Their new job details are:

Position: &POSITION

Job: &JOB

Organization: &ORGANIZATION

Location: &LOCATION

## Writing Proration Formulas

When the payroll run encounters an event (such as a grade change) that you have defined as a proration event for the element being processed, it creates two run results for the element--one for the payroll period up to the day before the event, and one from the date of the event to the end of the period. You must define a formula to handle this proration processing for the element. There are two ways to do this:

- Edit the element's Oracle Payroll formula so that it can handle proration, or
- Create an additional formula to run after the Oracle Payroll formula only in periods when a proration event is encountered. You select this formula in the Proration Formula field on the Proration tab of the Element window.

Using a separate proration formula has the advantage that proration takes place even when you enter a pay value directly on the element entry. Embedding the proration calculation in the Oracle Payroll formula avoids the overhead of calling the second formula in periods when proration events occur.

If you want to write a proration formula, you must follow these rules:

- Select the formula type **Payroll Run Proration**.
- The formula inputs can be:
  - any of the element input values
  - prorate_start (DATE)

    - prorate_end (DATE)

- The formula outputs can be:
    - any of the element input values

  Your localization team may have created example formulas that you can use as the basis for your own formulas.

## Writing Formulas for EEO Employment Categories

Use Oracle FastFormula to map your employment categories to the output values required for EEO4 and EEO5 reports. The EEO reports pick the list of employment categories from the formula.

Your formula must have the name PQH_EMPLOYMENT_CATEGORY, and the formula type must be Element Input Validation. If you want to use this formula, you must define it for each business group.

If you do not create a formula for employment categories, the default employment category list is used:

- FR - Full-time regulars
- FT - Full-time temps
- PR - Part-time regulars
- PT - Part-time temps

The formula output values are:

- Full_Time_Regulars
- Full_Time_Temps
- Part_Time_Regulars
- Part_Time_Temps

In the example below, employees with employment categories FR and FRD are treated as full-time regulars; those with employment categories FT, FTL, and FTJ are treated as full-time temporaries; those with PR are treated as as part-time regulars; and those with PT are treated as part-time temporaries.

```
/*Comma separated list of the employment categories with no spaces in
between */

full_time_regulars = 'FR,FRD'
```

```
full_time_temps = 'FT,FTL,FTJ'

part_time_regulars = 'PR'

part_time_temps = 'PT'

RETURN full_time_regulars, full_time_temps, part_time_regulars,
part_time_temps;
```

## Writing Formulas for Person Number Generation

When automatic local person numbering is in effect, Oracle HRMS allocates numbers for a person type (employees, contingent workers, or applicants) from a number sequence that is specific to the business group. When global person numbering is in effect, Oracle HRMS allocates numbers for a person type from a single global sequence across all business groups.

You can write a formula of type **Person Number Generation** to generate a *global* custom number sequence in place of the default local or global sequence. Once you have defined and validated a formula, Oracle HRMS executes the formula whenever automatic person numbering is active and a person number is required.

### Formula Names and Parameters

The formula names for the Person Number Generation type are:

- EMP_NUMBER_GENERATION (for employee numbers)
- APL_NUMBER_GENERATION (for applicant numbers)
- CWK_NUMBER_GENERATION (for contingent worker numbers)

You can define only one formula for each person type.

> **Important:** You must define person number generation formulas in the Setup business group, and they must have the names shown here. The formulas have no effect if you define them in any other business group or if you do not use the specified names.

The formula context value is Business Group ID.

The formula inputs are:

- Legislation code
- Person type
- Person number

- Party ID
- Person ID
- Date of birth
- Start date (which is the hire date for employees, the latest start date for applicants, and the placement start date for contingent workers)
- National identifier

The person number and person ID parameters are null when you create a new person record and nonnull when you update an existing person record. Although you cannot change a person number when you update a person record, Oracle HRMS checks that the number is valid. If the number is not valid (for example, if it is null), Oracle HRMS may execute the person number generation formula.

The formula outputs are:

- Next person number
- Completion message (for example, an error message)

The following general rules apply to person number formulas:

- Gaps in the number sequence are valid.
- Oracle HRMS checks that numbers from a custom sequence are unique in the business group.
- A custom number sequence applies to all business groups. However, you can write a formula that works differently in each business group.
- You can manage other business needs in the logic of the formula. For example, you may want employees to keep their employee numbers when they transfer to different business groups.

#### HR: Use Global Person Numbering Option

To use the default global person number sequence for a person type, you run the process "Change automatic person number generation to global sequencing," for the person type (Applicant, Contingent Worker, or Employee). This process sets the appropriate user profile option (HR: Use Global Applicant Numbering, HR: Use Global Contingent Worker Numbering, or HR: Use Global Employee Numbering) to Yes.

You do not need to run this process if you define a valid custom formula. The HR: Use Global Person Numbering options control the default global person number sequences only.

If you both define a valid formula for generating person numbers *and* run the process

"Change automatic person number generation to global sequencing," Oracle HRMS writes this message to the log: "A valid custom formula for generating person numbers exists. The default global number sequence will not be used." However, if the custom formula becomes invalid or you delete it, Oracle HRMS switches to the default global person number sequence because the profile option is set to Yes. Otherwise, Oracle HRMS switches to the default local sequence.

See Writing or Editing a Formula, page 1-146

## Writing Formulas for Rating Competencies and Objectives

During an appraisal, you can:

- Allocate performance, proficiency, and weighting values to individual competencies. HRMS supplies standard formulas that calculate a competency line score (the overall rating for an individual competency) based on specified combinations of each appraisal participant's performance, proficiency, and weighting values. Alternatively, you can define a formula to calculate competency line scores. This formula is of type Appraisal Competency Line Scoring.

- Allocate performance ratings to individual objectives. You can select each rating from a predefined performance-rating scale that you identify in the objective assessment template. Alternatively, you can define a formula to calculate ratings for individual objectives. This formula is of type Appraisal Objective Line Scoring.

- Allocate an overall rating to the appraisee. You select this value from the final-rating scale identified in the appraisal template. Alternatively, the application can set this value automatically by calculating a total score for the appraisal based on the final scores for competencies and objectives. This value appears as the appraisee's suggested overall rating in the Final Ratings page of the appraisal. To produce this value, you write a formula of type Appraisal Total Scoring.

For more information about how the application calculates competency and objective scores, see Assessing Competencies and Objectives, *Oracle Performance Management Implementation and User Guide*

#### Appraisal Objective Line Scoring

The formula context values are:

- BUSINESS_GROUP_ID

- ASSIGNMENT_ID

- ORGANIZATION_ID

- PERSON_ID

- DATE_EARNED (effective date)

The formula input values are:

- performance (number)
- weighting (number)
- line_object_id (number)
- appraisal_id (number)
- appr_template_id (number)
- appr_system_type (text)
- appr_type (text)

The formula output value is line_score (number).

HRMS provides sample formulas PERF and PERF_X_WEIGHTING. See: Sample Appraisal Objective Line Scoring Formulas, page 1-201

#### Appraisal Competency Line Scoring

The formula context values are:

- BUSINESS_GROUP_ID
- ASSIGNMENT_ID
- ORGANIZATION_ID
- PERSON_ID
- DATE_EARNED (effective date)

The formula input values are:

- performance (number)
- proficiency (number)
- weighting (number)
- line_object_id (number)
- appraisal_id (number)
- appr_template_id (number)

- appr_system_type (text)
- appr_type (text)

The formula output value is line_score (number).

HRMS provides sample formulas PERF_X_WEIGHTING, PERF_X_PROF, and PROF_X_WEIGHTING. See: Sample Appraisal Competency Line Scoring Formulas, page 1-205

#### Appraisal Total Scoring

The formula context values are:

- BUSINESS_GROUP_ID
- ASSIGNMENT_ID
- ORGANIZATION_ID
- PERSON_ID
- DATE_EARNED (effective date)

The formula input values are:

- competency_score (number)
- objective_score (number)
- appraisal_id (number)
- appr_template_id (number)
- appr_system_type (text)
- appr_type (text)

The formula output value is final_rating (number).

Note that the Appraisal Total Score formula must return a rating level ID associated with a level from the final-rating scale identified in the appraisal template. The application generates rating level IDs automatically. For simplicity, the sample Appraisal Total Score formulas return a level value (such as 1, 2, or 3) rather than the required rating level ID. To obtain the rating level ID:

1. In the Rating Scales window, query the final-rating scale identified in the appraisal template.
2. Select the first level value (for example, 1).

3. From the Help menu, select Diagnostics, then Examine. The Examine Field and Variable Values window appears.

4. Select the Field value RATING_LEVEL_ID. The rating level ID appears in the Value field.

Repeat this procedure for each level in the rating scale.

HRMS provides sample formulas AVG_COMP_AND_OBJ and SUM_COMP_AND_OBJ. See: Sample Appraisal Total Scoring Formulas, page 1-211

## Writing or Editing a Formula

You can choose one of the following methods to write and edit formulas:

- FastFormula Assistant, which you can use to make changes without keeping date-tracked records of your formulas or if you are new to formulas

- Formula window, which you can use to make changes and keep the date-tracked history of your formulas

### To write or edit a formula using the FastFormula Assistant:

1. Navigate to the Total Compensation : Basic menu and select the FastFormula Assistant option that you require. The Global FastFormula Assistant option provides access to formulas across all business groups whereas the FastFormula Assistant option provides access to all formulas in the current business group only.

2. Follow the instructions on the FastFormula Assistant pages to find out how to write and edit your formulas.

### To write or edit a formula using the Formula window:

1. Set your effective date to the date when you want to begin using the formula.

2. To write a new formula, enter the formula's name and select a type. To edit an existing formula, query its name.

   **Note:** You cannot create formulas that exceed 64K in the Formula window. You must split longer formulas into two.

3. Choose the Edit button to open a blank Edit Formula window where you can write a new formula, or modify an existing one.

4. If you want to select database items, choose the Show Items button to display the Database Items window and run a query. Copy and paste items from this window

to the Edit Formula window.

5. If you want your formula to use input value names that have been translated from English, choose the Input Values button. The Input Values window always displays translated names if translated names exist. To include translated input value names in your formula text, select the translated name and then choose the Paste Input button to paste to your formula text.

6. When you finish writing or editing the formula, choose the Verify button to compile it.

   This process identifies any syntax errors in your formula.

   > **Note:** Once you have compiled any formula, new functions or changes to existing functions made after the first time you compile, are not used. You must logout of Oracle HRMS and login again. You can now compile your formula and the new functions and/or changes to existing functions will be included.

7. When the formula is verified successfully, save it.

8. Your next step depends on the type of formula:

   - If the formula is of type Oracle Payroll, you must associate it with an element in the Formula Result Rules window.
   - If the formula is of type Element Skip, you select it in the Skip Rule field of the Element window.
   - If the formula is of type Element Input Validation, you select it in the Formula field of the Input Values window when you are defining an element.
   - If the formula is of type User Table Validation, you select it in the Formula field of the Columns window when you are defining a user table structure.
   - If the formula is of type Accrual, Accrual Carryover, or Accrual Ineligibility, you select it in the Accrual Plan window.
   - If the formula is of type Accrual Subformula, you call it from another formula of type Accrual.
   - If the formula is used for benefits administration, you select the formula in the Rules field of the appropriate benefits window.

   Test your formula in the situation you intend to use it (such as, in a test payroll run) to ensure it contains no logical errors.

## Writing Payroll Formulas for Elements

If you have defined your own payroll elements, you can write formulas to calculate earnings and deductions.

For guidance on writing efficient payroll calculation formulas, see: Formula Writing Techniques, page 1-16. For important information about using element input values in payroll formulas, see: Input Values in Payroll Formulas, page 1-24.

### To define elements and their formulas:

1. Design your element and how it will be calculated.
2. Write any formulas required to validate input values (formula type = Element Input Validation).
3. Write a formula, if required, to define the rules for skipping the element during payroll processing (formula type = Element Skip).
4. Define the element, referencing any formulas written in steps 2, page 1-148 and 3, page 1-148.
5. Write the formula or formulas for calculating the run results (formula type = Oracle Payroll).
6. Associate each Oracle Payroll type formula with the element in the Formula Result Rules window, and specify what happens to the formula results for this element.

> **Note:** You can associate several formulas with a single element, each one for use with a different employee assignment status. You can also use the same formula for more than one element. In this case, if the formula references pay or input values (through the Inputs statement), each element must have pay and input values with the same names.

## Writing Formulas for Element Skip Rules

If your payroll policies require conditional processing of an element, you can write a formula to define when the run should process the element and when it should skip it. For example, your formula could specify:

- process the Union Fees element every run unless the Union_Fees_Paid balance is greater than 10 000.

Your skip rule formula must be consistent with other processing rules defined for the

element, such as frequency rules, which determine in which period the element is normally processed. Notice that a skip rule cannot contravene any other processing rules in place for the element.

You can associate only one element skip rule formula with each element. You must write and validate the formula before you define the element so that you can select the formula from a list on the Element window.

> **Note:** Mexican elements do not employ skip rules.

### To write a formula defining a skip rule:

1. Select formula type Element Skip in the Formulas window.
2. Use as many input values as you require. The formula must set and return a local variable of type text, and this variable must be called skip_flag.

   If the returned value of this variable begins with the letter y (such as 'Yes'), all processing for the element is skipped. Otherwise the element processes as normal.

   The following example of a skip rule formula defines that the Union Fees element is not processed if the Union_Fees_Paid balance is greater than 10 000:

```
IF Union_Fees_Paid > 10000

  THEN

 skip_flag = 'yes'

  ELSE

 skip_flag = 'no'

RETURN skip_flag
```

## Copying and Adding Features to a QuickPaint Formula

When you save a QuickPaint Report definition, a formula is generated automatically. Formulas generated from QuickPaint do not include conditional logic or calculations. You may want to add these features, for example to sum up compensation elements or to report different data for different assignments.

> **Important:** If you want to add features to a generated QuickPaint formula, you must copy the formula and edit the copy. If you edit the original, your edits will be overwritten if the formula is regenerated from the QuickPaint definition.

### To make a copy of a QuickPaint formula:

1. In the Formula window, query your QuickPaint formula. It has the same name as your QuickPaint report.
2. Choose the Edit button. Select and copy the formula in the Edit Formula window.
3. Choose New Record from the Edit menu.
4. Enter a name for your edited copy and select the type QuickPaint.
5. Paste the text of the QuickPaint formula into the Edit Formula window.
6. Save your work.

## Writing Formulas for Validation

You can use Oracle FastFormula to validate user entries into the element input values, and to user tables that you define.

### To write a formula for validation purposes:

1. Write and validate the formula.

   You must do this before you define the element or table, so that you can select the formula from a list in the Element window or Columns window.
2. Define the element or table.
3. Select formula type Element Input Validation or User Table Validation in the Formulas window.

   #### Rules to Observe

   - There must be one input value, of type text, and it must be called entry_value.
   - The formula must set and return a local variable giving the status of the validation (success or error). This variable must be called formula_status and have the value 's' (success) or 'e' (error).
   - Optionally, the formula can also return a text variable giving an explanatory message. The returned message variable must be called formula_message and can contain any text. It can be returned with both successful and unsuccessful statuses.
   - The formula must not return any other results.

   For an element input value validation formula, you must also observe the following

rules:

- You cannot use the element's other pay and input values in the formula.
- You cannot return a value to another pay or input value.

All entry values are stored in the database as text items. Therefore, if you want to validate an entry value as a date or number, you must use Oracle FastFormula's conversion function to convert the text into a date or number type variable. For example:

```
TO_NUM (entry_value)

TO_DATE(entry_value,'DD-MON-YYYY')
```

## Writing Formulas for Default Assignment Costing

You can write a formula to specify the cost allocation key flexfields and their proportions that you use for default assignment costing.

> **Note:** As a prerequisite, you must set the profile option HR: Default Assignment Costing to Yes.

### To write a formula for default assignment costing:

1. Open the Formula window.
2. Required: enter the formula name exactly as below:
   - PER_DFLT_ASG_COST_ALLOCATION
3. Select a formula Type of Element Input Validation.
4. Enter a Description of the formula.
5. Choose the Edit button.
6. Write the formula. Oracle HRMS provides the following inputs:
   - assignment_id (number)
   - business_group_id (number)
   - position_id (number)
   - effective_date (date)

**Note:** Because no context is set, do not use FastFormula Database Items. Instead, use functions for complex formulas.

Use the sample code below as a guideline in writing your own formula.

```
USE_FORMULA = 'Y'

COST_ALLOCATION_KEYFLEX_ID1 = 101

COST_ALLOCATION_KEYFLEX_ID2 = 102

COST_ALLOCATION_KEYFLEX_ID3 = 103

PROPORTION1 = .5

PROPORTION2 = .3

PROPORTION3 = .2

RETURN USE_FORMULA,

COST_ALLOCATION_KEYFLEX_ID1, PROPORTION1,

COST_ALLOCATION_KEYFLEX_ID2, PROPORTION2,

COST_ALLOCATION_KEYFLEX_ID3, PROPORTION3
```

If the application does not return a value for USE_FORMULA; or, if the value for USE_FORMULA is not equal to 'Y', the default assignment costing will not use the accounts and proportions returned by the formula. Instead, default assignment costing will be calculated based on the position control budget for the business group, if one has been defined.

If USE_FORMULA = 'Y',the application creates the default assignment costing with the cost allocation flexfield id (COST_ALLOCATION_KEYFLEX_ID%) and the related proportion (PROPORTION%).

- If the COST_ALLOCATION_KEYFLEX_ID% does not contain a valid cost allocation id for the business group,then that entry will be ignored.
- If the PROPORTION% is greater than 1, it will be treated as 1.
- If the PROPORTION% is less than 0, the entry will be ignored.

## Writing Formulas for Rate By Criteria Calculations

Rate By Criteria (RBC) uses four formula types to help determine eligibility and rate

definitions. You call a seeded FastFormula function, RBC_Rate_Retrieval, to pay an employee through Oracle Payroll.

The function evaluates the eligible rate for an employee by processing the rate matrixes in the employee's business group. If this function is called in the Payroll formula attached to the element, it returns the Rate By Criteria rate for which the employee is eligible. The element must be associated with a criteria rate definition.

**Formula Types**

RBC specifically uses the four formula types below. For further details on each type, see the help topic Total Compensation Formula Types, page 1-93 and the *OAB FastFormula Reference Guide* on My Oracle Support, Note ID: 218059.1

| Formula Type | Module | Usage |
|---|---|---|
| Eligibility Access Calculation | Eligibility Criteria | Aids in setup of user-defined criteria and is evaluated in the eligibility determination process to retrieve the person information value for your criteria. |
| Compensation Calculation | Criteria Rate Definition | Determines a compensation amount to be used to pay a person from Rate by Criteria. |
| Rounding | Criteria Rate Definition | Rounds a number to a specified place or decimal. |
| Preferential Rate Calculation | Criteria Rate Definition | For a given criteria rate definition, if an employee is found eligible for multiple rates by the Rate by Criteria API, the Preferential Rate Calculation formula can resolve which of these rates/ combination of rates the employee receives. |

**The RBC Rate Retrieval Formula Function**

To call the seeded RBC_Rate_Retrieval function from a formula, declare a local variable within the Element FF text:

<*local variable1*> = RBC_Rate_Retrieval()

The function includes predefined contexts, so you include no additional parameters.

## Defining Global Values

Use global values to store information that does not change often, but you refer to frequently, as for example Company Name, or company-wide percentages used in the calculation of certain bonuses. You can use global values as variables in formulas by simply referring to the global value by name.

You can never change a global value using a formula. You change global values in the Globals window. Global values are datetracked so you can make date effective changes ahead of time.

Global values are available to all formulas within a Business Group.

### To define a global value:

1. Set your effective date to the date when you want to begin using the global value.
2. Enter a name, data type (number, text, or date), and value. You can also enter a description.

## Bulk Compiling Formulas

Where a formula has more than one version, you can compile all the versions in one process using the Bulk Compile Formulas process. For example, you run this process when you upgrade your legislative information, which contains formulas. The Bulk Compile Formulas process automatically generates the Formula Wrapper.

### To bulk compile formulas:

1. Select Single Request in the Submit a New Request window.
2. In the Name field, select Bulk Compile Formulas and submit your request.

   You can define when you want to run this process using the schedule options.

   **Note:** If you make any changes to a function after you have compiled a formula that uses it, you need to recompile the formula for the changes to take effect.

## Generating the Formula Wrapper

To get the best performance for executing formulas from PLSQL you need to generate the Formula Wrapper. The Bulk Compile Formulas process automatically generates the Formula Wrapper.

**Note:** You do not need to generate the Formula Wrapper to test a formula.

### To generate the Formula Wrapper:

1. Select Single Request in the Submit a New Request window.

2. In the Name field, select Generate Formula Wrapper and submit your request.

   You can define when you want to run this process using the schedule options.

## Registering a Function

You register a new function by naming and defining it, then creating contexts and parameters for it. Contexts are environment values that do not get passed to the function.

Where a function requires a mixture of contexts (from the FF_CONTEXTS table) and parameters, the contexts should be listed first in the function header followed by the function parameters. Only the function parameters, however, need to be used to call the function from FastFormula.

For example, a function requires eight values: three contexts and five parameters. All eight values are listed in the function header but only the five parameters are used to call the function.

You register the class of the function as external. External functions are further PL/SQL functions in addition to the ones already delivered with FastFormula. External functions can use contexts and parameters.

### To register a new function for Oracle FastFormula:

1. Enter a unique name for the new function.
2. Select date, number or text as its data type.
3. Select external as the class of the function.
4. Enter an alias for the function name if you require an alternative name for it. You can also enter a description to explain what the function is for. The Alias and Description fields are both optional.
5. Enter the definition of the function. Use the format: <package name>.<function name>.
6. Save your entries.

### To enter context usage and parameter information:

1. Choose the Context Usages button.
2. In the Context Usages window, select as many context items as you require for the function. The data type for each context displays automatically.

**Note:** The functionality that calls FastFormula, that is, QuickPaint or Payroll Processing, determines what contexts FastFormula has access to from the FF_CONTEXTS table.

3. Save your entries. The sequence number of each context is entered automatically when you do this.
4. Close the Context Usages window and choose the Parameters button.
5. In the Parameters window, enter the parameters, or operands, you require to define the function. Type and class display automatically.
6. Check the Optional check box if you want the corresponding parameter to be optional.
7. Check the Continuing check box if you want the function to make more than one call to the parameter.

   **Note:** You cannot define a parameter as continuing unless you also make it optional. However, an optional parameter does not have to be continuing.

8. Save your entries.

# Sample Formulas

## Sample Accrual Formula

This topic suggests how you can implement a whole range of accrual plan rules in your Accrual formula. The suggestions are based on a simple formula, which is similar to the seeded PTO_SIMPLE_MULTIPLIER formula. The sample formula is for a plan with the following rules:

- An accrual term of one calendar year starting 01 January.
- Monthly accrual periods and a fixed accrual of 2 days per month.
- An accrual ceiling of 20 days, fixed within the formula.
- Accrual for new hires begins on whichever of these dates is the latest: hire date, plan enrollment date, or continuous service date (which can be entered as an input value when you enroll an employee in a plan).

The top level formula repeatedly calls another formula in a loop to calculate the accrual for each period. Both the top level formula (PTO_ONE_YEAR_MULTIPLIER) and the looping formula (PTO_PERIOD_ACCRUAL) are given below.

**Formula**

```
1.   /*-----------------------------------------------
NAME : PTO_ONE_YEAR_MULTIPLER
This formula calculates the dates between which
an assignment is to accrue time
-----------------------------------------------*/

2.   DEFAULT FOR ACP_CONTINUOUS_SERVICE_DATE IS
      '31-DEC-4712' (date)
3.   DEFAULT FOR ACP_TERMINATION_DATE IS
      '31-DEC-4712' (date)
4.   DEFAULT FOR ACP_ENROLLMENT_START_DATE IS
      '31-DEC-4712' (date)
5.   DEFAULT FOR ACP_SERVICE_START_DATE IS
      '31-DEC-4712' (date)

6.   INPUTS ARE
     Calculation_Date (date)

7.   E = SET_NUMBER('CEILING', 20)
8.   IF ASG_STATUS = 'Suspended' THEN
     (
      E = SET_NUMBER('ACCRUAL_RATE', 0)
     )
     ELSE
     (
      E = SET_NUMBER('ACCRUAL_RATE', 2)
     )

9.   Accruing_Frequency = 'M'   /* Month */
10.  Accruing_Multiplier = 1

11.  E = SET_TEXT('ACCRUING_FREQUENCY',
                  Accruing_Frequency)
12.  E = SET_NUMBER('ACCRUING_MULTIPLIER',
                    Accruing_multiplier)

13.  Beginning_Of_Calculation_Year = to_date
             ('0101'||to_char(Calculation_Date,'YYYY'),'DDMMYYYY')

14.  E = SET_DATE('BEGINNING_OF_CALCULATION_YEAR',
```

```
Beginning_Of_Calculation_Year)

15.  /*-----------------------------------------------
     Set the start and end dates of the first accrual
     period in the calculation year.
     -----------------------------------------------*/

16.  E = GET_PERIOD_DATES
                (Beginning_of_Calculation_Year,
                 Accruing_Frequency,
                 Beginning_Of_Calculation_Year,
                 Accruing_Multiplier)

17.  First_Period_SD = get_date('PERIOD_START_DATE')
18.  First_Period_ED = get_date('PERIOD_END_DATE')

19.  /*-------------------------------------------
     Set the Calculation_Date to the Termination Date
     if not null
     ------------------------------------------- */

20.  IF NOT (ACP_TERMINATION_DATE WAS DEFAULTED) THEN
     (
21.  IF (ACP_TERMINATION_DATE < Calculation_Date) THEN
       (
22.     Calculation_Date = ACP_TERMINATION_DATE
        )
      )

23.  /* -----------------------------------------------
     Get the last whole period prior to the
     Calculation Date and ensure that it is within the
     Year (if the Calculation Date is the End of a
     Period then use that period)
     ------------------------------------------- */

24.  E = GET_PERIOD_DATES(Calculation_Date,
```

```
                        Accruing_Frequency,
                        Beginning_of_Calculation_Year,
                        Accruing_Multiplier)

25.  Calculation_Period_SD = get_date('PERIOD_START_DATE')
26.  Calculation_Period_ED = get_date('PERIOD_END_DATE')

27.  IF (Calculation_Date <> Calculation_Period_ED)
     THEN
     (
28.     E =  GET_PERIOD_DATES
                 (ADD_DAYS(Calculation_Period_SD,-1),
                  Accruing_Frequency,
                  Beginning_of_Calculation_Year,
                  Accruing_Multiplier)
29.     Calculation_Period_SD = get_date('PERIOD_START_DATE')
30.     Calculation_Period_ED = get_date('PERIOD_END_DATE')
      )

31.  /* ------------------------------------------
      Set the Continuous Service Global Variable using
      the Continuous Service Date (if it was entered
      when the employee enrolled in the plan) and
      otherwise using hire date, whilst also ensuring
      that the continuous service date is before the
      Calculation Period.
      ------------------------------------------ */

32.  IF (ACP_CONTINUOUS_SERVICE_DATE WAS DEFAULTED)
     THEN
33.  (
       E = set_date('CONTINUOUS_SERVICE_DATE',
                    ACP_SERVICE_START_DATE)
     )
34.  ELSE
```

```
(
35.   E = set_date('CONTINUOUS_SERVICE_DATE',
                   ACP_CONTINUOUS_SERVICE_DATE)
    )

36.  Continuous_Service_Date = get_date('CONTINUOUS_SERVICE_DATE')

37.  /* ---------------------------------------------
     Determine the date on which PTO actually starts
     accruing based on Continuous Service Date, the
     Start Date of the Calculation Year, and plan
     Enrollment Start Date. Remember, we have already
     determined whether to use hire date or CSD at
     lines 32 to 35 above.
     ---------------------------------------------- */

38.   Actual_Accrual_Start_Date =
               greatest(ACP_ENROLLMENT_START_DATE,
                        Continuous_Service_Date,
                        First_Period_SD)

39.  /* ---------------------------------------------
     Determine the start and end date of the first
     accrual period to use in the accrual calculation.
     Get the start and end dates of the accrual period
     in which the Actual Accrual Start Date falls. If
     the Actual Accrual Start Date does not fall on
     the first day of this period, start accruing from
     the next period.
     ----------------------------------------------*/

40. IF Actual_Accrual_Start_Date > First_Period_SD
      THEN
      (
41.  E = GET_PERIOD_DATES(Actual_Accrual_Start_Date,
                          Accruing_Frequency,
```

```
                        Beginning_Of_Calculation_Year,
                        Accruing_Multiplier)
42.  Accrual_Start_Period_SD = get_date('PERIOD_START_DATE')
43.  Accrual_Start_Period_ED = get_date('PERIOD_END_DATE')
44.  IF Actual_Accrual_Start_Date > Accrual_Start_Period_SD THEN
       (
45.     E = GET_PERIOD_DATES
                  (add_days(Accrual_Start_Period_ED,1),
                   Accruing_Frequency,
                   Beginning_of_Calculation_Year,
                   Accruing_Multiplier)
46.     Accrual_Start_Period_SD = get_date('PERIOD_START_DATE')
47.     Accrual_Start_Period_ED = get_date('PERIOD_END_DATE')
       )
      )
48.  ELSE
      (
49.  Accrual_Start_Period_SD = First_Period_SD
50.  Accrual_Start_Period_ED = First_Period_ED
       )

51.  /* ------------------------------------------
      Now set up the information that will be used
      when looping through the periods
      ------------------------------------------ */

52.  IF Calculation_Period_ED >= Accrual_Start_Period_ED THEN
      (
53.    E = set_date('PERIOD_SD',Accrual_Start_Period_SD)
54.    E = set_date('PERIOD_ED',Accrual_Start_Period_ED)
55.    E = set_date('LAST_PERIOD_SD',Calculation_Period_SD)
56.    E = set_date('LAST_PERIOD_ED',Calculation_Period_ED)
57.    E = set_number('TOTAL_ACCRUED_PTO',0)
58.    E = LOOP_CONTROL('PTO_PERIOD_ACCRUAL')
```

```
59.    Total_Accrued_PTO = get_number('TOTAL_ACCRUED_PTO')
      )

60.  IF Accrual_Start_Period_SD > Calculation_Period_ED THEN
      (
61.    Accrual_Start_Period_SD = First_Period_SD
      )

62.  Effective_start_date = Accrual_Start_Period_SD
63.  Effective_end_date = Calculation_Date
64.  Accrual_end_date = Calculation_Period_ED

65.  RETURN Total_Accrued_PTO,
66.         Effective_start_date,
67.         Effective_end_date,
68.         Accrual_end_date
```

**Looping Formula**

```
1.   /* ------------------------------------------
     NAME : PTO_PERIOD_ACCRUAL
     This formula calculates the dates between which an
     assignment is to accrue time
     ------------------------------------------------*/

2.  /*-----------------------------------------------
     Get the global variable to be used in this formula
     ------------------------------------------------*/

3.  Continuous_Service_Date = get_date('CONTINUOUS_SERVICE_DATE')
4.  Total_Accrued_PTO = get_number('TOTAL_ACCRUED_PTO')
5.  Period_SD = get_date('PERIOD_SD')
6.  Period_ED = get_date('PERIOD_ED')
7.  Last_Period_SD = get_date('LAST_PERIOD_SD')
8.  Last_Period_ED = get_date('LAST_PERIOD_ED')
9.  Accrual_Rate = get_number('ACCRUAL_RATE')
10. Accruing_Frequency = get_text('ACCRUING_FREQUENCY')
11. Accruing_Multiplier = get_number('ACCRUING_MULTIPLIER')
12. Beginning_of_Calculation_Year =
                        get_date('BEGINNING_OF_CALCULATION_YEAR')
13. Ceiling = get_number('CEILING')

14.  /* ------------------------------------------
     Calculate the Amount Accrued this Period
     ------------------------------------------------*/

15.  Period_Accrued_PTO = Accrual_Rate

16.  /*-----------------------------------------------
     Calculate any absence or bought/sold time etc. to
     be accounted for in this period.
     ------------------------------------------------*/

17.  Absence = GET_ABSENCE(Period_ED,
                           Beginning_of_Calculation_Year)
```

```
18.  CarryOver = GET_CARRY_OVER(Period_ED,
                         Beginning_of_Calculation_Year)
19.  Other = GET_OTHER_NET_CONTRIBUTION(Period_ED,
                         Beginning_of_Calculation_Year)

20.  Period_Others = CarryOver + Other - Absence

21.  /* ---------------------------------------------
     Now establish whether the Accrual this period has
     gone over the ceiling if one exists
     ---------------------------------------------*/

22.  IF (Ceiling > 0) THEN
     (
23.    IF (Total_Accrued_PTO + Period_Accrued_PTO +
                Period_Others > Ceiling) THEN
       (
24.      Amount_Over_Ceiling = Total_Accrued_PTO +
           Period_Accrued_PTO + Period_Others - Ceiling
25.      IF (Amount_Over_Ceiling > Period_Accrued_PTO)
         THEN
         (
26.        Period_Accrued_PTO = 0
         )
27.      ELSE
         (
28.        Period_Accrued_PTO = Period_Accrued_PTO -
                                Amount_Over_Ceiling
         )
       )
     )

29.  /*---------------------------------------------
     Set the Running Total
     ---------------------------------------------*/
```

```
30.  E = set_number
           ('TOTAL_ACCRUED_PTO',Total_Accrued_PTO +
             Period_Accrued_PTO)

31.  /* ----------------------------------------------
     Establish whether the current period is the last
     one, if so end the processing, otherwise get the
     next period
     ----------------------------------------------*/

32.  IF Period_SD = Last_Period_SD THEN
     (
33.    Continue_Processing_Flag = 'N'
     )
34.  ELSE
     (
35.    E = GET_PERIOD_DATES(ADD_DAYS(Period_ED,1),
                            Accruing_Frequency,
                            Beginning_of_Calculation_Year,
                            Accruing_Multiplier)
36.    E = set_date('PERIOD_SD', get_date('PERIOD_START_DATE'))
37.    E = set_date('PERIOD_ED', get_date('PERIOD_END_DATE'))
38.    Continue_Processing_Flag = 'Y'
     )

39.  Return Continue_Processing_Flag
```

#### Changing the Length of the Accrual Periods

The accrual period is determined by the variables Accruing_Frequency and Accruing_Multiplier (lines 9 and 10 of the top level formula). You can set Accruing_Frequency to M (month), D (day) or W (week). For example, if the frequency is set to W and the multiplier set to 2, time is accrued every two weeks.

These examples use the calendar to measure the length of accrual periods. You can also use payroll periods. In this case, you do not need to set the variables Accruing_Frequency and Accruing_Multiplier. At lines 16, 24, 28, 41, and 45, replace the call to get_period_dates with a call to get_payroll_period. For example, replace lines 24 to 26 with:

```
E = GET_PAYROLL_PERIOD (ADD_DAYS(Calculation_Period_SD, -1))
CALCULATION_PERIOD_SD = get_date('PAYROLL_PERIOD_START_DATE')
CALCULATION_PERIOD_ED = get_date('PAYROLL_PERIOD_END_DATE')
```

#### Changing the Accrual Term Start Date

The accrual term start date is set to 01 January at line 13 of the sample top level formula. To use another fixed date (such as 01 June) replace this line with the following:

```
Beginning_Of_Calculation_Year = to_date
            ('0106'||to_char(Calculation_Date,'YYYY'),'DDMMYYYY')

If Beginning_Of_Calculation_Year > Calculation_Date then
(
  Beginning_Of_Calculation_Year =
             add_months(Beginning_Of_Calculation_Year, -12)
)
```

To start an employee's accrual term on his or her hire date anniversary, replace line 13 with:

```
Beginning_Of_Calculation_Year = to_date(
                 to_char(ACP_SERVICE_START_DATE,'DD-MM') ||
                 to_char(Calculation_Date,'YYYY'),'DDMMYYYY')

If Beginning_Of_Calculation_Year > Calculation_Date then
(
  Beginning_Of_Calculation_Year =
                 add_months(Beginning_Of_Calculation_Year, -12)
)
```

This example uses the ACP_SERVICE_START_DATE database item, although any substitute may be used.

#### Adding Start Rules for New Hires

In this sample formula, accrual for new hires begins on whichever of these dates is the latest: hire date, plan enrollment date, or continuous service date. However, you may want your top level accrual formula to check whether there is a start rule defined for the plan. The seeded PTO_PAYROLL_CALCULATION formula shows you how to do this.

In summary, the formula has to:

- Check which start rule was entered for the plan, and calculate the

First_Eligible_To_Accrue_Date accordingly.

- Get the first full accrual period following the First_Eligible_To_Accrue_Date.
- Check whether the Actual_Accrual_Start_Date is after the First_Eligible_To_Accrue_Date and, if not, set the Accrual_Start_Period start and end dates to the First_Eligible_To_Accrue_Period start and end dates. This affects lines 40 to 50 of the sample formula.

There are three seeded start rules: hire date, beginning of calendar year after hire date, and six months after hire date. If you need a different start rule, define it as a value for the Lookup Type US_ACCRUAL_START_TYPE. Then add a line to your formula, of this form:

```
IF (ACP_START = '<your new Lookup Value>') THEN
(First_Eligible_To_Accrue_Date = <calculation for start date> )
```

#### Basing the Accrual Amount on Time or Pay Elements

You need to create database items for the element input values that determine the accrual amount. Suppose plan participants accrue one hour for every 10 hours worked. You could include this rule in your formula as follows:

```
Accrual = ACP_HOURS_WORKED / 10
```

where ACP_HOURS_WORKED is a database item. This kind of calculation would be found typically in the looping formula.

#### Using Up Front Accruals

The sample formula assumes that plan participants accrue a certain amount of time each month. But in some plans, participants accrue their full entitlement at the start of the accrual term. In this case the formula does not need to loop through each accrual period.

Here is a very basic sample formula, assuming an accrual of 20 days for every calendar year. There are no ineligibility rules or start rules in this example. It also does not calculate the accrual for part years (for example, for employees joining the plan midway through a year).

```
INPUTS ARE
Calculation_Date (date)

Total_Accrued_PTO = 20
Effective_start_date =
     to_date('0101'||to_char(calculation_date, 'YYYY'),'DDMMYYYY')
Effective_end_date =
     to_date('3112'||to_char(calculation_date, 'YYYY'),'DDMMYYYY')
Accrual_end_date =
     to_date('0101'||to_char(calculation_date, 'YYYY'),'DDMMYYYY')

RETURN Total_Accrued_PTO,
       Effective_start_date,
       Effective_end_date,
       Accrual_end_date
```

#### Changing the Ceiling

In the sample top level formula, the ceiling is set at line 7. You can change the ceiling within the formula, or set it to zero to remove the ceiling:

```
E = SET_NUMBER('CEILING', 0)
```

You can also set the ceiling outside the formula, using the Accrual Bands window or a user table.

#### Changing the Date Used for Continuous Service

The continuous service date is used in the formula to determine when a new hire begins to accrue time. If your plan has accrual bands based on length of service, this date also determines the amount the employee is eligible to accrue.

In the sample, length of service is calculated from hire date (using the database item ACP_SERVICE_START_DATE) or the continuous service date input value, if it was entered when the employee was enrolled in the plan (database item ACP_CONTINUOUS_SERVICE_DATE). Alternatively, you can define and use another database item. For example, replace lines 32 to 36 of the top level formula with:

```
E = set_date('CONTINUOUS_SERVICE_DATE', <NEW_DATABASE_ITEM>)

Continuous_Service_Date = get_date('CONTINUOUS_SERVICE_DATE')
```

#### Adding Rules for Suspended Assignments

Some accrual plans may not allow employees to accrue PTO while on certain types of leave, such as maternity leave or study leave. In this case, your formula needs to check

the status of the assignment. You can use the function GET_ASG_INACTIVE_DAYS.

Replace line 15 of the looping formula with:

```
Assignment_Inactive_Days = GET_ASG_INACTIVE_DAYS(Period_SD, Period_ED)
IF Assignment_Inactive_Days <> 0 THEN
    (
      Working_Days = GET_WORKING_DAYS(Period_SD, Period_ED)
          IF Working_Days = Assignment_Inactive_Days THEN
              (
                Multiplier = 0
                )
          ELSE
              (
                Multiplier = 1 - (Assignment_Inactive_Days /
Working_Days)
                )       )
 ELSE
    (
    Multiplier = 1
    )
   Period_Accrued_PTO = Accrual_Rate * Multiplier
```

#### Adding Rules for Part Time Employees

You can use the ASG_EMPLOYMENT_CATEGORY database item to check whether the assignment is part time or full time.

Suppose part timers accrue at only half the rate of full timers, and have a lower ceiling, then replace lines 7 and 8 with:

```
IF ASG_EMPLOYMENT_CATEGORY = 'FULL TIME' then

(

  E = SET_NUMBER('CEILING', 20)

  E = SET_NUMBER('ACCRUAL_RATE', 2)

)

ELSE

(

  E = SET_NUMBER('CEILING', 10)

  E = SET_NUMBER('ACCRUAL_RATE', 1)

)
```

If part time employees accrue at different rates depending on their hours as a percentage of full time, you could set up HR budgets to record the value of each assignment. Then you need to define a database item for the budget value.

#### Adding a Long Service Leave Entitlement

If you want to see how much long service leave has been awarded to an employee in the Accruals window, you need to set up a separate accrual plan. Otherwise, you can include the entitlement in the standard accrual plan. The following approach adds the long service accrual in a single accrual period, ignoring the usual accrual ceiling.

Suppose employees are entitled to a one time bonus of 10 extra days after 15 years of service. First, in the top level formula, add the following after line 57 to set up a variable that can be accessed in the looping formula:

```
e = set_number('LONG_SERVICE_ACCRUAL', 0)
```

Then, in the looping formula, calculate years service to ascertain whether the employee is entitled to long service leave. Add the following after line 13:

```
years_service = floor(months_between(period_ed,
                                     continuous_service_date)/12)
```

Next, set up a variable that can be used later to detect whether long service accrual has already been added. After line 29, add:

```
long_service_accrual = get_number('LONG_SERVICE_ACCRUAL')

IF (years_service > 15) and long_service_accrual = 0 THEN
(
  long_service_accrual = 10
  e = set_number('LONG_SERVICE_ACCRUAL', long_service_accrual)
)
ELSE
(
  long_service_accrual = 0
)
```

Finally, add any long service accrual to the total accrual, ignoring the ceiling. Replace line 30 with:

```
E = set_number('TOTAL_ACCRUED_PTO',Total_Accrued_PTO +
               Period_Accrued_PTO + long_service_accrual)
```

#### Using Accrual Bands Based on Length of Service

The seeded looping formulas demonstrate how to use accrual bands based on length of service criteria (entered in the Accrual Bands window). To change the sample formulas to use accrual bands:

**In the top level formula:**

- Remove lines 7 and 8
- Add the following after line 57:

```
E = set_number('ANNUAL_RATE',0)
E = set_number('UPPER_LIMIT',0)
E = set_number('CEILING',0)
```

**In the looping formula**

- Remove lines 9, 10, 11, and 13
- Add the following after line 12:

```
Annual_Rate = get_number('ANNUAL_RATE')
Upper_Limit = get_number('UPPER_LIMIT')
Ceiling = get_number('CEILING')

Years_Service = Floor(Months_Between(Period_ED,
                      Continuous_Service_Date)/12)

/*-----------------------------------------------
If the Upper Limit was defaulted or years service
is greater than or equal to the upper limit of
the current band reset the globals to the
appropriate band.
-----------------------------------------------*/

IF (Upper_Limit = 0 OR Years_Service >= Upper_Limit) THEN
(
  If (GET_ACCRUAL_BAND(Years_Service) = 0) THEN
  (
   Annual_Rate = get_number('ANNUAL_RATE')
   Upper_Limit = get_number('UPPER_LIMIT')
   Ceiling = get_number('CEILING')
  )
  ELSE   /*function returned an error */
  (
   Continue_Processing_Flag = 'N'
   Return Continue_Processing_Flag
  )
)

Accrual_Rate = Annual_Rate / 12
```

#### Changing the Length of Service Units

Normally the bands entered in the Accrual Bands window refer to years. However, you can change the formula to interpret the bands as another unit, such as six months. In the sample given above to change the looping formula to use accrual bands, you would simply replace '12' with '6' in the line:

```
Years_Service = Floor(Months_Between(Period_ED,
                      Continuous_Service_Date)/12)
```

#### Using Accrual Bands Based on Other Criteria

You can set up a user table to hold the values you require. For example:

| | Accrual | Ceiling | Max Carry Over |
|---|---|---|---|
| Grade A | 20 | 25 | 5 |
| Grade B | 24 | 25 | 8 |

You can also base your accrual bands on a combination of criteria (such as grade and length of service). In this case, you need to set up separate user tables for each value you want to hold (such as accrual amount, ceiling and maximum carry over). The table for accrual amount might look like this:

| | Grade A | Grade B |
|---|---|---|
| 0 to 5 years service | 20 | 25 |
| 5 to 50 years service | 24 | 28 |

To use data from a user table in a formula, use the GET_TABLE_VALUE function:

```
yearly_accrual = get_table_value(<table_name>, <column_name>,
                                 <row_value>, <effective_date>)
```

The effective date parameter is optional. Example:

```
yearly_accrual = get_table_value ('MY_TABLE', 'GRADE A', '5')
```

## Sample Proration Formula

This is a sample formula for calculating union dues. It serves as an example of how to use the core proration functionality.

```
/* $Header: pyusvoldedtmpltf.hdt 115.4 2004/05/20 17:16:32 meshah noship
$ */


/*
+==========================================================================+
   |                      Copyright (c) 1993 Oracle                 |
   |                   Redwood Shores, California, USA
|
   |                         All rights reserved.
|
+==========================================================================+
*/

/*************************************************************************
Checking version_chek 115.3 This version will be returned
$Header: pyusvoldedtmpltf.hdt 115.4 2004/05/20 17:16:32 meshah noship $
#dbdrv: none


FORMULA NAME:   SYSTEM_DEDN_CALC_FORMULA
DESCRIPTION:    Contains formula to handle ALL
                generated deductions.
*************************************************************************
FORMULA TEXT

Formula Results :

 dedn_amt        Direct Result for Deduction Amount
 bene_er_contr   Indirect Result for Benefit Employer Charges
 not_taken       Update Deduction Recurring Entry Not Taken
 to_arrears      Update Deduction Recurring Entry Arrears Contr
 to_purch_bal    Indirect Result for Bond Purchase
 STOP_ENTRY      Stop current recurring entry
 to_total_owed   Update Deduction Recurring Entry Accrued
 bond_refund     Indirect Result for Bond Refund
```

```
 clear_repl_amt  Update Recurring Replacement_Amount
 clear_addl_amt  Update Recurring Additional_Amount
 mesg            Message (Warning)
*****************************************************************/
```

```
/* ===== Database Item Defaults Begin ===== */



default for UNION_DUES_ROW_TYPE is 'NOT ENTERED'
default for UNION_DUES_PAYROLL_TABLE is 'NOT ENTERED'
default for UNION_DUES_PARTIAL_EE_CONTRIBUTIONS is 'NOT ENTERED'
default for UNION_DUES_ACCRUED_ASG_ITD is 0
default for UNION_DUES_ARREARS_ASG_ITD is 0
default for UNION_DUES_TOWARD_BOND_PURCHASE_ASG_ITD is 0
default for UNION_DUES_PERIOD_TYPE is 'NOT ENTERED'
default for PAY_PROC_PERIOD_START_DATE is'0001/01/01 00:00:00' (DATE)

default for PAY_PROC_PERIOD_END_DATE is'0001/01/01 00:00:00' (DATE)
Default for UNION_DUES_ADDITIONAL_ASG_GRE_ITD           is 0
Default for UNION_DUES_REPLACEMENT_ASG_GRE_ITD          is 0
Default for UNION_DUES_ASG_GRE_RUN                      is 0
Default for UNION_DUES_ASG_GRE_YTD                      is 0
/* 330329 begin */
Default for UNION_DUES_ASG_GRE_MONTH            is 0
/* 330329 end */

default for REGULAR_EARNINGS_ASG_RUN            is 0
default for NET_ASG_RUN                         is 0

default for ARREARS_FLAG                        is 'NOT ENTERED'
default for TERMINATED_EMPLOYEE                 is 'N'
default for FINAL_PAY_PROCESSED                 is 'N'
default for ELEMENT_CATEGORY                    is ' '
default for PER_AGE                             is 0
default for ASG_SALARY                          is 0
```

```
default for UNION_DUES_OVERLIMIT_ASG_GRE_RUN    is 0
default for UNION_DUES_ELIGIBLE_COMP_ASG_RUN    is 0


/* ===== Database Item Defaults End ===== */

/* ===== Input Value Defaults Begin ===== */

default for Purchase_Price                      is 0
default for Total_Owed                          is 0
default for Towards_Owed (text)                 is 'Y'
default for Clear_Arrears (text)                is 'N'
default for Amount                              is 0
default for Percentage                          is 0
default for Table_Column (text)                 is 'NOT ENTERED'
default for Coverage (text)                     is 'NOT ENTERED'
default for EE_Contr                            is 0
default for ER_Contr                            is 0
default for Guaranteed_Net                      is 0

default for prorate_start is '01-JAN-1900' (date)
default for prorate_end is '01-JAN-1900' (date)

/* ===== Input Value Defaults End ===== */

/* ===== Inputs Section Begin ===== */

Inputs are
             Amount                 /* Flat Amount calc rule */
        ,Percentage             /* % Earnings calc rule */
        ,Table_Column (text)    /* Payroll Table calc rule */
            ,Coverage (text)        /* Ben Table calc rule */
            ,EE_Contr                     /* Ben Table calc rule */
```

```
,ER_Contr                      /* Ben Table calc rule */
        ,Purchase_Price         /* EE Bond Price */
            ,Guaranteed_Net                         /* Involuntary
Garnishment dedn */
        ,Total_Owed             /* Total Reached stop rule */
        ,Towards_Owed (text)    /* Total Reached stop rule */
            ,Clear_Arrears (text)       /* Clears down any amount in
arrears */
    , PRORATE_START (Date)
    , PRORATE_END (Date)




/* ===== Inputs Section End ===== */


/* ===== Latest balance creation begin ==== */

              SOE_run = UNION_DUES_ASG_GRE_RUN
              SOE_ytd = UNION_DUES_ASG_GRE_YTD
          Dedn_report_dummy = UNION_DUES_ACCRUED_ASG_ITD


/* ===== Latest balance creation end ==== */


dedn_amt = 0


/* ===== AMOUNT CALCULATION SECTION BEGIN ===== */


  IF Percentage WAS DEFAULTED THEN      /* NOT % Earnings */


    IF Table_Column   WAS DEFAULTED THEN     /* NOT Payroll Table */


      IF Coverage WAS DEFAULTED THEN    /* NOT Benefits Table */


        IF Amount WAS DEFAULTED THEN    /* NOT Flat Amount either! */
```

```
            IF UNION_DUES_REPLACEMENT_ASG_GRE_ITD WAS DEFAULTED OR
               UNION_DUES_REPLACEMENT_ASG_GRE_ITD = 0 THEN


              ( /* No calculation method found for element */
                mesg = 'UNION_DUES: Could not find appropriate values to
calculate deduction.'
                RETURN mesg
              )


            ELSE


              (dedn_amt = UNION_DUES_REPLACEMENT_ASG_GRE_ITD
               /* Not subject to dedn_freq_factor */
               clear_repl_amt = -1 * UNION_DUES_REPLACEMENT_ASG_GRE_ITD
              )


          ELSE                    /* Flat Amount */


            IF UNION_DUES_REPLACEMENT_ASG_GRE_ITD WAS DEFAULTED OR
                UNION_DUES_REPLACEMENT_ASG_GRE_ITD = 0 THEN


              (
                deduction_frequency_factor = dedn_freq_factor
(UNION_DUES_PERIOD_TYPE)
                dedn_amt = Amount * deduction_frequency_factor


/* 330329 begin */
/* 401566 and 510356 : */


                IF UNION_DUES_PERIOD_TYPE = 'Calendar Month'  THEN


                          IF (dedn_amt + UNION_DUES_ASG_GRE_MONTH -
Amount ) <= .02  AND
                     ( UNION_DUES_ASG_GRE_MONTH + dedn_amt) > Amount THEN
```

```
                    dedn_amt = Amount - UNION_DUES_ASG_GRE_MONTH
/* 330329 end */

           )

         ELSE

           (deduction_frequency_factor = dedn_freq_factor
(UNION_DUES_PERIOD_TYPE)
            dedn_amt = UNION_DUES_REPLACEMENT_ASG_GRE_ITD *
deduction_frequency_factor
            clear_repl_amt = -1 * UNION_DUES_REPLACEMENT_ASG_GRE_ITD
           )

      ELSE              /* Using Benefits Table */

        IF UNION_DUES_REPLACEMENT_ASG_GRE_ITD WAS DEFAULTED OR
            UNION_DUES_REPLACEMENT_ASG_GRE_ITD = 0 THEN

         (deduction_frequency_factor = dedn_freq_factor
(UNION_DUES_PERIOD_TYPE)

          /* Check that user has not entered explicit value in input
val: */

          IF EE_Contr WAS DEFAULTED THEN

           (
             dedn_amt = 0 * deduction_frequency_factor

/* 330329 begin */
/* 401566 and 510356 : */

             IF UNION_DUES_PERIOD_TYPE = 'Calendar Month'  THEN
```

```
                    IF (dedn_amt + UNION_DUES_ASG_GRE_MONTH
                              - 0 ) <= .02  AND
                   (UNION_DUES_ASG_GRE_MONTH + dedn_amt) > 0  THEN

                  dedn_amt = 0 - UNION_DUES_ASG_GRE_MONTH
/* 330329 end */
            )

          ELSE

            ( dedn_amt = EE_Contr * deduction_frequency_factor

/* 330329 begin */
/* 401566 and 510356 : */

              IF UNION_DUES_PERIOD_TYPE = 'Calendar Month'  THEN

                   IF ( dedn_amt + UNION_DUES_ASG_GRE_MONTH - EE_Contr)
<= .02  AND
                  ( UNION_DUES_ASG_GRE_MONTH + dedn_amt) > EE_Contr THEN

                  dedn_amt = EE_Contr - UNION_DUES_ASG_GRE_MONTH
/* 330329 end */
            )

          /* Check that user has not entered explicit value in input
val: */

          IF ER_Contr WAS DEFAULTED THEN

            bene_er_contr = 0 * deduction_frequency_factor

          ELSE
```

```
bene_er_contr = ER_Contr * deduction_frequency_factor

         )

        ELSE   /* An override Deduction Amount has been entered. */
```

```
          ( dedn_amt = UNION_DUES_REPLACEMENT_ASG_GRE_ITD

            clear_repl_amt = -1 * UNION_DUES_REPLACEMENT_ASG_GRE_ITD


            deduction_frequency_factor = dedn_freq_factor
(UNION_DUES_PERIOD_TYPE)


          /* Still check for input value for ER Contr: */

            IF ER_Contr WAS DEFAULTED THEN


              bene_er_contr = 0 * deduction_frequency_factor


            ELSE


              bene_er_contr = ER_Contr * deduction_frequency_factor

            /* EE Contr from Benefits Contributions table has been
overridden. */


          )


    ELSE               /* Using Payroll Table */


      IF UNION_DUES_REPLACEMENT_ASG_GRE_ITD WAS DEFAULTED OR

         UNION_DUES_REPLACEMENT_ASG_GRE_ITD = 0 THEN


     /* not overridden */


        IF UNION_DUES_ROW_TYPE WAS DEFAULTED THEN


/* ---------------------------------------------------------------------


CUSTOMER :  Remove the following 2 lines and insert your own call to

            Get_Table_Value if you are not using our Row Type field in

            the Element Additional Information

 ---------------------------------------------------------------------
*/
```

```
         (
          mesg = 'UNION_DUES: Could not find a Row Type for this payroll
table.'
          RETURN mesg
         )

       ELSE

         IF UNION_DUES_ROW_TYPE = 'AGE' THEN

          (deduction_frequency_factor = dedn_freq_factor
(UNION_DUES_PERIOD_TYPE)

           table_amount = To_Number (
                              Get_Table_Value(
                                  UNION_DUES_PAYROLL_TABLE,
                                  Table_Column,
                                  To_Char(PER_AGE) ) )

           dedn_amt = table_amount * deduction_frequency_factor

/* 330329 begin */
/* 401566 and 510356 : */

           IF UNION_DUES_PERIOD_TYPE = 'Calendar Month'  THEN

                IF (dedn_amt + UNION_DUES_ASG_GRE_MONTH - table_amount)
<= .02  AND
               ( UNION_DUES_ASG_GRE_MONTH + dedn_amt) > table_amount
THEN

                  dedn_amt = table_amount - UNION_DUES_ASG_GRE_MONTH
/* 330329 end */
          )
```

```
ELSE

          IF UNION_DUES_ROW_TYPE = 'SAL' THEN

            (deduction_frequency_factor = dedn_freq_factor
(UNION_DUES_PERIOD_TYPE)

              table_amount = To_Number (
                                  Get_Table_Value(
                                       UNION_DUES_PAYROLL_TABLE,
                                       Table_Column,
                                       To_Char(ASG_SALARY) ) )

              dedn_amt = table_amount * deduction_frequency_factor

/* 330329 begin */
/* 401566 and 510356 : */

              IF UNION_DUES_PERIOD_TYPE = 'Calendar Month' THEN

                        IF (dedn_amt + UNION_DUES_ASG_GRE_MONTH -
table_amount) <= .02  AND
                   ( UNION_DUES_ASG_GRE_MONTH + dedn_amt  ) >
table_amount THEN

                     dedn_amt = table_amount - UNION_DUES_ASG_GRE_MONTH
/* 330329 end  */
             )

            ELSE

              ( deduction_frequency_factor = dedn_freq_factor
(UNION_DUES_PERIOD_TYPE)

                table_amount = To_Number ( Get_Table_Value(
```

```
UNION_DUES_PAYROLL_TABLE,
                                                Table_Column,
                                                'NOT ENTERED' ) )

                dedn_amt = table_amount * deduction_frequency_factor

/* 330329 begin */
/* 401566 and 510356 :  */

                IF UNION_DUES_PERIOD_TYPE = 'Calendar Month'  THEN

                           IF (dedn_amt + UNION_DUES_ASG_GRE_MONTH -
table_amount) <= .02  AND
                     ( UNION_DUES_ASG_GRE_MONTH + dedn_amt) >
table_amount THEN

                    dedn_amt = table_amount - UNION_DUES_ASG_GRE_MONTH
/* 330329 end */
            )

      ELSE /* Payroll table lookup overridden */

        ( dedn_amt = UNION_DUES_REPLACEMENT_ASG_GRE_ITD
          clear_repl_amt = -1 * UNION_DUES_REPLACEMENT_ASG_GRE_ITD
        )

  ELSE          /* Percent of Earnings Balance amount rule */

    IF UNION_DUES_REPLACEMENT_ASG_GRE_ITD WAS DEFAULTED OR
       UNION_DUES_REPLACEMENT_ASG_GRE_ITD = 0 THEN

   /* Not Overridden */

/*
---------------------------------------------------------------------
```

```
CUSTOMER :  The formula is generated with a default to use the Eligible
                        Compensation to calculate % of Earnings.  The
Eligible Compensation
                        balance is initially defined with a balance feed
from the Regular
                        Earnings balance.


                        You can modify the earnings basis for this
calculation by
                        adding and deleting balance feeds to the
                        UNION_DUES_ELIGIBLE_COMP balance.


                        If you want the formula to use another balance
of earnings in
                        the run, replace the
UNION_DUES_ELIGIBLE_COMP_ASG_RUN database item
                        reference below with the database item for the
balance of choice :
                        <BALANCE_NAME_IN_UPPERCASE/UNDERSCORES>_ASG_RUN


 ----------------------------------------------------------------------
*/
```

```
     (deduction_frequency_factor = dedn_freq_factor
(UNION_DUES_PERIOD_TYPE)

      dedn_amt = (Percentage * UNION_DUES_ELIGIBLE_COMP_ASG_RUN  / 100)
* deduction_frequency_factor

     )


    ELSE


     (dedn_amt = UNION_DUES_REPLACEMENT_ASG_GRE_ITD

      clear_repl_amt = -1 * UNION_DUES_REPLACEMENT_ASG_GRE_ITD

     )




/* ===== AMOUNT CALCULATION SECTION END ===== */


/* ===== Template Adjustments Section Begin ===== */


dedn_amt = dedn_amt + UNION_DUES_ADDITIONAL_ASG_GRE_ITD


IF UNION_DUES_ADDITIONAL_ASG_GRE_ITD <> 0 THEN


  clear_addl_amt = -1 * UNION_DUES_ADDITIONAL_ASG_GRE_ITD


IF UNION_DUES_REPLACEMENT_ASG_GRE_ITD <> 0 THEN


  clear_repl_amt = -1 * UNION_DUES_REPLACEMENT_ASG_GRE_ITD


/* ===== Template Adjustments Section Begin ===== */


/*

*******

For Pretax Deduction processing...

make the break here between calc and withholding ffs.

To do this, the following input values must be passed between
```

```
calc and withholding elements:
1. Clear Arrears
2. Guaranteed Net
3. Total Owed
4. Towards Owed
5. Purchase Price (if Pretax EE Bonds are valid requirement).
*******
*/

/* ===== Arrears Section Begin ===== */

IF Clear_Arrears = 'N' THEN

 (
  temp_to_arrears = 0
  temp_not_taken = 0

  dedn_amt = arrearage( UNION_DUES_PARTIAL_EE_CONTRIBUTIONS,
                                    NET_ASG_RUN,
                                    UNION_DUES_ARREARS_ASG_ITD,
                                    Guaranteed_Net,
                                    dedn_amt,
                                    temp_to_arrears,
                                    temp_not_taken)

  IF temp_to_arrears <> 0 THEN

    to_arrears = temp_to_arrears

  IF temp_not_taken <> 0 THEN

    not_taken = temp_not_taken
```

```
)

ELSE /* Clear down arrears balance. */

 (

  to_arrears = -1 * UNION_DUES_ARREARS_ASG_ITD

  set_clear = 'No'

  temp_to_arrears = 0

  temp_not_taken = 0

  /* Call with 0 amount for ARREARS_ASG_ITD b/c balance has been
cleared. */

  dedn_amt = arrearage( UNION_DUES_PARTIAL_EE_CONTRIBUTIONS,
                                    NET_ASG_RUN,
                                    0,
                                    Guaranteed_Net,
                                    dedn_amt,
                                    temp_to_arrears,
                                    temp_not_taken)

  IF temp_to_arrears <> 0 THEN

   to_arrears = to_arrears + temp_to_arrears

  IF temp_not_taken <> 0 THEN

    not_taken = temp_not_taken

 )

/* ===== Arrears Section End ===== */
```

```
/* ===== EE Bond Section Begin ===== */

IF Purchase_Price WAS NOT DEFAULTED THEN

  ( total_for_bond = dedn_amt +
                              UNION_DUES_TOWARD_BOND_PURCHASE_ASG_ITD

    IF  total_for_bond >= Purchase_Price  THEN

    /* Purchase new bonds - recalculate Toward Bond Purchase Balance */

      ( no_bonds = trunc ( total_for_bond / Purchase_Price )

        mesg = 'UNION_DUES: Please purchase '||to_char(no_bonds)||' bond
(s) for this employee.'

        to_purch_bal = (total_for_bond

                        - (no_bonds * Purchase_Price)

                        - UNION_DUES_TOWARD_BOND_PURCHASE_ASG_ITD)

      )

    ELSE

      to_purch_bal = dedn_amt

  )

/* ===== EE Bond Section End ===== */



/* ===== Stop Rule Section Begin ===== */

IF Total_Owed WAS NOT DEFAULTED THEN

 IF Towards_Owed = 'Y' THEN
```

```
/* Put towards total owed - ie reduce declining balance */
```

```
(total_accrued  = dedn_amt + UNION_DUES_ACCRUED_ASG_ITD

    IF total_accrued  > Total_Owed THEN

      (dedn_amt = Total_Owed - UNION_DUES_ACCRUED_ASG_ITD

       /* The total has been reached - the return will stop the entry
under
          these conditions.  Also, zero out Accrued balance.  */

       to_total_owed = -1 * UNION_DUES_ACCRUED_ASG_ITD
       STOP_ENTRY = 'Y'
       mesg = 'UNION_DUES entry has been stopped.'
      )

    ELSE /* Total Owed not reached yet. */

      to_total_owed = dedn_amt

    )

/* ===== Stop Rule Section End ===== */

/* ===== Final Pay Section Begin ===== */

IF (substr(ELEMENT_CATEGORY,1,5) != 'PENSN') AND
   (TERMINATED_EMPLOYEE = 'Y' AND FINAL_PAY_PROCESSED = 'N') THEN

  /* This IS final pay run, considerations:
       1) Try and collect remainder of Total Owed;
          (consider Partial Deduction, not Arrears)
       2) Refund EE Bond Towards Purchase balance
  */
```

```
/* Customer we will no longer end date an element in this
   method.  If you still desire to end date elements in
   this way you may un comment this code.
*/

 ( /* STOP_ENTRY = 'Y'
   mesg = 'UNION_DUES entry has been stopped.'
   */

   IF Purchase_Price WAS DEFAULTED THEN  /* IF not EE Bond then Return
*/

    (

     IF Total_Owed WAS NOT DEFAULTED THEN

     /* Try and collect Total Owed. */

      (dedn_left = Total_Owed - UNION_DUES_ACCRUED_ASG_ITD

       IF NET_ASG_RUN > dedn_left  THEN  /* Recoup all Remainder */

        ( dedn_amt = dedn_left
          to_total_owed = dedn_amt )

       ELSE

        ( dedn_amt = NET_ASG_RUN
          to_total_owed = dedn_amt )

      )

     RETURN dedn_amt, bene_er_contr, not_taken, to_arrears, STOP_ENTRY,
            to_total_owed, clear_repl_amt, clear_addl_amt, set_clear,
```

```
mesg
    )

   ELSE

      /* Refund Towards Purchase balance via      */
      /* direct payment; zero Towards Purch bal.  */

     (bond_refund = UNION_DUES_TOWARD_BOND_PURCHASE_ASG_ITD
      to_purch_bal = -1 * UNION_DUES_TOWARD_BOND_PURCHASE_ASG_ITD
      dedn_amt = 0
      to_total_owed = 0

      RETURN dedn_amt, bene_er_contr, not_taken, to_arrears, STOP_ENTRY,
             to_total_owed, bond_refund, to_purch_bal, clear_repl_amt,
             clear_addl_amt, set_clear, mesg
     )

 )

ELSE    /* Not Final Pay */

Total_days = DAYS_BETWEEN(PAY_PROC_PERIOD_END_DATE,
PAY_PROC_PERIOD_START_DATE) +1

Days_in_pro_period = DAYS_BETWEEN( Prorate_END, Prorate_Start)+1

Dedn_Amt = Amount * Days_in_pro_period/Total_days

  RETURN dedn_amt, bene_er_contr, not_taken, to_arrears, to_total_owed,
         to_purch_bal, clear_repl_amt, clear_addl_amt, STOP_ENTRY,
set_clear, Days_in_pro_period, Total_days,
             mesg
```

```
/* ===== Final Pay Section End ===== */


/*====================== End Payroll Formula Body
======================*/
```

## Editing a Quick Paint Formula Example

If you want to add features to a generated QuickPaint formula, you must copy the formula and edit the copy. If you edit the original, your edits will be overwritten if the formula is regenerated from the QuickPaint definition.

In the following example, an automatically generated QuickPaint formula has been edited to add Line 09, which totals the input values used in the report.

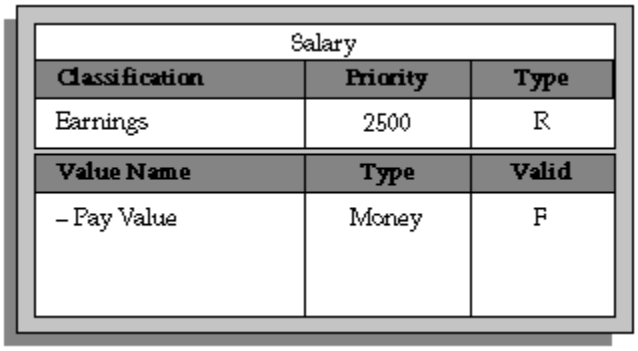
```
LINE01=''

LINE02=' Pay Items      Value this Period

LINE03=''

LINE04=' Salary Value : ' + TO_TEXT(trunc((SALARY_ANNUAL/12),2))

LINE05=' Item 1 Value : ' + TO_TEXT(ITEM_1_PAY_VALUE)

LINE06=' Item 2 Value : ' + TO_TEXT(ITEM_2_PAY_VALUE)

LINE07=' Bonus Value  : ' + TO_TEXT(BONUS_AMOUNT)

LINE08='                _____________________'

LINE09='        Total : '+ TO_TEXT(trunc((

(SALARY_ANNUAL/12)+ITEM_1_PAY_VALUE+ITEM_2_PAY_VALUE+BONUS_AMOUNT),2))

LINE10=''

LINE11=''

LINE12=''

RETURN LINE01, LINE02, LINE03, LINE04, LINE05, LINE06, LINE07,

       LINE08, LINE09, LINE10, LINE11, LINE12,
```

## Checking an Element Entry Example

You can use FastFormula to validate user entries in element input values. For example, you can make sure that entries are within a specified range or do not exceed a predefined value.

The formula below checks that the entry value of the Salary element does not exceed 200,000.

**Salary Element**



| Salary | | |
|---|---|---|
| **Classification** | **Priority** | **Type** |
| Earnings | 2500 | R |
| **Value Name** | **Type** | **Valid** |
| – Pay Value | Money | F |

```
/* Formula Name: Salary Range */

/* Formula Type: Element Input Validation */


INPUTS ARE entry_value (text)

IF TO_NUM(entry_value) > 200000

THEN

(

   formula_status = 'e'

   formula_message = 'Too much money . . . try again!'

)

ELSE

(

   formula_status = 's'

   formula_message = 'Fine'

)

RETURN formula_status, formula_message
```

## Checking a User Table Entry Example

You can use FastFormula to validate user entries into user tables that you define. For

example, you can make sure that entries are between a specified range or not a negative amount.

The formula below checks that the deduction entered in the Union A column of the Union Dues table is between 10.00 and 20.00.

```
/* Formula Name: Union A Dues Validation */

/* Formula Type: User Table Validation */


INPUTS ARE entry_value (text)

IF TO_NUMBER(entry_value) < 10.00 OR

 TO_NUMBER(entry_value) > 20.00

THEN

(

   formula_status = 'e'

   formula_message = 'Error: Union A dues must be between

                      $10.00 and $20.00.'

)

ELSE

(

   formula_status = 's'

   formula_message = ' '

)

RETURN formula_status, formula_message
```

## Sample Formula for Payroll Contact

You can write a formula to determine how you allocate a payroll contact for your employees. Modify this sample formula to match payroll contacts in your organization to employees on your payrolls.

```
/* ================================================================
               Set the default values
```

```
   ================================================================*/
default for l_first_letter is 'A'
default for l_contact_name is 'Name'
default for l_phone is '12345'
default for l_email is 'email@email.com'
default for l_ret_num is 1
/*=================================================================
PER_LAST_NAME is an existing predefined database item (DBI).
PER_LAST_NAME takes an assignment id and returns the last name for each
person.
You pass the assignment id automatically as a context usage for the DBI.
Use the substr function to return only the first letter of the last
name.
 ================================================================= */
l_first_letter = substr(PER_LAST_NAME,1,1)
/* ================================================================
  Set the contact details for those employees with a last name beginning
with A.
================================================================= */
if (l_first_letter = 'A')
then
    (l_contact_name = 'Melanie Morrisby'
     l_phone = '0118 9404040'
     l_email = 'melanie.morrisby@oracle.com'
     )
/* ================================================================
  Set the contact details for those employees with a last name beginning
with B.
================================================================*/
if (l_first_letter = 'B')
then
   (l_contact_name = 'Zaheer Ahmed'
    l_phone = '0118 9404040'
    l_email = 'zaheer.ahmed@oracle.com'
    )
/* ================================================================
  Set the contact details for those employees with a last name beginning
with C.
  ================================================================*/
if (l_first_letter = 'C' )
then
    (l_contact_name = 'Kieron Lancashire'
     l_phone = '0118 000111'
```

```
     l_email = 'kieron.lancashire@oracle.com'
      )
/* ================================================================
Set the contact details for those employees with a last name
beginning with any letter other than A, B or C.
 ================================================================*/
else
 (l_contact_name ='Jo Burgess'
  l_phone = '01189466377'
  l_email = 'josephine.burgess@oracle.com'
 )
/* ============================================================
  Write payroll details to assignment_extra_info table.
 ============================================================ */
l_ret_num = POPULATE_PAYROLL_CONTACT
(l_contact_name,l_phone,l_email)
```

## Sample Appraisal Objective Line-Scoring Formulas

This topic includes the two supplied sample formulas of type Appraisal Objective Line Scoring. When you define an objective assessment template with a rating-scale type of Fast Formula Based Line Scoring, the application uses the formula you select to calculate the score for each objective in the Objectives section of the appraisal.

### Sample Formula PERF

This sample formula returns a performance rating derived from a specified performance rating. Business group and legislation code are global (null).

```
/*************************************************************
 *
 * Formula Name : PERF
 *
 * Description  : For an objective appraisal line, this formula
 *                returns Performance.
 *
 * Formula Type : Appraisal Objective Line Scoring
 *
 * Inputs       : 1) performance, number, default required
 *                2) weighting, number, default required
 *                3) line_object_id (objective_id), number, always set
 *                4) appraisal_id, number, always set
 *                5) appr_template_id, number, always set
 *                6) appr_system_type (e.g.. SELF,EMP360,MGR360TRANS),
text, always set
 *                7) appr_type, text
 * Note         : For appraisal_id, appr_template_id, appr_system_type,
appr_type
 *                a) This is an input, not a context
 *                b) No seeded DBIs use this
 * Outputs      : 1) line_score, number
 *
 * Contexts     : Business Group, Assignment, Organization,
 *                Person, Date Earned
 *
 * Example DBIs : ptu_per_person_type, asg_grade, asg_job,
 *                asg_status, asg_type, asg_primary,
 *                asg_position, asg_hours, asg_salary
 *
 *************************************************************/
/* Defaults for optional inputs and database items */
DEFAULT FOR performance IS 0
DEFAULT FOR weighting   IS 0
```

```
/* Declare formula inputs */
INPUTS ARE  performance(number)
           ,weighting(number)
           ,line_object_id(number)
           ,appraisal_id(number)
           ,appr_template_id(number)
           ,appr_system_type(text)
           ,appr_type(text)
/* Main body of formula */
line_score = performance
/* Return the line score */
RETURN line_score
```

## Sample Formula PERF_X_WEIGHTING

This sample formula returns a performance rating derived from a specified performance rating and the weighting value specified for the objective. Business group and legislation code are global (null).

```
/*************************************************************
 *
 * Formula Name : PERF_X_WEIGHTING
 *
 * Description  : For an objective appraisal line, this formula
 *                multiplies Performance and Weighting.
 *
 * Formula Type : Appraisal Objective Line Scoring
 *
 * Inputs       : 1) performance, number, default required
 *                2) weighting, number, default required
 *                3) line_object_id (objective_id), number, always set
 *                4) appraisal_id, number, always set
 *                5) appr_template_id, number, always set
 *                6) appr_system_type (e.g.. SELF,EMP360,MGR360TRANS),
text, always set
 *                7) appr_type, text
 * Note         : For appraisal_id, appr_template_id, appr_system_type,
appr_type
 *                a) This is an input, not a context
 *                b) No seeded DBIs use this
 * Outputs      : 1) line_score, number
 *
 * Contexts     : Business Group, Assignment, Organization,
 *                Person, Date Earned
 *
 * Example DBIs : ptu_per_person_type, asg_grade, asg_job,
 *                asg_status, asg_type, asg_primary,
 *                asg_position, asg_hours, asg_salary
 *
 *************************************************************/
/* Defaults for optional inputs and database items */
DEFAULT FOR performance IS 0
DEFAULT FOR weighting   IS 0
```

```
/* Declare formula inputs */
INPUTS ARE  performance(number)
           ,weighting(number)
           ,line_object_id(number)
           ,appraisal_id(number)
           ,appr_template_id(number)
           ,appr_system_type(text)
           ,appr_type(text)
/* Main body of formula */
line_score = performance * weighting
/* Return the line score */
RETURN line_score
```

## Sample Appraisal Competency Line-Scoring Formulas

This topic includes the three supplied sample formulas of type Appraisal Competency Line Scoring. When you define a competency assessment template with a rating-scale type of Fast Formula Based Line Scoring, the application uses the formula you select to calculate the score for each competency in the Competencies section of the appraisal.

### Sample Formula PERF_X_PROF

This sample formula returns a competency rating derived from specified performance and proficiency ratings. Business group and legislation code are global (null).

```
/*************************************************************
 *
 * Formula Name : PERF_X_PROF
 *
 * Description  : For a competency appraisal line, this formula
 *                multiplies Performance and Proficiency.
 *
 * Formula Type : Appraisal Competency Line Scoring
 *
 * Inputs       : 1) performance, number, default required
 *                2) proficiency, number, default required
 *                3) weighting, number, default required
 *                4) line_object_id (competency_id), number, always set
 *                5) appraisal_id, number, always set
 *                6) appr_template_id, number, always set
 *                7) appr_system_type (e.g.. SELF,EMP360,MGR360TRANS),
text, always set
 *                8) appr_type, text
 * Note         : For appraisal_id, appr_template_id, appr_system_type,
appr_type
 *                a) This is an input, not a context
 *                b) No seeded DBIs use this
 * Outputs      : 1) line_score, number
 *
 * Contexts     : Business Group, Assignment, Organization,
 *                Person, Date Earned
 *
 * Example DBIs : ptu_per_person_type, asg_grade, asg_job,
 *                asg_status, asg_type, asg_primary,
 *                asg_position, asg_hours, asg_salary
 *
*************************************************************/
/* Defaults for optional inputs and database items */
DEFAULT FOR performance IS 0
DEFAULT FOR proficiency IS 0
```

```
DEFAULT FOR weighting   IS 0

/* Declare formula inputs */
INPUTS ARE  performance(number)
           ,proficiency(number)
           ,weighting(number)
           ,line_object_id(number)
           ,appraisal_id(number)
           ,appr_template_id(number)
           ,appr_system_type(text)
           ,appr_type(text)

/* Main body of formula */
line_score = performance * proficiency
/* Return the line score */
RETURN line_score
```

## Sample Formula PERF_X_WEIGHTING

This sample formula returns a competency rating derived from specified performance and weighting values. Business group and legislation code are global (null).

```
/*************************************************************
 *
 * Formula Name : PERF_X_WEIGHTING
 *
 * Description  : For a competency appraisal line, this formula
 *                multiplies Performance and Weighting.
 *
 * Formula Type : Appraisal Competency Line Scoring
 *
 * Inputs       : 1) performance, number, default required
 *                2) proficiency, number, default required
 *                3) weighting, number, default required
 *                4) line_object_id (competency_id), number, always set
 *                5) appraisal_id, number, always set
 *                6) appr_template_id, number, always set
 *                7) appr_system_type (e.g.. SELF,EMP360,MGR360TRANS),
text, always set
 *                8) appr_type, text
 * Note         : For appraisal_id, appr_template_id, appr_system_type,
appr_type
 *                a) This is an input, not a context
 *                b) No seeded DBIs use this
 * Outputs      : 1) line_score, number
 *
 * Contexts     : Business Group, Assignment, Organization,
 *                Person, Date Earned
 *
 * Example DBIs : ptu_per_person_type, asg_grade, asg_job,
 *                asg_status, asg_type, asg_primary,
 *                asg_position, asg_hours, asg_salary
 *************************************************************/
/* Defaults for optional inputs and database items */
DEFAULT FOR performance IS 0
DEFAULT FOR proficiency IS 0
```

```
DEFAULT FOR weighting   IS 0

/* Declare formula inputs */
INPUTS ARE  performance(number)
           ,proficiency(number)
           ,weighting(number)
           ,line_object_id(number)
           ,appraisal_id(number)
           ,appr_template_id(number)
           ,appr_system_type(text)
           ,appr_type(text)
/* Main body of formula */
line_score = performance * weighting
/* Return the line score */
RETURN line_score
```

## Sample Formula PROF _X_WEIGHTING

This sample formula returns a competency rating derived from specified proficiency and weighting values. Business group and legislation code are global (null).

```
/*************************************************************
 *
 * Formula Name : PROF_X_WEIGHTING
 *
 * Description  : For a competency appraisal line, this formula
 *                multiplies Proficiency and Weighting.
 *
 * Formula Type : Appraisal Competency Line Scoring
 *
 * Inputs       : 1) performance, number, default required
 *                2) proficiency, number, default required
 *                3) weighting, number, default required
 *                4) line_object_id (competency_id), number, always set
 *                5) appraisal_id, number, always set
 *                6) appr_template_id, number, always set
 *                7) appr_system_type (e.g.. SELF,EMP360,MGR360TRANS),
text, always set
 *                8) appr_type, text
 * Note         : For appraisal_id, appr_template_id, appr_system_type,
appr_type
 *                a) This is an input, not a context
 *                b) No seeded DBIs use this
 * Outputs      : 1) line_score, number
 *
 * Contexts     : Business Group, Assignment, Organization,
 *                Person, Date Earned
 *
 * Example DBIs : ptu_per_person_type, asg_grade, asg_job,
 *                asg_status, asg_type, asg_primary,
 *                asg_position, asg_hours, asg_salary
 *************************************************************/
/* Defaults for optional inputs and database items */
DEFAULT FOR performance IS 0
DEFAULT FOR proficiency IS 0
```

```
DEFAULT FOR weighting   IS 0

/* Declare formula inputs */
INPUTS ARE  performance(number)
           ,proficiency(number)
           ,weighting(number)
           ,line_object_id(number)
           ,appraisal_id(number)
           ,appr_template_id(number)
           ,appr_system_type(text)
           ,appr_type(text)
/* Main body of formula */
line_score = proficiency * weighting
/* Return the line score */
RETURN line_score
```

## Sample Appraisal Total Scoring Formulas

This topic includes the two supplied formulas of type Appraisal Total Scoring. When you define an appraisal template, you can select an Appraisal Total Score Formula to calculate the appraisee's suggested overall rating using the total scores for competencies and objectives. This total score appears in the Overall Ratings region of the Final Ratings page.

> **Note:** The Appraisal Total Score Formula must return a rating level ID rather than a rating value.
>
> See: Writing Formulas for Rating Competencies and Objectives, page 1-143

### Sample Formula SUM_COMP_AND_OBJ

This sample formula adds the final scores for objectives and competencies and uses the result to identify a final rating.

```
/*************************************************************
 *
 * Formula Name : SUM_COMP_AND_OBJ
 *
 * Description  : This sums the competency and objective scores,
 *                and then uses hard-coded bands to calculate
 *                a final rating.
 *
 * Formula Type : Appraisal Total Scoring
 *
 * Inputs       : 1) competency_score, number, always set
 *                2) objective_score, number, always set
 *                3) appraisal_id, number, always set
 *                4) appr_template_id, number, always set
 *                5) appr_system_type (e.g.. SELF,EMP360,MGR360TRANS),
text, always set
 *                6) appr_type, text
 * Note         : For appraisal_id, appr_template_id, appr_system_type,
appr_type
 *                a) This is an input, not a context
 *                b) No seeded DBIs use this
 *
 * Outputs      : 1) final_rating, number
 *
 * Contexts     : Business Group, Assignment, Organization,
 *                Person, Date Earned
 *
 * Example DBIs : ptu_per_person_type, asg_grade, asg_job,
 *                asg_status, asg_type, asg_primary,
 *                asg_position, asg_hours, asg_salary
*************************************************************/
/* Defaults for optional inputs and database items */
DEFAULT FOR competency_score IS 0
DEFAULT FOR objective_score IS 0
```

```
/* Declare formula inputs */
INPUTS ARE  competency_score(number)
           ,objective_score(number)
           ,appraisal_id(number)

/* Main body of formula. */
total_score = competency_score + objective_score

/* Band the total score to give a final rating */
IF total_score < 50 THEN
  final_rating = 1
IF total_score >= 50 AND total_score < 100 THEN
  final_rating = 2
IF total_score >= 100 AND total_score < 150 THEN
  final_rating = 3
IF total_score >= 150 AND total_score < 200 THEN
  final_rating = 4
IF total_score >= 200 THEN
  final_rating = 5

/* Return the final rating */
RETURN final_rating
```

## Sample Formula AVG_COMP_AND_OBJ

This sample formula calculates the average of the total scores for objectives and competencies and uses that value to identify a final rating.

```
/*************************************************************
 *
 * Formula Name : AVG_COMP_AND_OBJ
 *
 * Description  : This takes the average of the competency and
 *                objective scores and then uses this to
 *                determine the final rating.
 *
 * Formula Type : Appraisal Total Scoring
 *
 * Inputs       : 1) competency_score, number, always set
 *                2) objective_score, number, always set
 *                3) appraisal_id, number, always set
 *                4) appr_template_id, number, always set
 *                5) appr_system_type (e.g.. SELF,EMP360,MGR360TRANS),
text, always set
 *                6) appr_type, text
 * Note         : For appraisal_id, appr_template_id, appr_system_type,
appr_type
 *                a) This is an input, not a context
 *                b) No seeded DBIs use this
 *
 * Outputs      : 1) final_rating, number
 *
 * Contexts     : Business Group, Assignment, Organization,
 *                Person, Date Earned
 *
 * Example DBIs : ptu_per_person_type, asg_grade, asg_job,
 *                asg_status, asg_type, asg_primary,
 *                asg_position, asg_hours, asg_salary
 *************************************************************/
/* Defaults for optional inputs and database items */
DEFAULT FOR competency_score IS 0
DEFAULT FOR objective_score IS 0
```

```
/* Declare formula inputs */
INPUTS ARE  competency_score(number)
           ,objective_score(number)
           ,appraisal_id(number)

/* Main body of formula. */
avg_score = (competency_score + objective_score) / 2

/* Convert the average into a final rating */
IF avg_score < 1.5 THEN
  final_rating = 1
IF avg_score >= 1.5 AND avg_score < 2.5 THEN
  final_rating = 2
IF avg_score >= 2.5 AND avg_score < 3.5 THEN
  final_rating = 3
IF avg_score >= 3.5 AND avg_score < 4.5 THEN
  final_rating = 4
IF avg_score >= 4.5 THEN
  final_rating = 5

/* Return the final rating */
RETURN final_rating
```

# Legislative Formulas

## Sample Accrual Formula (Belgium)

This topic suggests how you can implement a whole range of accrual plan rules in your Accrual formula. The suggestions are based on the sample seeded Belgium PTO_BE_PREVYR_MULTIPLIER formula. The sample formula is for a plan with the following rules:

- An accrual term of one calendar year starting 01 January.
- Monthly accrual periods and a fixed accrual of 2 days per month.
- Accrual for new hires begins on their hire date.

The top level formula repeatedly calls another formula in a loop to calculate the accrual for each period. Both the top level formula (PTO_BE_PREVYR_MULTIPLIER) and the looping formula (PTO_BE_PREVYR_PERIOD_ACCRUAL) are given below.

```
1.   /*-----------------------------------------------

NAME : PTO_BE_PREVYR_MULTIPLER

This formula calculates the PTO accrued at a given

point in time

-----------------------------------------------*/

2.   DEFAULT FOR EMP_HIRE_DATE IS

      '4712/12/31 00:00:00' (date)

3.   DEFAULT FOR ACP_ENROLLMENT_START__DATE IS

      '0001/01/01 00:00:00' (date)

4.   DEFAULT FOR ACP_ENROLLMENT_END_DATE IS

      '4712/12/31 00:00:00' (date)

5.   DEFAULT FOR ACP_TERMINATION_DATE IS

      '4712/12/31 00:00:00' (date)

6.   DEFAULT FOR PEOPLE_BE_SCHOOL_LEAVER IS
```

```
      'NO'
7.  /*-----------------------------------------------
     The following is the date on which the number of accrued
     PTO days is required.
     -----------------------------------------------*/
8.   INPUTS ARE
     Calculation_Date (date)
9.  /*-----------------------------------------------
     Get the basic dates and information used in the
     calculation of the accrued PTO.
     -----------------------------------------------*/
10.  PrevYr = TO_CHAR(ADD_MONTHS(calculation date,-12),'YYYY')
11.  PrevYrStartDate=TO_DATE(PrevYr||'0101',YYYYMMDD')
12.  PrevYrEndDate=TO_DATE(PrevYr||'1231', 'YYYYMMDD')
13.  /*-------------------------------------------------
     Calculate the start and end dates of the current year.
.    -------------------------------------------------*/
14.  CurrYr=TO_CHAR(calculation_date, 'YYYY')
15.  CurrYrStartDate=TO_DATE(CurrYr||'0101','YYYYMMDD')
16.  CurrYrEndDate=TO_DATE(CurrYr||'1231','YYYYMMDD')
17.  /*-------------------------------------------------
     Calculate the total period over which PTO can be
     accrued for the plan.
     -------------------------------------------------*/
18.  AccrualPeriodStartDate=ACP_ENROLLMENT_START_DATE
```

```
19.  AccrualPeriodEndDate=LEAST(ACP_ENROLLMENT_END_DATE,
     ACP_TERMINATION_DATE)
20.  /*-------------------------------------------------
     Calculate the total period over which absences can be
     recorded for the plan.
     -------------------------------------------------*/
21.  AbsencePeriodStartDate=ACP_ENROLLMENT_START_DATE
     AbsencePeriodEnd=ACP_ENROLLMENT_END_DATE
22   /*-----------------------------------------------
     Get the date on which the employee was hired.
      -----------------------------------------------*/
23.  EmpHireDate=EMP_HIRE_DATE
24.  /*-----------------------------------------------
     The next section calculates the PTO they accrued
     in the previous year.
     -------------------------------------------------*/
25.  /*-----------------------------------------------
     School Leavers
     If the school leaver joined the company and was enrolled
     on the plan in the previous year, then they automatically
     accrue the full entitlement for the next year i.e. 24 days.
     If this condition is not met then the general rules apply.
     -------------------------------------------------*/
26.  IF((EmpHireDate>=PrevYrStartDate)AND
27.  (EmpHireDate<=PrevYrEndDate) AND
```

```
28.  (AccrualPeriodStartDate<=PrevYrEndDate) AND
29.  (AccrualPeriodEndDate>=PrevYrStartDate) AND
30.  (PEOPLE_BE_SCHOOL_LEAVER='YES')) THEN
     (
31.  PrevYrAccruedPTO=24
     )
32.  /*-----------------------------------------------
      General rule: Accrue 2 days per complete calendar month in
      the previous year when they were enrolled on the plan.
      If they joined the plan on or before the 15th of the month
      then the month is included. If they were terminated, then
      only include the month if it was the last day of the month.
      ------------------------------------------------------------*/
      ELSE
33.  /*-----------------------------------------------------
     They started accruing PTO after the end of the previous year
     or ceased to accrue PTO before the start of the previous
     year.
     ------------------------------------------------------------*/
34.  IF ((AccrualPeriodStartDate>PrevYrEndDate)OR
(AccrualPeriodEndDate<PrevYrStartDate))
     THEN
     (
35.  PeriodsToCount=0
     )
36.  /*-----------------------------------------------------
```

```
     They were eligible for accruing PTO at some point during

     the previous year.

     ----------------------------------------------------------*/

     ELSE

     (

37.  /*-------------------------------------------------

     Calculate the date in the previous year from which they

     started accruing PTO.

     ----------------------------------------------------*/

38.  PrevYrAccrualPeriodStartDate=GREATEST(PrevYrStartDate,
AccrualPeriodStartDate)

39.  /*-------------------------------------------------

     Calculate the date in the previous year from which they

     stopped accruing PTO.

     ----------------------------------------------------*/

40.  PrevYrAccrualPeriodEndDate=LEAST(PrevYrEndDate,
AccrualPeriodEndDate)

41.  /*-------------------------------------------------

     Their eligibility started on or before the 15th of the month
so include it as a month where they can accrue PTO.

     ----------------------------------------------------*/

42.  IF(TO_NUMBER(TO_CHAR(PrevYrAccrualPeriodStartDate, 'DD'))<=15)THEN

(

43.  StartPeriodNum=TO_NUMBER(TO_CHAR(PrevYrAccrualPeriodStartDate,
'MM'))

    )

44.  /*-------------------------------------------------------
```

```
     Their eligibility started after the 15th of the month so
exclude it as a month they can accrue PTO.

     ------------------------------------------------------------*/

    ELSE

    (

45.  StartPeriodNum=TO_NUMBER(TO_CHAR(PrevYrAccrualPeriodStartDate,
'MM'))+1

 )

46.  /*------------------------------------------------------------

     Their eligibility ended at the end of the month so include it

     as a month where they can accrue PTO

     ------------------------------------------------------------*/

47.  IF (TO_NUMBER(TO_CHAR(PrevYrAccrualPeriodStartDate,'DD'))<=15)THEN

 (

StartPeriodNum=TO_NUMBER(TO_CHAR(PrevYrAccrualPeriodStartDate,'MM'))

)

48.  /* ------------------------------------------------------------

      Their eligibility started after the 15th of the month so

      exclude it as a month where they can accrue PTO.

      ------------------------------------------------------------*/

    ELSE

    (

49.  StartPeriodNum=TO_NUMBER(TO_CHAR(PrevYrAccrualPeriodStartDate,
'MM'))+1

    )

50.  /* ------------------------------------------------------------

     Their eligibility ended at the end of the month so include it
```

```
     as a month where they can accrue PTO.

     ------------------------------------------------------------*/

51.   LastDayOfMonth=ADD_DAYS(TO_DATE(TO_CHAR(ADD_MONTHS
(PrevYrAccrualPeriodEndDate,1),'YYYYMM')||'01','YYYYMMDD'),-1)

52.  IF (PrevYrAccrualPeriodEndDate=LastDayOfMonth) THEN

    (

53.  EndPeriodNum=TO_NUMBER(TO_CHAR(PrevYrAccrualPeriodEndDate,'MM'))

    )

54.  /* ----------------------------------------------------------

     Their eligibility ended during the month so exclude it

     as a month where they can accrue PTO.

     ------------------------------------------------------------*/

    ELSE

    (

55.  EndPeriodNum=TO_NUMBER(TO_CHAR(PrevYrAccrualPeriodEndDate, 'MM'))-1

    )

56.  /*----------------------------------------------------------

    There were no periods in the previous year from which to

    accrue PTO.

    ------------------------------------------------------------*/

57.  IF((StartPeriodNum>12)OR(EndPeriodNum<1)OR
(StartPeriodNum>EndPeriodNum))

    THEN

    (

58.  PeriodsToCount=0

    )
```

```
59.  /*-----------------------------------------------------
     Calculate the number of periods in the previous year from
     which to accrue PTO.
      -----------------------------------------------------*/
    ELSE
    (
60.  PeriodsToCount=EndPeriodNum-StartPeriodNum+1
     )
     )
61.  /*-------------------------------------------------------------
     They have at least one calendar month in which they can
     accrue PTO so loop for each month and sum the total
     accrued PTO.
    -------------------------------------------------------------*/
62.  IF (PeriodsToCount>0)THEN
.    (
63.  /* -----------------------------------------------------
      Set up variables to be used by the subformula.
      -----------------------------------------------------*/
64.  E=set_number('total_accrued_pto',0)
65.  E=set_number('periods_to_count',PeriodsToCount)
66.  E = set_text('prev_year',PrevYr)
67.  E = set_number('period_count',1)
68.  E = set_number('period_num',StartPeriodNum)
69.  /*-----------------------------------------------------
```

```
     Loop for each eligible period within the previous year.
     ------------------------------------------------------------*/
70.  E = LOOP_CONTROL('PTO_BE_PREVYR_PERIOD_ACCRUAL')
71.  /*------------------------------------------------------------
     Get the total calculated accrued PTO.
     ------------------------------------------------------------*/
72.  PrevYrAccruedPTO=get_number('total_accrued_pto')
     )
73.  /*------------------------------------------------------------
     They have no calendar months in which they can accrue PTO
     ------------------------------------------------------------*/
     Else
    (
74.  PrevYrAccruedPTO=0
    )
    (
75.  /*------------------------------------------------------------
  Calculate the adjustment to be made to the total accrued PTO for
  this year i.e any holidays already taken, any carryover, and
  also any other extra entitlements/reductions (holidays
  bought/sold).
  ------------------------------------------------------------*/
76.  /*------------------------------------------------------------
     Calculate the start date in the current year over
     which to count absences.
```

```
     ------------------------------------------------------------*/

77.  CurrYrAbsenceStartDate=GREATEST(CurrYrStartDate,
AbsencePeriodSTartDate)

78.  /*-----------------------------------------------------------

     Calculate the end date in the current year over

     which to count absences.

     ------------------------------------------------------------*/

79.  CurrYrAbsenceEndDate=calculation_date

80.  /*-----------------------------------------------------------

    Calculate the outstanding accrued PTO for the current year.

    ------------------------------------------------------------*/

81.  total_accrued_pto=PrevYrAccruedPTO

82.  effective_start_date=CurrYrAbsenceStartDate

83.  effective_end_date=CurrYrAbsenceEndDate

84.  effective_end_date=CurrYrAbsenceEndDate

85.  RETURN total_accrued_pto, effective_start_date, effective_end_date,
accrual_end_date
```

**Looping Formula**

```
1.   /* ----------------------------------------------

    NAME : PTO_BE_PREVYR_PERIOD_ACCRUAL

    This formula calculates the accrued PTO for a particular

    calendar month.

    ----------------------------------------------*/

2.  TotalAccruedPTO=get_number('total_accrued_pto')

3.  PeriodsToCount=get_number('periods_to_count')

4.  PrevYr=get_text('prev_year')
```

```
5.  PeriodCount=get_number('period_count')

6.  PeriodNum=get_number('period_num')

8.   /*-----------------------------------------------

    Calculate the start and end dates of the period.

9.  Note: This is always a calendar month.

....-----------------------------------------------------*/

11. CurrPerStartDate=TO_DATE(PrevYr||LPAD(TO_CHAR(PeriodNum),2,'0')
||'01','YYYYMMDD')

12. CurrPerEndDate=ADD_DAYS(ADD_MONTHS(CurrPerStartDate,1),-1)

13  /*-----------------------------------------------------

    Calculate the accrued PTO for the period. In this example 2

    days are accrued per calendar month.

    */----------------------------------------------------------

14.  PeriodAccruedPTO=2

15.  /* ------------------------------------------------

     Add the accrued PTO for the period to the running total.

     -------------------------------------------------------*/

16. E=set_number('total_accrued_pto',TotalAccruedPTO+PeriodAccruedPTO)

17.  /*-----------------------------------------------

     There is at least one more period to process.

     ------------------------------------------------*/

18.  IF PeriodCount+1<=PeriodsToCount THEN

     (

19.  E=set_number('period_count',PeriodCount+1)

     E=set_number('period_num',PeriodNum+1)

20.  continue_processing_flag='Y'
```

```
      )
21.  ELSE
      (
22.  continue_processing_flag='N'
      )
23.  RETURN continue_processing_flag
```

#### Changing the Accrual Term Start and End Dates

The accrual term start date is set to 01 January at line 11 of the sample top level formula. To use another fixed date (such as 01 June) replace this line with the following:

```
PrevYrStartDate = to_date(PrevYr||'0601','YYYYMMDD')
```

The end date is set to 31st December at line 12 of the sample formula. To use another fixed date (such as 31st May) replace this line with the following:

```
PrevYrEndDate = to_date(PrevYr||'3105','YYYYMMDD')
```

#### Changing the Ceiling

In the sample top level formula, the ceiling is set to 0, which means that no ceiling is set. You can change the ceiling within the formula if you want to include a ceiling amount:

```
E = SET_NUMBER('CEILING', 20)
```

You can also set the ceiling outside the formula, using the Accrual Bands window or a user table.

#### Changing the School Leavers PTO Entitlement

The sample formula sets the amount of accrued PTO that a school leaver can receive in line 31. You can edit this amount to suit your business requirements. If the formula uses the School leaver calculation, the looping formula is not called because it does not need to loop through the eligible periods and calculate accrued PTO.

#### Changing the Eligibility Periods

The sample formula assumes that for an employee to be eligible to accrue PTO in a calendar month they must have joined on or before the 15th of the month. You can change this date in line 42. Lines 43 through 60 contain the logic to see if an employee is eligible to accrue the PTO for a month.

#### Changing the Amount of Accrued Time Allowed

The sample formula specifies a fixed amount of PTO that is allowed per calendar

month. This is set in line 14 of the looping formula. You can change this to suit your business requirements:

```
PeriodAccruedPTO=3
```

A further example of an accrual formula and suggested ways of implementing accrual plan rules is described in Sample Accrual Formula, page 1-157.

## Sample Accrual Formulas for Absence (Hungary)

This topic provides four sample accrual formulas for calculating the employee's holiday entitlements as required for the Hungarian Absence Report. You can record and report the entitlement amounts for the following holiday types:

- Base Holiday
- Additional holiday for bringing up children
- Other Additional Holiday
- Sickness Holiday

The following table lists the sample accrual formulas that you can use, as supplied, or use them as models to create your own, incorporating the rules required for your accrual plans.

***Hungarian Sample Formulas***

| Holiday Type | Formula | Description |
|---|---|---|
| Base Holiday | HU_BASE_HOLIDAY MULTIPLIER | This formula calculates the accrual rate for base holiday in a specific period. See: Sample Formula for Base Holiday, page 1-229 |
| Additional Holiday for bringing up children | HU_ADD_CHILD_CARE_HOLIDAY_MULTIPLIER | This formula calculates the Additional Holiday for bringing up children. See: Sample Formula for Additional Holiday, page 1-238 |

| Holiday Type | Formula | Description |
|---|---|---|
| Other Additional Holiday | HU_OTHER_ADD_HOLIDAY_MULTIPLIER | This formula calculates the Other Additional Holiday such as Youthful holiday and Additional Health Holiday. See: Sample Formula for Other Additional Holiday, page 1-250 |
| Sickness Holiday | HU_SICKNESS_HOLIDAY_MULTIPLIER | This formula calculates the Sickness Holiday. See: Sample Formula for Sickness Holiday, page 1-257 |

You can use the Carry Over Formula, HU_ABS_CARRYOVER_FORMULA to calculate any carry over for the above holiday entitlements. See: Sample Formula for Carry Over Absence, page 1-265

## Sample Formula for Base Holiday (Hungary)

Oracle HRMS provides the HU_BASE_HOLIDAY MULTIPLIER sample formula to calculate the accrual rate for Base Holiday in a specific period as required for the Hungarian Absence Report. The sample formula is for an accrual plan with the following rules:

- Entitlement for Base Holiday depends on the employee's age.
- Entitlement depends on the number of days worked in a week and takes in to account any work pattern changes within the leave year.
- Base holiday for new hires begins on their hire date.
- The formula calculates proportionately if the employee's birthday falls within the current year.

The following table lists the Base Holiday entitlements for employees based on their age and a five day work pattern.

***Base Holiday entitlements***

| Employee's Age | Base Holiday entitlement |
|---|---|
| Until age 24 | 20 working days holiday |
| From age 25 | 21 working days holiday |
| From age 28 | 22 working days holiday |
| From age 31 | 23 working days holiday |
| From age 33 | 24 working days holiday |
| From age 35 | 25 working days holiday |
| From age 37 | 26 working days holiday |
| From age 39 | 27 working days holiday |
| From age 41 | 28 working days holiday |
| From age 43 | 29 working days holiday |
| From age 45 | 30 working days holiday |

The following formula HU_BASE_HOLIDAY MULTIPLIER considers the employee's age and a five-day work pattern for employees to calculate the Base Holiday entitlement. The HU_BASE_HOLIDAY MULTIPLIER formula repeatedly calls another formula in a loop, HU_BASE_HOLIDAY_PERIOD_ACCRUAL to calculate the accrual for each period. You use the above formulas along with the carry over formula HU_ABS_HOLIDAY_CARRY_OVER. See: Sample Formula for Carry Over Absence, page 1-265

The top-level formula HU_BASE_HOLIDAY_MULTIPLIER and the looping formula HU_BASE_HOLIDAY_PERIOD_ACCRUAL are given below.

```
/* ------------------------------------------------------------------
  NAME : HU_BASE_HOLIDAY_MULTIPLER
  This formula calculates the total accrued base holiday for a
  specific period
  ------------------------------------------------------------------*/

DEFAULT FOR ACP_INELIGIBILITY_PERIOD_TYPE IS 'CM'
DEFAULT FOR ACP_INELIGIBILITY_PERIOD_LENGTH IS 0
DEFAULT FOR ACP_CONTINUOUS_SERVICE_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_ENROLLMENT_END_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_TERMINATION_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_ENROLLMENT_START_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_SERVICE_START_DATE IS '4712/12/31 00:00:00' (date)

INPUTS ARE Calculation_Date (date)

Accruing_Frequency  = ' '
Accruing_Multiplier = 0

E = SET_DATE('CALCULATION_DATE',Calculation_Date)

/* ------------------------------------------------------------------
   Set the payroll period, accruing frequency, and accruing multiplier
   based on the payroll.
   ------------------------------------------------------------------*/
Payroll_period = HU_PAYROLL_PERIODS(Calculation_Date
                                   ,Accruing_Frequency
                                  ,Accruing_Multiplier)

E = SET_TEXT('ACCRUING_FREQUENCY', Accruing_Frequency)
E = SET_NUMBER('ACCRUING_MULTIPLIER', Accruing_Multiplier)

Beginning_Of_Calculation_Year =
                      TO_DATE('0101'||TO_CHAR(Calculation_Date,'YYYY')
                             ,'DDMMYYYY')

IF Beginning_Of_Calculation_Year > Calculation_Date THEN
(
  Beginning_of_Calculation_Year =
                        ADD_MONTHS(Beginning_Of_Calculation_Year, -12)
)

/* ------------------------------------------------------------------
  Set the start and end dates of the first accrual period in the
  calculation year
  ------------------------------------------------------------------*/

E = SET_DATE('BEGINNING_OF_CALCULATION_YEAR'
                                      , Beginning_Of_Calculation_Year)

E = GET_PERIOD_DATES(Beginning_of_Calculation_Year,
                     Accruing_Frequency,
                     Beginning_Of_Calculation_Year,
                     Accruing_Multiplier)

First_Period_SD = GET_DATE('PERIOD_START_DATE')
First_Period_ED = GET_DATE('PERIOD_END_DATE')

/* ------------------------------------------------------------------
   Set the Calculation_Date to the Termination Date if not null
   ------------------------------------------------------------------*/

IF NOT (ACP_TERMINATION_DATE WAS DEFAULTED) OR
        NOT (ACP_ENROLLMENT_END_DATE WAS DEFAULTED) THEN
 (
  Early_End_Date = least(ACP_TERMINATION_DATE, ACP_ENROLLMENT_END_DATE)
```

```
  IF (Early_End_Date < Calculation_Date) THEN
   (
    Calculation_Date = Early_End_Date
   )
 )

/* ---------------------------------------------------------------------
   Get the last whole period prior to the Calculation_Date and ensure
   that it is within the year (if the Calculation_Date is the end of
   a period then use that period)
   ---------------------------------------------------------------------*/

E = GET_PERIOD_DATES(Calculation_Date
                    ,Accruing_Frequency
                    ,Beginning_of_Calculation_Year
                    ,Accruing_Multiplier)

Calculation_Period_SD  = GET_DATE('PERIOD_START_DATE')
Calculation_Period_ED  = GET_DATE('PERIOD_END_DATE')

/* ---------------------------------------------------------------------
   Set the Continuous Service Global Variable, whilst also
   ensuring that the continuous service date is before the Calculation
   Period
  ---------------------------------------------------------------------*/

IF (ACP_CONTINUOUS_SERVICE_DATE WAS DEFAULTED) THEN
    (
    E = SET_DATE('CONTINUOUS_SERVICE_DATE', ACP_SERVICE_START_DATE)
    )
ELSE IF(ACP_CONTINUOUS_SERVICE_DATE > Calculation_Period_SD) THEN
   (
    E = SET_DATE('CONTINUOUS_SERVICE_DATE'
                                        , ACP_CONTINUOUS_SERVICE_DATE)
   )
ELSE
   (
    E = SET_DATE('CONTINUOUS_SERVICE_DATE'
                                        , ACP_CONTINUOUS_SERVICE_DATE)
   )

Continuous_Service_Date = GET_DATE('CONTINUOUS_SERVICE_DATE')

First_Eligible_To_Accrue_Date  = Continuous_Service_Date

/*---------------------------------------------------------------------
   Determine the date on which accrued PTO may first be registered,
   i.e. the date on which the Ineligibility Period expires
   ---------------------------------------------------------------------*/

Accrual_Ineligibility_Expired_Date = First_Eligible_To_Accrue_Date

IF (ACP_INELIGIBILITY_PERIOD_LENGTH > 0) THEN
   (
   IF ACP_INELIGIBILITY_PERIOD_TYPE = 'BM' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                         ADD_MONTHS(Continuous_Service_Date,
                                    ACP_INELIGIBILITY_PERIOD_LENGTH*2)
      )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'F' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                          ADD_DAYS(Continuous_Service_Date,
                                   ACP_INELIGIBILITY_PERIOD_LENGTH*14)
      )
```

```
ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'CM' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                           ADD_MONTHS(Continuous_Service_Date,
                                      ACP_INELIGIBILITY_PERIOD_LENGTH)
      )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'LM' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                           ADD_DAYS(Continuous_Service_Date,
                                    ACP_INELIGIBILITY_PERIOD_LENGTH*28)
      )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'Q' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                          ADD_MONTHS(Continuous_Service_Date,
                                     ACP_INELIGIBILITY_PERIOD_LENGTH*3)
      )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'SM' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                          ADD_MONTHS(Continuous_Service_Date,
                                     ACP_INELIGIBILITY_PERIOD_LENGTH/2)
      )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'SY' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                          ADD_MONTHS(Continuous_Service_Date,
                                     ACP_INELIGIBILITY_PERIOD_LENGTH*6)
      )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'W' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                           ADD_DAYS(Continuous_Service_Date,
                                    ACP_INELIGIBILITY_PERIOD_LENGTH*7)
      )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'Y' THEN
      (
      Accrual_Ineligibility_Expired_Date =
                         ADD_MONTHS(Continuous_Service_Date,
                                    ACP_INELIGIBILITY_PERIOD_LENGTH*12)
      )

   IF Accrual_Ineligibility_Expired_Date > First_Eligible_To_Accrue_Date
   AND Calculation_Date < Accrual_Ineligibility_Expired_Date THEN
    (
     First_Eligible_To_Accrue_Date = Accrual_Ineligibility_Expired_Date
    )
   )

/* ---------------------------------------------------------------------
   Get the first full period following the
   First_Eligible_To_Accrue_Date
   (if it falls on the beginning of the period then use that period)
---------------------------------------------------------------------*/

IF First_Eligible_To_Accrue_Date > Beginning_Of_Calculation_Year THEN
(
  E = GET_PERIOD_DATES(First_Eligible_To_Accrue_Date
                      ,Accruing_Frequency
                      ,Beginning_Of_Calculation_Year
                      ,Accruing_Multiplier)

  First_Eligible_To_Accrue_Period_SD  = GET_DATE('PERIOD_START_DATE')
  First_Eligible_To_Accrue_Period_ED  = GET_DATE('PERIOD_END_DATE')
```

```
  IF (First_Eligible_To_Accrue_Period_SD > Calculation_Period_ED) THEN
     (
       Total_Accrued_PTO = 0
       E = PUT_MESSAGE('HR_52793_PTO_FML_ASG_INELIG')
    )
)
ELSE
(
  First_Eligible_To_Accrue_Period_SD  = First_Period_SD
  First_Eligible_To_Accrue_Period_ED  = First_Period_ED
)
/* ---------------------------------------------------------------------
   Determine the date on which PTO actually starts accruing based on
   Hire Date, Continuous Service Date and Plan Enrollment Start Date.
    ---------------------------------------------------------------------
*/

IF Continuous_Service_date = ACP_CONTINUOUS_SERVICE_DATE THEN
(
  Actual_Accrual_Start_Date = Continuous_service_Date
)
ELSE
(
  Actual_Accrual_Start_Date = GREATEST(Continuous_Service_Date,
                                       ACP_ENROLLMENT_START_DATE,
                                       First_Period_SD)
)
/* ---------------------------------------------------------------------
   Determine the actual start date and end date of the first accrual
   period to use in the accrual calculation. Get the start date and
   end dates of the accrual period in which the Actual Accrual
   Start Date falls.
   ---------------------------------------------------------------------*/
IF (Actual_Accrual_Start_Date > First_Period_SD AND
    Actual_Accrual_Start_Date > First_Eligible_To_Accrue_Period_SD) THEN
    (
     E = GET_PERIOD_DATES(Actual_Accrual_Start_Date,
                          Accruing_Frequency,
                          Beginning_Of_Calculation_Year,
                          Accruing_Multiplier)

     Accrual_Start_Period_SD = GET_DATE('PERIOD_START_DATE')
     Accrual_Start_Period_ED = GET_DATE('PERIOD_END_DATE')


/* ---------------------------------------------------------------------
   If the Actual Accrual Period is after the Calculation Period then
   end the processing.
   ---------------------------------------------------------------------*/
IF (Accrual_Start_Period_SD > Calculation_Period_ED) THEN
           (
           Total_Accrued_PTO = 0
           E = PUT_MESSAGE('HR_52797_PTO_FML_ACT_ACCRUAL')
           )
     )

ELSE IF (First_Eligible_To_Accrue_Period_SD > First_Period_SD) THEN
     (
          Accrual_Start_Period_SD = First_Eligible_To_Accrue_Period_SD
          Accrual_Start_Period_ED = First_Eligible_To_Accrue_Period_ED
     )
ELSE
     (
          Accrual_Start_Period_SD = First_Period_SD
          Accrual_Start_Period_ED = First_Period_ED
    )
```

```
 /* ---------------------------------------------------------------
    Now set up the information that will be used in when looping
    through the periods and call the accrual sub formula.
    ---------------------------------------------------------------*/
IF Calculation_Period_ED >= Accrual_Start_Period_ED THEN
(
    E = set_date('PERIOD_SD',Accrual_Start_Period_SD)
    E = set_date('PERIOD_ED',Accrual_Start_Period_ED)
    E = set_date('LAST_PERIOD_SD',Calculation_Period_SD)
    E = set_date('LAST_PERIOD_ED',Calculation_Period_ED)
    E = set_number('TOTAL_ACCRUED_PTO',0)

    E = LOOP_CONTROL('HU_BASE_HOLIDAY_PERIOD_ACCRUAL')

    Total_Accrued_PTO = ROUND(get_number('TOTAL_ACCRUED_PTO'))
)

IF Accrual_Start_Period_SD <= Calculation_Period_SD THEN
(
    Accrual_end_date = Calculation_Period_ED
)

Effective_Start_Date = Accrual_Start_Period_SD
Effective_End_Date   = Calculation_Period_ED

IF Effective_Start_Date >= Effective_End_Date THEN
(
    Effective_Start_Date = Effective_End_Date
)

RETURN Total_Accrued_PTO
      ,Effective_start_date
      ,Effective_end_date
      ,Accrual_end_date
```

**Looping Formula**

```
/* ------------------------------------------------------------------
  NAME : HU_BASE_HOLIDAY_PERIOD_ACCRUAL
  This formula calculates the number of base holiday accrued for a
  particular period.
  ------------------------------------------------------------------*/

/*------------------------------------------------------------------
    Get the global variable to be used in this formula
  ------------------------------------------------------------------*/

DEFAULT FOR ACP_TERMINATION_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR EMP_HIRE_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR person_dob  IS '4712/12/31 00:00:00' (date)

Continuous_Service_Date = GET_DATE('CONTINUOUS_SERVICE_DATE')
Total_Accrued_PTO       = GET_NUMBER('TOTAL_ACCRUED_PTO')
Period_SD               = GET_DATE('PERIOD_SD')
Period_ED               = GET_DATE('PERIOD_ED')
Last_Period_SD          = GET_DATE('LAST_PERIOD_SD')
Last_Period_ED          = GET_DATE('LAST_PERIOD_ED')
Accruing_Frequency      = GET_TEXT('ACCRUING_FREQUENCY')
Accruing_Multiplier     = GET_NUMBER('ACCRUING_MULTIPLIER')
Beginning_of_Calculation_Year
                        = GET_DATE('BEGINNING_OF_CALCULATION_YEAR')
Calculation_Date        = GET_DATE('CALCULATION_DATE')

Accrual_Rate = 0

/* ------------------------------------------------------------------
   Get the person date of birth and compute the age.
   ------------------------------------------------------------------*/

Person_dob = HU_PERSON_DOB(Calculation_Date)

Age        = FLOOR(MONTHS_BETWEEN(Period_ED,Person_dob)/12)

/* ------------------------------------------------------------------
   Set the payroll period, accruing frequency and accruing multiplier
   based on the payroll.
   ------------------------------------------------------------------*/

Payroll_period = HU_PAYROLL_PERIODS(Calculation_Date
                                    ,Accruing_Frequency
                                  ,Accruing_Multiplier)
/* ------------------------------------------------------------------
   Set period start date and period end date as employee hire date and
   employee termination date if the hire date and termination date
   falls with in the calculation period.
   ------------------------------------------------------------------*/

IF EMP_HIRE_DATE > Period_SD AND EMP_HIRE_DATE < Period_ED THEN
   Period_SD = EMP_HIRE_DATE

IF ACP_TERMINATION_DATE > Period_SD
                          AND ACP_TERMINATION_DATE < Period_ED THEN
   Period_ED =  ACP_TERMINATION_DATE

DOB = TO_DATE(TO_CHAR(Person_dob,'DD/MM/')
             ||TO_CHAR(PERIOD_SD,'YYYY'),'DD/MM/YYYY')

/* ------------------------------------------------------------------
   Set the accrual rate based on the age of the employee and on five
   day work pattern.
   ------------------------------------------------------------------*/

IF   (DOB >= PERIOD_SD   AND DOB <= PERIOD_ED) AND
```

```
(AGE = 25 OR AGE = 28 OR AGE = 31 OR AGE = 33 OR AGE = 35  OR
      AGE = 37 OR AGE = 39 OR AGE = 41 OR AGE = 43 OR AGE = 45) THEN
(
    X = HU_ABS_GET_WORKING_DAYS(PERIOD_SD,ADD_DAYS(DOB,-1))
    Y = HU_ABS_GET_WORKING_DAYS(DOB,Period_ED)

    IF  Age = 25 THEN
        Accrual_Rate = ((20/260)* X +(21/260)* Y)
    ELSE IF Age = 28 THEN
        Accrual_Rate = ((21/260)* X +(22/260)* Y)
    ELSE IF Age = 31 THEN
        Accrual_Rate = ((22/260)* X +(23/260)* Y)
    ELSE IF Age = 33 THEN
        Accrual_Rate = ((23/260)* X +(24/260)* Y)
    ELSE IF Age = 35 THEN
        Accrual_Rate = ((24/260)* X +(25/260)* Y)
    ELSE IF Age = 37 THEN
        Accrual_Rate = ((25/260)* X +(26/260)* Y)
    ELSE IF Age = 39 THEN
        Accrual_Rate = ((26/260)* X +(27/260)* Y)
    ELSE IF Age = 41 THEN
        Accrual_Rate = ((27/260)* X +(28/260)* Y)
    ELSE IF Age = 43 THEN
        Accrual_Rate = ((28/260)* X +(29/260)* Y)
    ELSE IF Age = 45 THEN
        Accrual_Rate = ((29/260)* X +(30/260)* Y)
    )
ELSE
(
    X = HU_ABS_GET_WORKING_DAYS(PERIOD_SD,PERIOD_ED)

    IF  Age <= 24 THEN
        Accrual_Rate = ((20/260)*X)
    ELSE IF (Age >= 25 AND Age < 28) THEN
        Accrual_Rate = ((21/260)*X)
    ELSE IF (Age >= 28 AND Age < 31) THEN
        Accrual_Rate = ((22/260)*X)
    ELSE IF (Age >= 31 AND Age < 33) THEN
        Accrual_Rate = ((23/260)*X)
    ELSE IF (Age >= 33 AND Age < 35) THEN
        Accrual_Rate = ((24/260)*X)
    ELSE IF (Age >= 35 AND Age < 37) THEN
        Accrual_Rate = ((25/260)*X)
    ELSE IF (Age >= 37 AND Age < 39) THEN
        Accrual_Rate = ((26/260)*X)
    ELSE IF (Age >= 39 AND Age < 41) THEN
        Accrual_Rate = ((27/260)*X)
    ELSE IF (Age >= 41 AND Age < 43) THEN
        Accrual_Rate = ((28/260)*X)
    ELSE IF (Age >= 43 AND Age < 45) THEN
        Accrual_Rate = ((29/260)*X)
    ELSE
        Accrual_Rate = ((30/260)*X)
)

Period_Accrued_PTO = Accrual_Rate

E = SET_NUMBER('TOTAL_ACCRUED_PTO', Total_Accrued_PTO
                                      +Period_Accrued_PTO)

/* ---------------------------------------------------------------
    Establish whether the current period is the last one, if so end
    the processing, otherwise get the  next period.
   ---------------------------------------------------------------*/

IF Period_SD >= Last_Period_SD THEN
```

```
(
    Continue_Processing_Flag = 'N'
    )
ELSE
   (
    E = GET_PERIOD_DATES(ADD_DAYS(Period_ED,1),
                         Accruing_Frequency,
                         Beginning_of_Calculation_Year,
                         Accruing_Multiplier)

    E = SET_DATE('PERIOD_SD', GET_DATE('PERIOD_START_DATE'))
    E = SET_DATE('PERIOD_ED', GET_DATE('PERIOD_END_DATE'))

   Continue_Processing_Flag = 'Y'
   )

Return Continue_Processing_Flag
```

## Sample Formula for Additional Holiday (Hungary)

Oracle HRMS provides the HU_ADD_CHILD_CARE_HOLIDAY_MULTIPLIER sample formula to calculate the accrual rate for Additional Holiday for bringing up children in a specific period as required for the Hungarian Absence Report. The sample formula is for an accrual plan with the following rules:

- Entitlement depends on the age and number of children (given in the table below)
- Both parents work and share the entitlement. This eligibility is based on the value in the Holiday for Child Care field in the Further Contact Relationship Info on the Contact window.
- Additional Holiday for bringing up children for new hires depends on Hire Date.
- Entitlement begins from the birth date of the child, if a child is born during the leave year.

The following table lists the Additional Holiday entitlements for employees based on the number of children.

***Additional Holiday entitlements***

| Number of Children | Additional Holiday entitlement |
|---|---|
| After 1 child | 2 working days |
| After 2 children | 4 working days |
| After more than 2 children | 7 working days |

The HU_ADD_CHILD_CARE_HOLIDAY_MULTIPLIER formula calls another formula

HU_ADD_CHILD_CARE_HOLIDAY_PERIOD_ACCRUALcalculate the accrual for each period. You use the sample formulas along with the carry over formula HU_ABS_HOLIDAY_CARRY_OVER. See: Sample Formula for Carry Over Absence, page 1-265

The sample HU_ADD_CHILD_CARE_HOLIDAY_MULTIPLIER and the looping HU_ADD_CHILD_CARE_HOLIDAY_PERIOD_ACCRUAL formula are given below:

```
/* ------------------------------------------------------------------
    NAME : HU_ADD_CHILD_CARE_HOLIDAY_MULTIPLIER

    This formula calculates the total accrued additional child care
    holiday for a specific period.
   ------------------------------------------------------------------*/

DEFAULT FOR ACP_INELIGIBILITY_PERIOD_TYPE IS 'CM'
DEFAULT FOR ACP_INELIGIBILITY_PERIOD_LENGTH IS 0
DEFAULT FOR ACP_CONTINUOUS_SERVICE_DATE IS '4712/12/31 00:00:00'(DATE)
DEFAULT FOR ACP_ENROLLMENT_END_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR ACP_TERMINATION_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR ACP_ENROLLMENT_START_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR ACP_SERVICE_START_DATE IS '4712/12/31 00:00:00' (DATE)

INPUTS ARE
Calculation_Date (DATE)

Accruing_Frequency = ' '
Accruing_Multiplier = 0

/* ------------------------------------------------------------------
   Set the payroll period, accruing frequency, and accruing multiplier
   based on the payroll.
   ------------------------------------------------------------------*/

No_of_Payroll_Periods = HU_PAYROLL_PERIODS (Calculation_Date
                                           ,Accruing_Frequency
                                           ,Accruing_Multiplier)

E = SET_TEXT('ACCRUING_FREQUENCY', Accruing_Frequency)
E = SET_NUMBER('ACCRUING_MULTIPLIER', Accruing_Multiplier)

/* ------------------------------------------------------------------
  Calculate the start and end dates of the current leave year.
  ------------------------------------------------------------------*/


Beginning_Of_Calculation_Year=TO_DATE('0101'||to_char(Calculation_Date
                                    ,'YYYY'),'DDMMYYYY')
End_Of_Calculation_Year = TO_DATE('3112'||to_char(Calculation_Date
                                            ,'YYYY'),'DDMMYYYY')

/* ------------------------------------------------------------------
  Set the start and end dates of the first accrual period in the
  calculation year.
  ------------------------------------------------------------------*/

IF Beginning_Of_Calculation_Year > Calculation_Date THEN
(
  Beginning_of_Calculation_Year =
                       ADD_MONTHS(Beginning_Of_Calculation_Year, -12)
)

E = SET_DATE('BEGINNING_OF_CALCULATION_YEAR'
            ,Beginning_Of_Calculation_Year)

E = GET_PERIOD_DATES(Beginning_of_Calculation_Year
                    ,Accruing_Frequency
                    ,Beginning_Of_Calculation_Year
                    ,Accruing_Multiplier)

First_Period_SD = GET_DATE('PERIOD_START_DATE')
First_Period_ED = GET_DATE('PERIOD_END_DATE')

E = GET_PERIOD_DATES(End_Of_Calculation_Year
```

```
,Accruing_Frequency
                    ,Beginning_Of_Calculation_Year
                    ,Accruing_Multiplier)

Last_Period_SD = GET_DATE('PERIOD_START_DATE')
Last_Period_ED = GET_DATE('PERIOD_END_DATE')

/* ------------------------------------------------------------------
   Set the Calculation_Date to the Termination Date if not null
   ------------------------------------------------------------------*/

IF NOT (ACP_TERMINATION_DATE WAS DEFAULTED) OR
   NOT (ACP_ENROLLMENT_END_DATE WAS DEFAULTED) THEN
(
  Early_End_Date = LEAST(ACP_TERMINATION_DATE,ACP_ENROLLMENT_END_DATE)

  IF (Early_End_Date < First_Period_SD) THEN
  (
    Total_Accrued_PTO = 0
    E = PUT_MESSAGE('HR_52794_PTO_FML_ASG_TER')
   )
  IF (Early_End_Date < Last_Period_ED) THEN
  (
    E = GET_PERIOD_DATES(Early_End_Date
                         ,Accruing_Frequency
                         ,Beginning_Of_Calculation_Year
                         ,Accruing_Multiplier)

    Last_Period_SD = GET_DATE('PERIOD_START_DATE')
    Last_Period_ED = GET_DATE('PERIOD_END_DATE')
   )
  IF (Early_End_Date < Calculation_Date) THEN
  (
    Calculation_Date = Early_End_Date
  )
)

/* ------------------------------------------------------------------
   Get the last whole period prior to the Calculation Date and ensure
   that it is within the  Year (if the Calculation Date is the End of
   a Period then use that period)
   ------------------------------------------------------------------*/

E = GET_PERIOD_DATES(Calculation_Date
                    ,Accruing_Frequency
                    ,Beginning_of_Calculation_Year
                    ,Accruing_Multiplier)

Calculation_Period_SD  = GET_DATE('PERIOD_START_DATE')
Calculation_Period_ED = GET_DATE('PERIOD_END_DATE')

IF (Calculation_Period_ED < First_Period_SD) THEN
   (
    Total_Accrued_PTO = 0
    E = PUT_MESSAGE('HR_52795_PTO_FML_CALC_DATE')
    )

/* ------------------------------------------------------------------
   Set the Continuous Service Global Variable, whilst also
   ensuring that the continuous service date is before the Calculation
   Period
  ------------------------------------------------------------------*/

IF (ACP_CONTINUOUS_SERVICE_DATE WAS DEFAULTED) THEN
(
    E = SET_DATE('CONTINUOUS_SERVICE_DATE', ACP_SERVICE_START_DATE)
```

```
)
ELSE IF(ACP_CONTINUOUS_SERVICE_DATE > Calculation_Period_ED) THEN
(
    Total_Accrued_PTO = 0
    E = PUT_MESSAGE('HR_52796_PTO_FML_CSD')
    E = SET_DATE('CONTINUOUS_SERVICE_DATE'
                ,ACP_CONTINUOUS_SERVICE_DATE)
)
ELSE IF(ACP_CONTINUOUS_SERVICE_DATE > First_Period_SD) THEN
(
    E = GET_PERIOD_DATES(ACP_CONTINUOUS_SERVICE_DATE
                         ,Accruing_Frequency
                         ,Beginning_Of_Calculation_Year
                         ,Accruing_Multiplier)

    First_Period_SD = GET_DATE('PERIOD_START_DATE')
    First_Period_ED = GET_DATE('PERIOD_END_DATE')
)
ELSE
(
    E = SET_DATE('CONTINUOUS_SERVICE_DATE'
                , ACP_CONTINUOUS_SERVICE_DATE)
)

Continuous_Service_Date = GET_DATE('CONTINUOUS_SERVICE_DATE')

First_Eligible_To_Accrue_Date  = Continuous_Service_Date

/*---------------------------------------------------------------------
   Determine the date on which accrued PTo may first be registered,
   i.e the date on which the   Ineligibility Period expires
   ---------------------------------------------------------------------*/

Accrual_Ineligibility_Expired_Date = First_Eligible_To_Accrue_Date

IF (ACP_INELIGIBILITY_PERIOD_LENGTH > 0) THEN
(
   IF ACP_INELIGIBILITY_PERIOD_TYPE = 'BM' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_MONTHS(Continuous_Service_Date
                                ,ACP_INELIGIBILITY_PERIOD_LENGTH*2)
   )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'F' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_DAYS(Continuous_Service_Date
                             ,ACP_INELIGIBILITY_PERIOD_LENGTH*14)
   )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'CM' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_MONTHS(Continuous_Service_Date
                                ,ACP_INELIGIBILITY_PERIOD_LENGTH)
   )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'LM' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_DAYS(Continuous_Service_Date
                             ,ACP_INELIGIBILITY_PERIOD_LENGTH*28)
   )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'Q' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_MONTHS(Continuous_Service_Date
                                ,ACP_INELIGIBILITY_PERIOD_LENGTH*3)
```

```
)
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'SM' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_MONTHS(Continuous_Service_Date
                                ,ACP_INELIGIBILITY_PERIOD_LENGTH/2)
   )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'SY' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_MONTHS(Continuous_Service_Date
                                ,ACP_INELIGIBILITY_PERIOD_LENGTH*6)
   )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'W' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_DAYS(Continuous_Service_Date
                             ,ACP_INELIGIBILITY_PERIOD_LENGTH*7)
   )
   ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'Y' THEN
   (
      Accrual_Ineligibility_Expired_Date =
                     ADD_MONTHS(Continuous_Service_Date
                                ,ACP_INELIGIBILITY_PERIOD_LENGTH*12)
   )

   IF Accrual_Ineligibility_Expired_Date
                > First_Eligible_To_Accrue_Date
      AND Calculation_Date
                < Accrual_Ineligibility_Expired_Date THEN
  (
      First_Eligible_To_Accrue_Date =
                                Accrual_Ineligibility_Expired_Date
  )
)

/* ---------------------------------------------------------------------
 Get the first full period following the First_Eligible_To_Accrue_Date
 (if it falls on the beginning of the period then use that period)
--------------------------------------------------------------------- */

IF First_Eligible_To_Accrue_Date > Beginning_Of_Calculation_Year THEN
(
   E = GET_PERIOD_DATES(First_Eligible_To_Accrue_Date
                       ,Accruing_Frequency
                       ,Beginning_Of_Calculation_Year
                       ,Accruing_Multiplier)

   First_Eligible_To_Accrue_Period_SD  = GET_DATE('PERIOD_START_DATE')
   First_Eligible_To_Accrue_Period_ED  = GET_DATE('PERIOD_END_DATE')

   IF (First_Eligible_To_Accrue_Period_SD > Calculation_period_ED)THEN
   (
      Total_Accrued_PTO = 0
       E = PUT_MESSAGE('HR_52793_PTO_FML_ASG_INELIG')
   )
)
ELSE
(
   First_Eligible_To_Accrue_Period_SD  = First_Period_SD
   First_Eligible_To_Accrue_Period_ED  = First_Period_ED
)
/* ---------------------------------------------------------------------
   Determine the date on which PTO actually starts accruing based on
   Hire Date, Continuous Service Date and plan Enrollment Start Date.
    ---------------------------------------------------------------------
```

```
*/

IF Continuous_Service_date = ACP_CONTINUOUS_SERVICE_DATE THEN
(
  Actual_Accrual_Start_Date = Continuous_service_Date
)
ELSE
(
  Actual_Accrual_Start_Date = GREATEST(Continuous_Service_Date,
                                       ACP_ENROLLMENT_START_DATE,
                                       First_Period_SD)
)

/* ---------------------------------------------------------------------
       Determine the actual start of the accrual calculation
---------------------------------------------------------------------*/
IF (Actual_Accrual_Start_Date > First_Period_SD AND
    Actual_Accrual_Start_Date > First_Eligible_To_Accrue_Period_SD)
THEN
(
     E = GET_PERIOD_DATES(Actual_Accrual_Start_Date
                         ,Accruing_Frequency
                         ,Beginning_Of_Calculation_Year
                         ,Accruing_Multiplier)

     Accrual_Start_Period_SD = GET_DATE('PERIOD_START_DATE')
     Accrual_Start_Period_ED = GET_DATE('PERIOD_END_DATE')

   /*---------------------------------------------------------------------
   If the Actual Accrual Period is after the Calculation Period then
   end the processing.
   --------------------------------------------------------------------- */
     IF (Accrual_Start_Period_SD > Calculation_period_ED) THEN
     (
         Total_Accrued_PTO = 0
         E = PUT_MESSAGE('HR_52797_PTO_FML_ACT_ACCRUAL')
     )
)
ELSE IF (First_Eligible_To_Accrue_Period_SD > First_Period_SD) THEN
(
    Accrual_Start_Period_SD = First_Eligible_To_Accrue_Period_SD
    Accrual_Start_Period_ED = First_Eligible_To_Accrue_Period_ED
)
ELSE
(
    Accrual_Start_Period_SD = First_Period_SD
    Accrual_Start_Period_ED = First_Period_ED
)

/*---------------------------------------------------------------------
  Now set up the information that will be used in when looping through
  the periods
  --------------------------------------------------------------------- */
IF Last_period_ED >= Accrual_Start_Period_ED THEN
(
   E = SET_DATE('PERIOD_SD',Accrual_Start_Period_SD)
   E = SET_DATE('PERIOD_ED',Accrual_Start_Period_ED)
   E = SET_DATE('LAST_PERIOD_SD',Calculation_period_SD)
   E = SET_DATE('LAST_PERIOD_ED',Calculation_period_ED)
   E = SET_NUMBER('TOTAL_ACCRUED_PTO',0)

   E = LOOP_CONTROL('HU_ADD_CHILD_CARE_HOLIDAY_PERIOD_ACCRUAL')

   Total_Accrued_PTO = ROUND(GET_NUMBER('TOTAL_ACCRUED_PTO'))
)
```

```
IF Accrual_Start_Period_SD <= Calculation_period_ED THEN
(
  Accrual_end_date = Calculation_period_ED
)

Effective_Start_Date = Accrual_Start_Period_SD
Effective_End_Date = Calculation_period_ED

IF Effective_Start_Date >= Effective_End_Date THEN
(
  Effective_Start_Date = Effective_End_Date
)

RETURN Total_Accrued_PTO
      ,Effective_start_date
      ,Effective_end_date
      ,Accrual_end_date
```

**Looping Formula**

```
/* ----------------------------------------------------------------
    NAME : HU_ADD_CHILD_CARE_HOLIDAY_PERIOD_ACCRUAL
    This formula calculates the amount of PTO accrued for a particular
    period
   ----------------------------------------------------------------*/

/*----------------------------------------------------------------
    Get the global variable to be used in this formula
 ----------------------------------------------------------------*/

DEFAULT FOR ACP_TERMINATION_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR ACP_SERVICE_START_DATE IS '4712/12/31 00:00:00' (DATE)

Continuous_Service_Date = GET_DATE('CONTINUOUS_SERVICE_DATE')
Total_Accrued_PTO = GET_NUMBER('TOTAL_ACCRUED_PTO')
Period_SD = GET_DATE('PERIOD_SD')
Period_ED = GET_DATE('PERIOD_ED')
Last_Period_SD = GET_DATE('LAST_PERIOD_SD')
Last_Period_ED = GET_DATE('LAST_PERIOD_ED')
Termination_date = GET_DATE('ACP_TERMINATION_DATE')
Hire_date = GET_DATE('ACP_SERVICE_START_DATE')

Accruing_Frequency = GET_TEXT('ACCRUING_FREQUENCY')
Accruing_Multiplier = GET_NUMBER('ACCRUING_MULTIPLIER')
Beginning_of_Calculation_Year =
                         GET_DATE('BEGINNING_OF_CALCULATION_YEAR')

First_Child_Date_of_birth = TO_DATE('01-01-4712','dd-mm-yyyy')
Second_Child_Date_of_birth = TO_DATE('01-01-4712','dd-mm-yyyy')
Third_Child_Date_of_birth = TO_DATE('01-01-4712','dd-mm-yyyy')

/* ----------------------------------------------------------------
   Set the payroll period, accruing frequency, and accruing multiplier
   based on the payroll.
   ----------------------------------------------------------------*/

No_of_Payroll_Periods = HU_PAYROLL_PERIODS (Period_SD
                                           ,Accruing_Frequency
                                           ,Accruing_Multiplier)

No_Of_Children_less_16 = 0
No_Of_Children_16 = 0

/* ----------------------------------------------------------------
  Retrieve child information for the current employee.
  ---------------------------------------------------------------- */

E = HU_ABS_GET_CHILD_INFO(Period_SD, Period_ED
                         ,No_Of_Children_less_16
                         ,No_Of_Children_16
                         ,First_Child_Date_of_birth
                         ,Second_Child_Date_of_birth
                         ,Third_Child_Date_of_birth)

Total_children = No_Of_Children_less_16 + No_Of_Children_16
Period_Accrued_PTO = 0

Total_days = DAYS_BETWEEN(Period_ED , Period_SD)
days_valid = DAYS_BETWEEN(Period_ED , Period_SD)
Accrual_Rate = 0

/* ----------------------------------------------------------------
  Set accrual rate based on number of children and their age.
  ---------------------------------------------------------------- */

IF Total_children = 1 THEN
```

```
(  Accrual_Rate = 2 )
ELSE IF Total_children = 2 THEN
(  Accrual_Rate = 4  )
ELSE IF Total_children > 2 THEN
(  Accrual_Rate = 7)

Accrual_rate_per_period = Accrual_Rate/No_of_Payroll_Periods

/* ---------------------------------------------------------------------
   Days worked prorated based on hire date and termination date of
   employee.
  ---------------------------------------------------------------------*/

IF Hire_date >= Period_SD
   AND Hire_date <= Period_ED
   AND Termination_date >= Period_SD
   AND Termination_date <= Period_ED THEN
(
   days_valid = DAYS_BETWEEN(Termination_date,Hire_date)
)
ELSE IF Hire_date >= Period_SD
    AND Hire_date <= Period_ED  THEN
(
   days_valid = DAYS_BETWEEN(Period_ED,Hire_date)
)
ELSE IF Termination_date >= Period_SD
    AND Termination_date <= Period_ED THEN
(
   days_valid = DAYS_BETWEEN(Termination_date,Period_SD)
)
Accrual_rate_per_period = Accrual_rate_per_period
                          *(days_valid/ Total_days)


/* ---------------------------------------------------------------------
    Calculate the Amount Accrued this Period-
   ---------------------------------------------------------------------*/
IF No_Of_Children_16 = 0 or No_Of_Children_less_16 > 2 THEN
(
  Period_Accrued_PTO = Accrual_rate_per_period
)
ELSE
(
  IF No_Of_Children_16 = 1 THEN
  (
     days_valid = DAYS_BETWEEN(First_Child_Date_of_birth,Period_SD)

     Period_Accrued_PTO =  Accrual_rate_per_period
                             *(days_valid / Total_days )
     IF No_Of_Children_less_16  = 1 THEN
     (
       Period_Accrued_PTO = Period_Accrued_PTO
                            +(2*((Total_days-days_valid)/Total_days))
                            /No_of_Payroll_Periods
     )
     IF No_Of_Children_less_16 = 2 THEN
     (
       Period_Accrued_PTO = Period_Accrued_PTO
                            +(4*((Total_days-days_valid)/Total_days))
                            /No_of_Payroll_Periods
     )
   )
   ELSE IF No_Of_Children_16 = 2 THEN
   (
     IF No_Of_Children_less_16 > 1 THEN
     (
```

```
days_valid = DAYS_BETWEEN(Second_Child_Date_of_birth,Period_SD)

        Period_Accrued_PTO =  Accrual_rate_per_period
                              *(days_valid / Total_days)
        Period_Accrued_PTO = Period_Accrued_PTO
                              +(4*((Total_days-days_valid)/Total_days))
                              /No_of_Payroll_Periods
     )
     ELSE IF No_Of_Children_less_16 > 0 THEN
     (
       days_valid = DAYS_BETWEEN(First_Child_Date_of_birth,Period_SD)

       Period_Accrued_PTO =  Accrual_rate_per_period
                              * (days_valid / Total_days)

        days_valid = DAYS_BETWEEN(Second_Child_Date_of_birth
                                  ,First_Child_Date_of_birth)

        Period_Accrued_PTO = Period_Accrued_PTO
                              +(4 *(days_valid / Total_days))
                              /No_of_Payroll_Periods

        days_valid = DAYS_BETWEEN(Period_ED
                                  ,Second_Child_Date_of_birth)

        Period_Accrued_PTO = Period_Accrued_PTO
                              +(2 *(days_valid / Total_days))
                              /No_of_Payroll_Periods
     )
     ELSE IF No_Of_Children_less_16 = 0 THEN
     (
       days_valid = DAYS_BETWEEN(First_Child_Date_of_birth,Period_SD)

       Period_Accrued_PTO =  Accrual_rate_per_period
                              *(days_valid / Total_days)

        days_valid = DAYS_BETWEEN(Second_Child_Date_of_birth
                                  ,First_Child_Date_of_birth)

        Period_Accrued_PTO = Period_Accrued_PTO +
                            (2 * (days_valid / Total_days))
                            /No_of_Payroll_Periods
      )
   )
   ELSE IF No_Of_Children_16 = 3 THEN
   (
     IF No_Of_Children_less_16 > 1 THEN
     (
 days_valid = DAYS_BETWEEN(Third_Child_Date_of_birth,Period_SD)

        Period_Accrued_PTO =  Accrual_rate_per_period
                               *(days_valid / Total_days)
        Period_Accrued_PTO = Period_Accrued_PTO
                              +(4*((Total_days-days_valid)
                              / Total_days))
                              /No_of_Payroll_Periods
      )
      ELSE IF No_Of_Children_less_16 > 0 THEN
      (
       days_valid = DAYS_BETWEEN(Second_Child_Date_of_birth
                                  ,Period_SD)
        Period_Accrued_PTO = Accrual_rate_per_period
                             *(days_valid / Total_days)

        days_valid = DAYS_BETWEEN(Third_Child_Date_of_birth
                                  ,Second_Child_Date_of_birth)
```

```
        Period_Accrued_PTO = Period_Accrued_PTO
                             +(4 * (days_valid / Total_days))
                             /No_of_Payroll_Periods

 days_valid = DAYS_BETWEEN(Period_ED,Third_Child_Date_of_birth)


        Period_Accrued_PTO = Period_Accrued_PTO
                             +(2 * (days_valid / Total_days))
                             /No_of_Payroll_Periods
      )
      ELSE IF No_Of_Children_less_16 = 0 THEN
      (
 days_valid = DAYS_BETWEEN(First_Child_Date_of_birth,Period_SD)

 Period_Accrued_PTO =  Accrual_rate_per_period
                              *(days_valid / Total_days)

        days_valid = DAYS_BETWEEN(Second_Child_Date_of_birth
                                  ,First_Child_Date_of_birth)

        Period_Accrued_PTO = Period_Accrued_PTO
                             +(4 * (days_valid / Total_days))
                             /No_of_Payroll_Periods

       days_valid = DAYS_BETWEEN(Third_Child_Date_of_birth
                                  ,Second_Child_Date_of_birth)

        Period_Accrued_PTO = Period_Accrued_PTO
                             +(2 * (days_valid / Total_days))
                             /No_of_Payroll_Periods
      )
    )
 )


/*---------------------------------------------------------------------
    Set the Running Total
  ---------------------------------------------------------------------*/

E = SET_NUMBER('TOTAL_ACCRUED_PTO'
              ,Total_Accrued_PTO + Period_Accrued_PTO)


/* ---------------------------------------------------------------------
    Establish whether the current period is the last one, if so end
    the processing, otherwise get the  next period
   ---------------------------------------------------------------------*/

IF Period_SD >= Last_Period_SD THEN
(
    Continue_Processing_Flag = 'N'
)
ELSE
(
    E = GET_PERIOD_DATES(ADD_DAYS(Period_ED,1)
                        ,Accruing_Frequency
                        ,Beginning_of_Calculation_Year
                        ,Accruing_Multiplier)

    E = SET_DATE('PERIOD_SD'
                ,GET_DATE('PERIOD_START_DATE'))
    E = SET_DATE('PERIOD_ED'
                ,GET_DATE('PERIOD_END_DATE'))

   Continue_Processing_Flag = 'Y'
```

```
)
RETURN Continue_Processing_Flag
```

## Sample Formula for Other Additional Holiday (Hungary)

Oracle HRMS provides the HU_OTHER_ADD_HOLIDAY_MULTIPLIER formula to calculate the Other Additional Holiday as required for the Hungarian Absence Report. The Other Additional Holiday comprises of Youthful Holiday and Additional Health Holiday. The sample formula is for an accrual plan with the following rules:

- Employee under 18 years of age entitled to an additional 5 days holiday.
- Blind employee entitled to an additional 5 days holiday. This eligibility is based on the information recorded on the Blind field on the Disabilities window.
- Employee working underground or exposed to radiation entitled to an additional 5 days holiday. This eligibility is based on the information recorded on the Additional Health Holiday field on the Further Job Info in the Job window.
- Entitlement depends on the number of days worked in a week. The formula takes into account any work pattern changes.
- Entitlement begins or ends from the date of change, if the shift or disability change during the leave year.
- Other Additional Holiday for new hires begins on their Hire Date.

The top level formula, HU_OTHER_ADD_HOLIDAY_MULTIPLIER calls another formula called HU_OTHER_ADD_HOLIDAY_PERIOD_ACCRUAL calculate the accrual for each period. You use the formulas along with the carry over formula HU_ABS_HOLIDAY_CARRY_OVER. See: Sample Formula for Carry Over Absence, page 1-265

The top-level formula, HU_OTHER_ADD_HOLIDAY_MULTIPLIER and the looping formula HU_OTHER_ADD_HOLIDAY_PERIOD_ACCRUAL are given below.

```
/* ----------------------------------------------------------------
   NAME : HU_OTHER_ADD_HOLIDAY_MULTIPLIER

   This formula calculates the total accrued other additional holiday
   for a specific period.
   ----------------------------------------------------------------*/
DEFAULT FOR ACP_INELIGIBILITY_PERIOD_TYPE IS 'CM'
DEFAULT FOR ACP_INELIGIBILITY_PERIOD_LENGTH IS 0
DEFAULT FOR ACP_CONTINUOUS_SERVICE_DATE IS '4712/12/31 00:00:00'(date)
DEFAULT FOR ACP_ENROLLMENT_END_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_TERMINATION_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_ENROLLMENT_START_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_SERVICE_START_DATE IS '4712/12/31 00:00:00' (date)

INPUTS ARE Calculation_Date (date)
Accruing_Frequency = ' '
Accruing_Multiplier = 0
E = SET_DATE('CALCULATION_DATE',Calculation_Date)

/* ----------------------------------------------------------------
   Set the payroll period, accruing frequency, and accruing multiplier
   based on the payroll.
   ----------------------------------------------------------------*/

Payroll_period = HU_PAYROLL_PERIODS(Calculation_Date
                                   ,Accruing_Frequency
                        ,Accruing_Multiplier)


E = SET_TEXT('ACCRUING_FREQUENCY', Accruing_Frequency)
E = SET_NUMBER('ACCRUING_MULTIPLIER', Accruing_Multiplier)

/* ----------------------------------------------------------------
  Calculate the start and end dates of the current leave year
  ----------------------------------------------------------------*/

Beginning_Of_Calculation_Year =
                     TO_DATE('0101'||TO_CHAR(Calculation_Date,'YYYY')
                            ,'DDMMYYYY')

IF Beginning_Of_Calculation_Year > Calculation_Date THEN
(
  Beginning_of_Calculation_Year =
                      ADD_MONTHS(Beginning_Of_Calculation_Year, -12)
)

E = SET_DATE('BEGINNING_OF_CALCULATION_YEAR'
            , Beginning_Of_Calculation_Year)

E = GET_PERIOD_DATES(Beginning_of_Calculation_Year
                    ,Accruing_Frequency
                    ,Beginning_Of_Calculation_Year
                    ,Accruing_Multiplier)

First_Period_SD = GET_DATE('PERIOD_START_DATE')
First_Period_ED = GET_DATE('PERIOD_END_DATE')

/* ----------------------------------------------------------------
   Set the Calculation_Date to the Termination Date if not null
   ----------------------------------------------------------------*/

IF NOT (ACP_TERMINATION_DATE WAS DEFAULTED) OR
    NOT (ACP_ENROLLMENT_END_DATE WAS DEFAULTED) THEN
(
    Early_End_Date = LEAST(ACP_TERMINATION_DATE
                          ,ACP_ENROLLMENT_END_DATE)
```

```
    IF (Early_End_Date < Calculation_Date) THEN
    (
        Calculation_Date = Early_End_Date
    )
)

/* ---------------------------------------------------------------------
   Get the last whole period prior to the Calculation Date and ensure
   that it   is within the Year (if the Calculation Date is the End
   of a Period then use that period)
   ---------------------------------------------------------------------*/

E = GET_PERIOD_DATES(Calculation_Date
                    ,Accruing_Frequency
                    ,Beginning_of_Calculation_Year
                    ,Accruing_Multiplier)

Calculation_Period_SD  = GET_DATE('PERIOD_START_DATE')
Calculation_Period_ED  = GET_DATE('PERIOD_END_DATE')

/* ---------------------------------------------------------------------
   Set the Continuous Service Global Variable, whilst also
   ensuring that the continuous service date is before the
   Calculation Period
  ---------------------------------------------------------------------*/

IF (ACP_CONTINUOUS_SERVICE_DATE WAS DEFAULTED) THEN
(
    E = SET_DATE('CONTINUOUS_SERVICE_DATE', ACP_SERVICE_START_DATE)
)
ELSE IF(ACP_CONTINUOUS_SERVICE_DATE > Calculation_Period_SD) THEN
(
    E = SET_DATE('CONTINUOUS_SERVICE_DATE'
                ,ACP_CONTINUOUS_SERVICE_DATE)
)
ELSE
(
    E = SET_DATE('CONTINUOUS_SERVICE_DATE'
                ,ACP_CONTINUOUS_SERVICE_DATE)
)

Continuous_Service_Date = GET_DATE('CONTINUOUS_SERVICE_DATE')

First_Eligible_To_Accrue_Date  = Continuous_Service_Date

/*---------------------------------------------------------------------
   Determine the date on which accrued PTO may first be registered,
   i.e the date on which the Ineligibility Period expires
  ---------------------------------------------------------------------*/

Accrual_Ineligibility_Expired_Date = First_Eligible_To_Accrue_Date

IF (ACP_INELIGIBILITY_PERIOD_LENGTH > 0) THEN
(
    IF ACP_INELIGIBILITY_PERIOD_TYPE = 'BM' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_MONTHS(Continuous_Service_Date
                               ,ACP_INELIGIBILITY_PERIOD_LENGTH*2)
    )
    ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'F' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_DAYS(Continuous_Service_Date
                            ,ACP_INELIGIBILITY_PERIOD_LENGTH*14)
    )
```

```
ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'CM' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_MONTHS(Continuous_Service_Date
                               ,ACP_INELIGIBILITY_PERIOD_LENGTH)
    )
    ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'LM' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_DAYS(Continuous_Service_Date
                            ,ACP_INELIGIBILITY_PERIOD_LENGTH*28)
    )
    ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'Q' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_MONTHS(Continuous_Service_Date
                               ,ACP_INELIGIBILITY_PERIOD_LENGTH*3)
    )
    ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'SM' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_MONTHS(Continuous_Service_Date
                               ,ACP_INELIGIBILITY_PERIOD_LENGTH/2)
    )
    ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'SY' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_MONTHS(Continuous_Service_Date
                               ,ACP_INELIGIBILITY_PERIOD_LENGTH*6)
    )
    ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'W' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_DAYS(Continuous_Service_Date
                            ,ACP_INELIGIBILITY_PERIOD_LENGTH*7)
    )
    ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'Y' THEN
    (
        Accrual_Ineligibility_Expired_Date =
                    ADD_MONTHS(Continuous_Service_Date
                               ,ACP_INELIGIBILITY_PERIOD_LENGTH*12)
    )

    IF Accrual_Ineligibility_Expired_Date >
                                        First_Eligible_To_Accrue_Date
     AND Calculation_Date < Accrual_Ineligibility_Expired_Date THEN
     (
        First_Eligible_To_Accrue_Date =
                                    Accrual_Ineligibility_Expired_Date
     )
)

/* ------------------------------------------------------------------
   Get the first full period following the
   First_Eligible_To_Accrue_Date(if it falls on the beginning of the
   period then use that period)
 ------------------------------------------------------------------*/

IF First_Eligible_To_Accrue_Date > Beginning_Of_Calculation_Year THEN
(
  E = GET_PERIOD_DATES(First_Eligible_To_Accrue_Date
                      ,Accruing_Frequency
                      ,Beginning_Of_Calculation_Year
                      ,Accruing_Multiplier)

  First_Eligible_To_Accrue_Period_SD  = GET_DATE('PERIOD_START_DATE')
```

```
First_Eligible_To_Accrue_Period_ED  = GET_DATE('PERIOD_END_DATE')


  IF (First_Eligible_To_Accrue_Period_SD > Calculation_Period_ED) THEN
  (
    Total_Accrued_PTO = 0
    E = PUT_MESSAGE('HR_52793_PTO_FML_ASG_INELIG')
  )
)
ELSE
(
  First_Eligible_To_Accrue_Period_SD  = First_Period_SD
  First_Eligible_To_Accrue_Period_ED  = First_Period_ED
)
/* ---------------------------------------------------------------------
   Determine the date on which PTO actually starts accruing based on
   Hire Date, Continuous Service Date and plan Enrollment Start Date.
   ---------------------------------------------------------------------*/

IF Continuous_Service_date = ACP_CONTINUOUS_SERVICE_DATE THEN
(
  Actual_Accrual_Start_Date = Continuous_service_Date
)
ELSE
(
  Actual_Accrual_Start_Date = GREATEST(Continuous_Service_Date
                                      ,ACP_ENROLLMENT_START_DATE
                                      ,First_Period_SD)
)

/* ---------------------------------------------------------------------
   Determine the actual start of the accrual calculation
   ---------------------------------------------------------------------*/
IF (Actual_Accrual_Start_Date > First_Period_SD AND
  Actual_Accrual_Start_Date > First_Eligible_To_Accrue_Period_SD) THEN
(
    E = GET_PERIOD_DATES(Actual_Accrual_Start_Date
                        ,Accruing_Frequency
                        ,Beginning_Of_Calculation_Year
                        ,Accruing_Multiplier)

     Accrual_Start_Period_SD = GET_DATE('PERIOD_START_DATE')
     Accrual_Start_Period_ED = GET_DATE('PERIOD_END_DATE')


/* ---------------------------------------------------------------------
   If the Actual Accrual Period is after the Calculation Period then
   end the processing.
   ---------------------------------------------------------------------*/
    IF (Accrual_Start_Period_SD > Calculation_Period_ED) THEN
    (
        Total_Accrued_PTO = 0
        E = PUT_MESSAGE('HR_52797_PTO_FML_ACT_ACCRUAL')
     )
)

ELSE IF (First_Eligible_To_Accrue_Period_SD > First_Period_SD) THEN
(
    Accrual_Start_Period_SD = First_Eligible_To_Accrue_Period_SD
    Accrual_Start_Period_ED = First_Eligible_To_Accrue_Period_ED
)
ELSE
(
    Accrual_Start_Period_SD = First_Period_SD
    Accrual_Start_Period_ED = First_Period_ED
)
```

```
/* ------------------------------------------------------------------

   Now set up the information that will be used in when looping
   through the periods.
   ------------------------------------------------------------------*/
IF Calculation_Period_ED >= Accrual_Start_Period_ED THEN
(
    E = SET_DATE('PERIOD_SD',Accrual_Start_Period_SD)
    E = SET_DATE('PERIOD_ED',Accrual_Start_Period_ED)
    E = SET_DATE('LAST_PERIOD_SD',Calculation_Period_SD)
    E = SET_DATE('LAST_PERIOD_ED',Calculation_Period_ED)
    E = SET_NUMBER('TOTAL_ACCRUED_PTO',0)

    E = LOOP_CONTROL('HU_OTHER_ADD_HOLIDAY_PERIOD_ACCRUAL')

    Total_Accrued_PTO = ROUND(GET_NUMBER('TOTAL_ACCRUED_PTO'))
)

IF Accrual_Start_Period_SD <= Calculation_Period_SD THEN
(
    Accrual_end_date = Calculation_Period_ED
)

Effective_Start_Date = Accrual_Start_Period_SD
Effective_End_Date   = Calculation_Period_ED

IF Effective_Start_Date >= Effective_End_Date THEN
(
    Effective_Start_Date = Effective_End_Date
)

RETURN Total_Accrued_PTO
      ,Effective_start_date
      ,Effective_end_date
      ,Accrual_end_date
```

### Looping Formula

```
/* ------------------------------------------------------------------
  NAME : HU_OTHER_ADD_HOLIDAY_PERIOD_ACCRUAL

  This formula calculates the amount of PTO accrued for a particular
  period.
  ------------------------------------------------------------------*/

DEFAULT FOR PER_DATE_OF_BIRTH IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_TERMINATION_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR EMP_HIRE_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR Person_dob IS '4712/12/31 00:00:00' (date)

Continuous_Service_Date = GET_DATE('CONTINUOUS_SERVICE_DATE')
Total_Accrued_PTO = GET_NUMBER('TOTAL_ACCRUED_PTO')
Period_SD = GET_DATE('PERIOD_SD')
Period_ED = GET_DATE('PERIOD_ED')
Last_Period_SD = GET_DATE('LAST_PERIOD_SD')
Last_Period_ED = GET_DATE('LAST_PERIOD_ED')
Payroll_Year_SD = GET_DATE('PAYROLL_YEAR_SD')
Accruing_Frequency = GET_TEXT('ACCRUING_FREQUENCY')
Accruing_Multiplier = GET_NUMBER('ACCRUING_MULTIPLIER')
beginining_year = GET_DATE('BEGINNING_OF_CALCULATION_YEAR')
Calculation_Date=GET_DATE('CALCULATION_DATE')

/* ------------------------------------------------------------------
   Get the person date of birth and compute the age.
   ------------------------------------------------------------------*/

Person_dob = HU_PERSON_DOB(Calculation_Date)
Age = FLOOR(MONTHS_BETWEEN(Period_ED, Person_dob)/12)

/* ------------------------------------------------------------------
   Set the payroll period, accruing frequency, and accruing multiplier
   based on the payroll.
   ------------------------------------------------------------------*/

P= HU_PAYROLL_PERIODS(Calculation_Date
                     ,Accruing_Frequency
                     ,Accruing_Multiplier)
Accrual_Rate =0
st_date=beginining_year
ed_date=TO_DATE('3112'||TO_CHAR(beginining_year,'YYYY'),'DDMMYYYY')

IF EMP_HIRE_DATE > Period_SD AND EMP_HIRE_DATE < Period_ED THEN
(
 Period_SD= EMP_HIRE_DATE
)
IF ACP_TERMINATION_DATE > Period_SD AND
   ACP_TERMINATION_DATE < Period_ED THEN
(
 Period_ED=ACP_TERMINATION_DATE
)

DOB=TO_DATE(TO_CHAR(Person_dob,'DD/MM/')||TO_CHAR(PERIOD_SD,'YYYY')
           ,'DD/MM/YYYY')

/* ------------------------------------------------------------------
  Set accrual rate based on age and working pattern.
  ------------------------------------------------------------------*/

IF Age = 18 and DOB>=PERIOD_SD and DOB<=PERIOD_ED THEN
(
 x1 = HU_ABS_GET_WORKING_DAYS(PERIOD_SD,ADD_DAYS(DOB,-1))
 Accrual_Rate  = (5/260*x1)
)
ELSE IF Age<18 THEN
```

```
(
    x1 = HU_ABS_GET_WORKING_DAYS(PERIOD_SD,PERIOD_ED)
    Accrual_Rate =5/260*x1
)
ELSE
(
 Accrual_Rate =0
)
Period_Accrued_PTO=Accrual_Rate
/* ---------------------------------------------------------------------
   Checking for Blind Days
   ---------------------------------------------------------------------*/
Accrual_Rate =0
Accrual_Rate =HU_ABS_GET_BLIND_DAYS(Period_SD,Period_ED)*5/260

Period_Accrued_PTO=Period_Accrued_PTO+Accrual_Rate

/* ---------------------------------------------------------------------
  Checking the additional health of employees.
  ---------------------------------------------------------------------*/

Accrual_Rate = 0
Accrual_Rate = HU_ABS_GET_JOB_DAYS(Period_SD,Period_ED)*5/260
Period_Accrued_PTO = Period_Accrued_PTO + Accrual_Rate

/*---------------------------------------------------------------------
    Set the Running Total for Total_Accrued_PTO
  ---------------------------------------------------------------------*/
E = SET_NUMBER('TOTAL_ACCRUED_PTO',Total_Accrued_PTO
              + Period_Accrued_PTO)


/* ---------------------------------------------------------------------
    Establish whether the current period is the last one, if so end
    the processing, otherwise get the     next period
---------------------------------------------------------------------*/
IF Period_SD >= Last_Period_SD THEN
(
    Continue_Processing_Flag = 'N'
)
ELSE
(
    E = GET_PAYROLL_PERIOD(ADD_DAYS(Period_ED,1))
    E = SET_DATE('PERIOD_SD',GET_DATE('PAYROLL_PERIOD_START_DATE'))
    E = SET_DATE('PERIOD_ED',GET_DATE('PAYROLL_PERIOD_END_DATE'))
    Continue_Processing_Flag = 'Y'
)

RETURN Continue_Processing_Flag
```

## Sample Formula for Sickness Holiday (Hungary)

Oracle HRMS provides the HU_SICKNESS_HOLIDAY_MULTIPLIER to calculate the Sickness Holiday as required for the Hungarian Absence Report. The sickness holiday is based on the following rules:

- Annual sickness holiday of fifteen days.
- Entitlement depends on the number of days worked in a week. The formula calculates proportionately if the work pattern changes within the leave year.

- Entitlement depends on the sickness holiday the employee has taken in the previous employment.

The HU_SICKNESS_HOLIDAY_MULTIPLIER calls another formula, HU_SICKNESS_HOLIDAY_PERIOD_ACCRUAL to calculate the accrual within a specific period. There is no carry over formula for Sickness Holiday.

The top-level formula, HU_SICKNESS_HOLIDAY_MULTIPLIER, and the looping formula HU_SICKNESS_HOLIDAY_PERIOD_ACCRUAL are given below.

```
/* ----------------------------------------------------------------------
NAME : HU_SICKNESS_HOLIDAY_MULTIPLIER
This formula calculates the total accrued sickness holiday for a
specific period.
----------------------------------------------------------------------*/

DEFAULT FOR ACP_INELIGIBILITY_PERIOD_TYPE IS 'CM'
DEFAULT FOR ACP_INELIGIBILITY_PERIOD_LENGTH IS 0
DEFAULT FOR ACP_CONTINUOUS_SERVICE_DATE IS '4712/12/31 00:00:00'(DATE)
DEFAULT FOR ACP_ENROLLMENT_END_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR ACP_TERMINATION_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR ACP_ENROLLMENT_START_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR ACP_SERVICE_START_DATE IS '4712/12/31 00:00:00' (DATE)

INPUTS ARE Calculation_Date (DATE)

Accruing_Frequency = ' '
Accruing_Multiplier = 0

/* ----------------------------------------------------------------------
Set the payroll period, accruing frequency, and accruing multiplier
based on the payroll.
----------------------------------------------------------------------*/
No_of_Payroll_Periods = HU_PAYROLL_PERIODS (Calculation_Date
                                           ,Accruing_Frequency
                                           ,Accruing_Multiplier)

E = SET_TEXT('ACCRUING_FREQUENCY', Accruing_Frequency)
E = SET_NUMBER('ACCRUING_MULTIPLIER', Accruing_Multiplier)

/* ----------------------------------------------------------------------
Calculate the start and end Dates of the current leave year
----------------------------------------------------------------------*/
Beginning_Of_Calculation_Year = TO_DATE('0101'
                                        ||TO_CHAR(Calculation_Date
                                                  ,'YYYY')
                                        ,'DDMMYYYY')
End_Of_Calculation_Year = to_DATE('3112'||TO_CHAR(Calculation_Date
                                                  ,'YYYY')
                                  ,'DDMMYYYY')

IF Beginning_Of_Calculation_Year > Calculation_Date THEN
(
  Beginning_of_Calculation_Year = ADD_MONTHS
(Beginning_Of_Calculation_Year
                                           , -12)
)

E = SET_DATE('BEGINNING_OF_CALCULATION_YEAR'
            , Beginning_Of_Calculation_Year)

E = GET_PERIOD_DATES(Beginning_of_Calculation_Year
                    ,Accruing_Frequency
                    ,Beginning_Of_Calculation_Year
                    ,Accruing_Multiplier)

First_Period_SD = GET_DATE('PERIOD_START_DATE')
First_Period_ED = GET_DATE('PERIOD_END_DATE')

E = GET_PERIOD_DATES(End_Of_Calculation_Year
                    ,Accruing_Frequency
                    ,Beginning_Of_Calculation_Year
                    ,Accruing_Multiplier)

Last_Period_SD = GET_DATE('PERIOD_START_DATE')
Last_Period_ED = GET_DATE('PERIOD_END_DATE')
```

```
/* ------------------------------------------------------------------
   Set the Calculation_Date to the Termination Date if not null
------------------------------------------------------------------*/

IF NOT (ACP_TERMINATION_DATE WAS DEFAULTED) OR
NOT (ACP_ENROLLMENT_END_DATE WAS DEFAULTED) THEN
(
  Early_End_Date = LEAST(ACP_TERMINATION_DATE
                        ,ACP_ENROLLMENT_END_DATE)
  IF (Early_End_Date < First_Period_SD) THEN
  (
    Total_Accrued_PTO = 0
    E = PUT_MESSAGE('HR_52794_PTO_FML_ASG_TER')
  )
  IF (Early_End_Date < Last_Period_ED) THEN
  (
    E = GET_PERIOD_DATES(Early_End_Date
                      ,Accruing_Frequency
                      ,Beginning_Of_Calculation_Year
                      ,Accruing_Multiplier)
    Last_Period_SD = GET_DATE('PERIOD_START_DATE')
    Last_Period_ED = GET_DATE('PERIOD_END_DATE')
  )
  IF (Early_End_Date < Calculation_Date) THEN
  (
    Calculation_Date = Early_End_Date
  )
)

/* ------------------------------------------------------------------
Get the last whole period prior to the Calculation Date and ensure
that it is within the Year (if the Calculation Date is the End of a
Period then use that period)
------------------------------------------------------------------ */

E = GET_PERIOD_DATES(Calculation_Date
                    ,Accruing_Frequency
                    ,Beginning_of_Calculation_Year
                    ,Accruing_Multiplier)

Calculation_Period_SD  = GET_DATE('PERIOD_START_DATE')
Calculation_Period_ED  = GET_DATE('PERIOD_END_DATE')

IF (Calculation_Period_ED < First_Period_SD) THEN
(
  Total_Accrued_PTO = 0
  E = PUT_MESSAGE('HR_52795_PTO_FML_CALC_DATE')
)
/* ------------------------------------------------------------------
Set the Continuous Service Global Variable, whilst also
ensuring that the continuous service date is before the
Calculation Period
------------------------------------------------------------------ */

IF (ACP_CONTINUOUS_SERVICE_DATE WAS DEFAULTED) THEN
(
  E = SET_DATE('CONTINUOUS_SERVICE_DATE', ACP_SERVICE_START_DATE)
)
ELSE IF(ACP_CONTINUOUS_SERVICE_DATE > Calculation_Period_ED) THEN
(
  Total_Accrued_PTO = 0
  E = PUT_MESSAGE('HR_52796_PTO_FML_CSD')
  E = SET_DATE('CONTINUOUS_SERVICE_DATE'
              , ACP_CONTINUOUS_SERVICE_DATE)
)
ELSE IF(ACP_CONTINUOUS_SERVICE_DATE > First_Period_SD) THEN
```

```
(
  E = GET_PERIOD_DATES(ACP_CONTINUOUS_SERVICE_DATE,
                       Accruing_Frequency,
                       Beginning_Of_Calculation_Year,
                       Accruing_Multiplier)
  First_Period_SD = GET_DATE('PERIOD_START_DATE')
  First_Period_ED = GET_DATE('PERIOD_END_DATE')
)
ELSE
(
  E = SET_DATE('CONTINUOUS_SERVICE_DATE'
              , ACP_CONTINUOUS_SERVICE_DATE)
)

Continuous_Service_Date = GET_DATE('CONTINUOUS_SERVICE_DATE')
First_Eligible_To_Accrue_Date  = Continuous_Service_Date

/*---------------------------------------------------------------------
Determine the date on which accrued PTo may first be registered,
i.e the date on which the Ineligibility Period expires
 ---------------------------------------------------------------- */
Accrual_Ineligibility_Expired_Date = First_Eligible_To_Accrue_Date

IF (ACP_INELIGIBILITY_PERIOD_LENGTH > 0) THEN
(
  IF ACP_INELIGIBILITY_PERIOD_TYPE = 'BM' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                     ADD_MONTHS(Continuous_Service_Date
                               ,ACP_INELIGIBILITY_PERIOD_LENGTH*2)
  )
  ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'F' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                    ADD_DAYS(Continuous_Service_Date
                            ,ACP_INELIGIBILITY_PERIOD_LENGTH*14)
  )
  ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'CM' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                  ADD_MONTHS(Continuous_Service_Date
                            ,ACP_INELIGIBILITY_PERIOD_LENGTH)
  )
  ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'LM' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                 ADD_DAYS(Continuous_Service_Date
                         ,ACP_INELIGIBILITY_PERIOD_LENGTH*28)
  )
  ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'Q' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                  ADD_MONTHS(Continuous_Service_Date
                            ,ACP_INELIGIBILITY_PERIOD_LENGTH*3)
  )
  ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'SM' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                 ADD_MONTHS(Continuous_Service_Date
                           ,ACP_INELIGIBILITY_PERIOD_LENGTH/2)
  )
  ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'SY' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                ADD_MONTHS(Continuous_Service_Date
                          ,ACP_INELIGIBILITY_PERIOD_LENGTH*6)
```

```
)
  ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'W' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                  ADD_DAYS(Continuous_Service_Date
                          ,ACP_INELIGIBILITY_PERIOD_LENGTH*7)
  )
  ELSE IF ACP_INELIGIBILITY_PERIOD_TYPE = 'Y' THEN
  (
    Accrual_Ineligibility_Expired_Date =
                 ADD_MONTHS(Continuous_Service_Date
                           ,ACP_INELIGIBILITY_PERIOD_LENGTH*12)
  )
  IF Accrual_Ineligibility_Expired_Date > First_Eligible_To_Accrue_Date
  AND Calculation_Date < Accrual_Ineligibility_Expired_Date THEN
  (
    First_Eligible_To_Accrue_Date = Accrual_Ineligibility_Expired_Date
  )
)

/* ---------------------------------------------------------------------
Get the first full period following the
First_Eligible_To_Accrue_Date (if it falls on the beginning of the
period then use that period)
--------------------------------------------------------------------- */

IF First_Eligible_To_Accrue_Date > Beginning_Of_Calculation_Year THEN
(
  E = GET_PERIOD_DATES(First_Eligible_To_Accrue_Date
                      ,Accruing_Frequency
                      ,Beginning_Of_Calculation_Year
                      ,Accruing_Multiplier)
  First_Eligible_To_Accrue_Period_SD  = GET_DATE('PERIOD_START_DATE')
  First_Eligible_To_Accrue_Period_ED  = GET_DATE('PERIOD_END_DATE')

  IF (First_Eligible_To_Accrue_Period_SD > Calculation_period_ED) THEN
  (
    Total_Accrued_PTO = 0
    E = PUT_MESSAGE('HR_52793_PTO_FML_ASG_INELIG')
  )
)
ELSE
(
  First_Eligible_To_Accrue_Period_SD  = First_Period_SD
  First_Eligible_To_Accrue_Period_ED  = First_Period_ED
)

/* ---------------------------------------------------------------------
Determine the date on which PTO actually starts accruing based
on Hire Date,Continuous Service Date and plan Enrollment Start
Date.
--------------------------------------------------------------------- */

IF Continuous_Service_date = ACP_CONTINUOUS_SERVICE_DATE THEN
(
  Actual_Accrual_Start_Date = Continuous_service_Date
)
ELSE
(
  Actual_Accrual_Start_Date = GREATEST(Continuous_Service_Date
                                      ,ACP_ENROLLMENT_START_DATE
                                      ,First_Period_SD)
)

/* ---------------------------------------------------------------------
    Determine the actual start of the accrual calculation
```

```
-------------------------------------------------------------------*/

IF (Actual_Accrual_Start_Date > First_Period_SD AND
    Actual_Accrual_Start_Date > First_Eligible_To_Accrue_Period_SD) THEN
(
  E = GET_PERIOD_DATES(Actual_Accrual_Start_Date
                      ,Accruing_Frequency
                      ,Beginning_Of_Calculation_Year
                      ,Accruing_Multiplier)
  Accrual_Start_Period_SD = GET_DATE('PERIOD_START_DATE')
  Accrual_Start_Period_ED = GET_DATE('PERIOD_END_DATE')
  /*-----------------------------------------------------------------
  If the Actual Accrual Period is after the Calculation Period then
  end the processing.
  ----------------------------------------------------------------- */

  IF (Accrual_Start_Period_SD > Calculation_period_ED) THEN
  (
    Total_Accrued_PTO = 0
    E = PUT_MESSAGE('HR_52797_PTO_FML_ACT_ACCRUAL')
  )
)
ELSE IF (First_Eligible_To_Accrue_Period_SD > First_Period_SD) THEN
(
  Accrual_Start_Period_SD = First_Eligible_To_Accrue_Period_SD
  Accrual_Start_Period_ED = First_Eligible_To_Accrue_Period_ED
)
ELSE
(
  Accrual_Start_Period_SD = First_Period_SD
  Accrual_Start_Period_ED = First_Period_ED
)

/*-----------------------------------------------------------------
Retrieve sickness information for previous employment
----------------------------------------------------------------- */

Previous_employment = 'N'
Prev_Sickness_Leave = 0

IF TO_CHAR(ACP_SERVICE_START_DATE,'yyyy') = TO_CHAR
   (Calculation_Date,'yyyy') THEN
(
  Prev_Sickness_Leave = HU_ABS_GET_PREV_EMP_SICKNESS_LEAVE
                            (TO_CHAR(Calculation_Date,'yyyy')
                            ,Previous_employment)
)
Accrued_PTO = 0

/* ----------------------------------------------------------------
Now set up the information that will be used in when looping
through the periods
---------------------------------------------------------------- */

IF Last_period_ED >= Accrual_Start_Period_ED THEN
(
  E = SET_DATE('PERIOD_SD',Accrual_Start_Period_SD)
  E = SET_DATE('PERIOD_ED',Accrual_Start_Period_ED)
  E = SET_DATE('LAST_PERIOD_SD',Calculation_period_SD)
  E = SET_DATE('LAST_PERIOD_ED',Calculation_period_ED)
  E = SET_NUMBER('TOTAL_ACCRUED_PTO',Accrued_PTO)
  E = LOOP_CONTROL('HU_SICKNESS_HOLIDAY_PERIOD_ACCRUAL')
  Total_Accrued_PTO = ROUND(GET_NUMBER('TOTAL_ACCRUED_PTO'))
  IF Previous_employment = 'Y' AND  Total_Accrued_PTO >
                          15 - Prev_Sickness_Leave THEN
  (
```

```
Total_Accrued_PTO = 15 - Prev_Sickness_Leave
    E = SET_NUMBER('TOTAL_ACCRUED_PTO',Total_Accrued_PTO)
  )
)

IF Accrual_Start_Period_SD <= Calculation_period_ED THEN
(
  Accrual_end_date = Calculation_period_ED
)

Effective_Start_Date = Accrual_Start_Period_SD
Effective_End_Date = Calculation_period_ED
IF Effective_Start_Date >= Effective_End_Date THEN
(
  Effective_Start_Date = Effective_End_Date
)

RETURN Total_Accrued_PTO, Effective_start_date
       , Effective_end_date, Accrual_end_date



Looping formula

/* ---------------------------------------------------------------------
NAME : HU_SICKNESS_HOLIDAY_PERIOD_ACCRUAL
This formula calculates the amount of PTO accrued for a particular
period
---------------------------------------------------------------------*/

/*---------------------------------------------------------------------
  Get the global variable to be used in this formula
---------------------------------------------------------------------*/

DEFAULT FOR ACP_TERMINATION_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR EMP_HIRE_DATE IS '4712/12/31 00:00:00' (DATE)
DEFAULT FOR ACP_SERVICE_START_DATE IS '4712/12/31 00:00:00' (DATE)
Continuous_Service_Date = GET_DATE('CONTINUOUS_SERVICE_DATE')
Total_Accrued_PTO = GET_NUMBER('TOTAL_ACCRUED_PTO')
Period_SD = GET_DATE('PERIOD_SD')
Period_ED = GET_DATE('PERIOD_ED')
Last_Period_SD = GET_DATE('LAST_PERIOD_SD')
Last_Period_ED = GET_DATE('LAST_PERIOD_ED')

/* ---------------------------------------------------------------------
Set the payroll period, accruing frequency, and accruing multiplier
based on the payroll.
---------------------------------------------------------------------*/

Accruing_Frequency = GET_TEXT('ACCRUING_FREQUENCY')
Accruing_Multiplier = GET_NUMBER('ACCRUING_MULTIPLIER')
Beginning_of_Calculation_Year =
                          GET_DATE('BEGINNING_OF_CALCULATION_YEAR')

Period_Accrued_PTO = 0
IF ACP_SERVICE_START_DATE >= Period_SD
  AND ACP_SERVICE_START_DATE <= Period_ED
  AND ACP_TERMINATION_DATE  >= Period_SD
  AND ACP_TERMINATION_DATE <= Period_ED THEN
(

  Days_valid = HU_ABS_GET_WORKING_DAYS(ACP_SERVICE_START_DATE
                                      ,ACP_TERMINATION_DATE)
)
ELSE IF ACP_SERVICE_START_DATE >= Period_SD AND
        ACP_SERVICE_START_DATE <= Period_ED THEN
```

```
(
  Days_valid = HU_ABS_GET_WORKING_DAYS(ACP_SERVICE_START_DATE
                                      , Period_ED)
)
ELSE IF ACP_TERMINATION_DATE  >= Period_SD AND
        ACP_TERMINATION_DATE <= Period_ED THEN
(
 Days_valid = HU_ABS_GET_WORKING_DAYS(Period_SD
                                     , ACP_TERMINATION_DATE)
)
ELSE
(
 Days_valid = HU_ABS_GET_WORKING_DAYS(Period_SD,Period_ED)
)

/* ---------------------------------------------------------------
 Calculate the Amount Accrued this Period
---------------------------------------------------------------*/
Period_Accrued_PTO = 15 * (Days_valid/260)

/*---------------------------------------------------------------
Calculate any absence or bought/sold time etc. to be accounted for in
this period.
---------------------------------------------------------------*/
Absence = GET_ABSENCE(Period_ED, Beginning_of_Calculation_Year)
CarryOver = GET_CARRY_OVER(Period_ED, Beginning_of_Calculation_Year)
Other = GET_OTHER_NET_CONTRIBUTION(Period_ED
                                  ,Beginning_of_Calculation_Year)
Period_Others = CarryOver + Other - Absence

/*---------------------------------------------------------------
    Set the Running Total
---------------------------------------------------------------*/

E = SET_NUMBER('TOTAL_ACCRUED_PTO'
              ,Total_Accrued_PTO + Period_Accrued_PTO)

/* ---------------------------------------------------------------
Establish whether the current period is the last one, if so end
the processing, otherwise get the next period
---------------------------------------------------------------*/

IF Period_SD = Last_Period_SD THEN
(
  Continue_Processing_Flag = 'N'
)
ELSE
(
  E = GET_PERIOD_DATES(ADD_DAYS(Period_ED,1)
                      ,Accruing_Frequency
                      ,Beginning_of_Calculation_Year
                      ,Accruing_Multiplier)
  E = SET_DATE('PERIOD_SD', GET_DATE('PERIOD_START_DATE'))
  E = SET_DATE('PERIOD_ED', GET_DATE('PERIOD_END_DATE'))
  Continue_Processing_Flag = 'Y'
)

RETURN Continue_Processing_Flag
```

## Sample Formula for Carry Over Absence (Hungary)

Oracle HRMS provides the HU_ABS_CARRYOVER_FORMULA to calculate their unused Base Holiday, Additional Holiday for bringing up children, and Other

Additional Holiday except Sickness Holiday. Employees can carry over these holidays up to 30th June of the following year or, if the collective agreement permits, up to 31st December of the following year. The formula considers 30th June as the carry over expiry date.

You can use the sample formula HU_ABS_CARRYOVER_FORMULA to calculate the above entitlements. The sample formula is given below:

```
/*=====================================================================
Formula Title : HU_ABS_HOLIDAY_CARRY_OVER
Description   : This Formula carries over the holidays remaining to
                next period.
=====================================================================*/
DEFAULT FOR ACP_CONTINUOUS_SERVICE_DATE IS '4712/12/31 00:00:00' (date)
DEFAULT FOR ACP_SERVICE_START_DATE IS '4712/12/31 00:00:00' (date)

INPUTS ARE  Calculation_Date (date)
           ,Accrual_Term     (text)

IF  ACP_CONTINUOUS_SERVICE_DATE WAS DEFAULTED THEN
(
  Continuous_Service_Date = ACP_SERVICE_START_DATE
)
ELSE
(
  Continuous_Service_Date = ACP_CONTINUOUS_SERVICE_DATE
)
calculation_period_end_date  =
         TO_DATE('3112'|| TO_CHAR(Calculation_Date,'YYYY'),'DDMMYYYY')

IF  Accrual_Term = 'PREVIOUS' THEN
(
  Effective_Date = ADD_YEARS(calculation_period_end_date, -1)
)
ELSE
(
Effective_Date = calculation_period_end_date
)

Expiry_Date = ADD_MONTHS(effective_date, 6)
Years_service = FLOOR(MONTHS_BETWEEN(ADD_DAYS(Effective_date,1)
                                     , Continuous_Service_Date) / 12)

IF  (GET_ACCRUAL_BAND(Years_service) = 0) THEN
(
  Max_carryover = GET_NUMBER('MAX_CARRY_OVER')
)
ELSE
  Max_carryover  = 30

Process = 'YES'

RETURN Max_Carryover, Effective_date, Expiry_Date, Process
```

## Sample Formula for Validation of YEA Information (Korea)

Oracle provides a predefined legislative formula KR_VALIDATE_YEA_DATA that you can use as a template to write a custom formula to validate YEA information for each business group. The formula lists employee information that you can use as input for the custom formula. If you require more information, you must write formula functions

to query the saved data from the PER_KR_ASSIGNMENT_YEA_INFO table.

If you do not want to define the formula VALIDATE_YEA_DATA, then the application skips the custom validation step.

When you write the custom formula, you must observe the following rules:

- Name the formula as VALIDATE_YEA_DATA
- Ensure that the custom formula returns an output parameter named STATUS with value VALID if data passes the custom validation. If the formula returns any other value, then the validation fails
- The custom formula can have a maximum of 10 error messages as output variables to display on the data entry pages, besides STATUS as the output

In the custom formula, you can use database items that use only the following contexts:

- PERSON_ID
- ASSIGNMENT_ID
- DATE_EARNED

**Formula Input**

The YEA Information Entry passes the following parameters to the formula that you can choose or not, as inputs in the formula. If you want to use these parameters, then the INPUT statement within the custom formula must include the desired parameter with the same name as given in the following table:

| PARAMETER NAME | DESCRIPTION |
|---|---|
| **Special Tax Exemption** | |
| EE_EDUC_EXP | Employee Education Expense |
| HOUSING_SAVING_TYPE | Housing Saving Type |
| HOUSING_SAVING | Housing Saving |
| HOUSING_PURCHASE_DATE | Housing Purchase Date |
| HOUSING_LOAN_DATE | Housing Loan Date |
| HOUSING_LOAN_REPAY | Housing Loan Repay Longterm Housing Loan Date due |

| PARAMETER NAME | DESCRIPTION |
|---|---|
| LT_HOUSING_LOAN_DATE | Long term Housing Loan Date due less than 15 Years |
| LT_HOUSING_LOAN_INTEREST_REPAY | Long term Housing Loan Interest Repay due less than 15 Years |
| DONATION1 | Statutory (100%) |
| POLITICAL_DONATION1 | Political Since 2004.3.12 |
| HI_PREM | Health Insurance Premium |
| POLITICAL_DONATION2 | Political Before 2004.3.12 |
| POLITICAL_DONATION3 | Political3 (Obsolete) |
| DONATION2 | Specified (10% Limit) |
| DONATION3 | Special (50% Limit) |
| MED_EXP_EMP | Medical Expense for Employee |
| LT_HOUSING_LOAN_DATE_1 | Long term Housing Loan Date due greater than 15 Yrs |
| LT_HOUSING_LOAN_INT_REPAY_1 | Long term Housing Loan Interest Repay due greater than 15 years |
| MFR_MARRIAGE_OCCASIONS | Marriage Exemption |
| MFR_FUNERAL_OCCASIONS | Funeral Exemption |
| MFR_RELOCATION_OCCASIONS | Relocation Exemption |
| EI_PREM | Employment Insurance Premium |
| ESOA_DONATION | ESOA (30% Limit) |
| PERS_INS_NAME | Personal Insurance Name |

| PARAMETER NAME | DESCRIPTION |
|---|---|
| PERS_INS_PREM | Personal Insurance Premium |
| DISABLED_INS_PREM | Disabled Insurance Premium |
| MED_EXP | Medical Expense |
| MED_EXP_DISABLED | Medical Expense for Disabled |
| MED_EXP_AGED | Medical Expense for Aged |
| EE_OCCUPATION_EDUC_EXP | Employee Occupational Education Expense |
| **Foreign Worker Tax Break** | |
| IMMIGRATION_PURPOSE | Immigration Purpose |
| CONTRACT_DATE | Contract Date |
| STAX_APPLICABLE_FLAG | Special Tax Applicable |
| FWTB_APPLICATION_DATE | Application Date |
| FWTB_SUBMISSION_DATE | Submission Date |
| **Overseas Tax Break** | |
| TAX_PAID_DATE | Tax Paid Date |
| OTB_SUBMISSION_DATE | Submission Date |
| KR_OVS_LOCATION | Location Overseas |
| KR_OVS_WORK_PERIOD | Overseas Work Period |
| KR_OVS_RESPONSIBILITY | Responsibility of Overseas Work |
| TERRITORY_CODE | Country |
| CURRENCY_CODE | Currency |

| PARAMETER NAME | DESCRIPTION |
|---|---|
| TAXABLE | Taxable Earnings |
| TAXABLE_SUBJ_TAX_BREAK | Taxable Earnings subject to Tax Break |
| TAX_BREAK_RATE | Tax Break Rate |
| TAX_FOREIGN_CURRENCY | Tax (Foreign Currency) |
| TAX | Tax (KRW) |
| OTB_APPLICATION_DATE | Application Date |
| **Tax Break** | |
| HOUSING_LOAN_INTEREST_REPAY | Housing Loan Interest Repay |
| STOCK_SAVING | Stock Saving |
| LT_STOCK_SAVING1 | Long term Stock Saving (1st Year) |
| LT_STOCK_SAVING2 | Long term Stock Saving (2nd Year) |
| **Other Tax Exemptions** | |
| DIRECT_CARD_EXP | Direct Payment for Employee |
| DPNT_DIRECT_EXP | Direct Payment for Spouse and Dependent |
| GIRO_TUITION_PAID_EXP | Tuition Paid in GIRO |
| CASH_RECEIPT_EXP | Cash Receipt Expenses |
| KR_MED_EXP_PAID_IN_CARD | Medical Expense Paid in card, direct payment, cash receipt |
| NP_PREM | National Pension Premium |
| PERS_PENSION_PREM | Personal Pension Premium |
| PERS_PENSION_SAVING | Personal Pension Saving |

| PARAMETER NAME | DESCRIPTION |
|---|---|
| INVEST_PARTNERSHIP_FIN1 | Investment Partnership Financing Type1 |
| INVEST_PARTNERSHIP_FIN2 | Investment Partnership Financing Type2 |
| CREDIT_CARD_EXP | Credit Card for Employee |
| EMP_STOCK_OWN_PLAN_CONTRI | Employee Stock Ownership Plan Contribution |
| CREDIT_CARD_EXP_DPNT | Credit Card for Spouse and Dependents |

You can query these values from the table PER_KR_ASSIGNMENT_YEA_INFO. If you need any information other than the parameters specified above, then query the values from the table PER_KR_ASSIGNMENT_YEA_INFO.

**Formula Outputs**

If you pass an output variable STATUS in the RETURN statement of the formula and if value of STATUS is VALID, then the employee's YEA information is valid. Any other output value indicates that employee's YEA information is invalid.

You can return any error messages to display on incorrect data entry on the YEA Information Entry pages. You can pass up to a maximum of 10 error messages. This is an optional feature.

## Sample Formula for Processing Separation Pension (Korea)

Oracle HRMS enables you to calculate the lump sum amount while processing separation pension for employees.

Oracle may change this formula in future releases. This formula is strictly for example or prototype uses only, and is not intended to provide a ready-made solution. You can make a copy of this formula but you should not change your copied version. Always write a formula of your own to meet your own requirements. This formula may contain certain hard-coded values for simplified use.

Oracle HRMS provides the sample formula KR_SEPARATION_PENSION that you can use to calculate the Separation Pension Lump Sum Amount and Separation Pension Lump Sum Non Taxable Amount. You can attach this formula to the Separation Pension Details element as a Standard processing rule. This creates indirect results for separation lump sum amount and non-taxable amount.

If you want to create your own formula, ensure to include the following validation logic. The payroll run will terminate with error message if any of the validations fail:

- Principal and Interest is greater than 0

- Total Received is greater than or equal to 0
- Personal Contribution is greater than Pension Exemption
- Principal and interest is greater than Personal Contribution
- Lump sum amount is calculated only for Defined Contribution

> **Note:** Formula Results are not predefined for this formula. If you decide to use this formula, you must create formula results at business group level as an implementation step.

```
Use this formula for calculating Lump Sum Amount then
follow the steps below:

1. Attach this formula to the seeded element: "Separation
Pension Details"
   as "Standard" Processing Rule using Formula Result
Form.
2. Create Formula Results for the return values as
mentioned below:

   o_lump_sum_amount              -> Indirect Result to
the seeded element
      Separation Pension Lump Sum Amount

   o_non_taxable_amount           -> Indirect Result to
the seeded element
      Separation Pension Lump Sum Non Taxable Amount

   o_incorrect_defined_type       -> Fatal Error Message
   o_incorrect_total_received     -> Fatal Error Message
   o_incorrect_prn_and_interest   -> Fatal Error Message
   o_incorrect_pension_exem       -> Fatal Error Message
   o_incorrect_pers_contribution  -> Fatal Error Message
************************************************************
***************/
default for PERSONAL_CONTRIBUTION        is 0
default for PENSION_EXEMPTION            is 0
default for TOTAL_RECEIVED               is 0
default for PRINCIPAL_AND_INTEREST       is 0
default for
SEPARATION_PENSION_ACCOUNT_DETAILS_DEFINED_TYPE_ENTRY_VALU
E  is 'NA'

INPUTS ARE
    TOTAL_RECEIVED            (number),
    PRINCIPAL_AND_INTEREST    (number),
    PERSONAL_CONTRIBUTION     (number),
    PENSION_EXEMPTION         (number)

o_lump_sum_amount = 0
l_valid = 'Y'

/************ Validation ************/
if TOTAL_RECEIVED < 0 then (
    l_dummy = set_message_name('PAY',
'PAY_KR_SEP_PEN_INV_TOTAL_RCVD')
    o_incorrect_total_received = get_message()
    l_valid = 'N'
)
if PRINCIPAL_AND_INTEREST <= 0 then(
    l_dummy = set_message_name('PAY',
'PAY_KR_SEP_PEN_INV_PRINCPL_INT')
    o_incorrect_prn_and_interest = get_message()
    l_valid = 'N'
)
if PENSION_EXEMPTION > PERSONAL_CONTRIBUTION then(
    l_dummy = set_message_name('PAY',
'PAY_KR_SEP_PEN_INV_PEN_EXEM')
    o_incorrect_pension_exem = get_message()
    l_valid = 'N'
)
if PERSONAL_CONTRIBUTION > PRINCIPAL_AND_INTEREST then (

    l_dummy = set_message_name('PAY',
```

```
'PAY_KR_SEP_PEN_INV_PERS_CONTRI')
    o_incorrect_pers_contribution = get_message()
    l_valid = 'N'
)
if
SEPARATION_PENSION_ACCOUNT_DETAILS_DEFINED_TYPE_ENTRY_VALU
E <> 'DC' then(
    l_dummy = set_message_name('PAY',
'PAY_KR_SEP_PEN_INV_DEF_TYPE')
    o_incorrect_defined_type = get_message()
    l_valid = 'N'
)
/********* end of validation *********/
if l_valid = 'Y' then(
    o_lump_sum_amount = TOTAL_RECEIVED *
        ( 1 - (PERSONAL_CONTRIBUTION - PENSION_EXEMPTION)
/PRINCIPAL_AND_INTEREST )
    o_lump_sum_amount = greatest(0,trunc
(o_lump_sum_amount))
    o_non_taxable_amount = TOTAL_RECEIVED -
o_lump_sum_amount
)
RETURN
    o_lump_sum_amount,
    o_non_taxable_amount,
    o_incorrect_total_received,
    o_incorrect_prn_and_interest,
    o_incorrect_pension_exem,
    o_incorrect_pers_contribution,
    o_incorrect_defined_type
```

## Netherlands Reporting ABP Part Time Percentage

Oracle HRMS enables you to override the ABP Part Time percentage at the assignment level. You define a FastFormula with the name NL_ABP_REPORTING_PART_TIME_PERCENTAGE, returning a value called Part_Time_Percentage. The formula code uses any available database items or balances to obtain a value.

**Note:** This action is only required if you want to override the assignment level information.

## Netherlands Working Hours Formula

Oracle HRMS enables you to derive the weekly working hours using a formula. You define a FastFormula with the name NL_WEEKLY_WORKING_HOURS, to return a parameter called Working_Hours with a value for weekly working hours in the format 99.99. The formula code uses any database items or balances to obtain a value for the weekly working hours.

**Note:** If you do not define a FastFormula, the application draws the

working hours from the Individual Working Hours held on the assignment.

## Netherlands Payee Name Formulas

Oracle HRMS enables you to present the Payee Name of an employee/organization in a specified format, in the EFT Payment File. Creating a FastFormula entitled NL_PAYEE_REPORTING_NAME, you can decide the format the name appears on the file.

Prior to using the FastFormula, the Payee Name of the person/organization appearing on the EFT Payment is taken from the Payee Name field on the Personal Payment Method form. If this field is blank, a check is carried out to find the NL_PAYEE_REPORTING_NAME FastFormula, and use the information generated there. The Payee Name is set to Last Name +"+ Initials of the person being paid, if no formula has been defined.

The Payee Address is taken from the City of the primary address of the relevant employee/organization identified in the Payee Name record. This is left blank if it does not exist. If the Payee Name is derived from the Payee Name field on the Personal Payment Method form, the address used depends on whether the Payee Name specified is a person or an organization. The Payee Address for an organization is taken from the City on the appropriate location address, otherwise, it is taken from the City on the employee's address.

However, if a FastFormula is selected, or the Payee Name consists of the Last Name +"+ Initials, then the City from the employee address is used.

## Netherlands EFT Payment Override Formula

Oracle HRMS enables you to override the contents of the Description field in the Transaction Description Record 160 of the EFT Payment file and enter your own information. You define a FastFormula with the standard name NL_TRANSACTION_DESCRIPTION, returning a value called Transaction_Description. The code in the formula can access any database item to derive the Transaction Description value. This description can be up to 128 characters in length, and spread over a maximum of 4 Transaction Description records.

> **Note:** If you use the override, none of the derived values, such as Employee Name and Assignment Number, is included in the Transaction Description records unless added in the FastFormula.

## Formulas for Netherlands Wage Tax Subsidies

Oracle HRMS provides you with FastFormulas for each of the following Wage Tax Subsidies:

- Wage Tax Eligibility - NL_WAGE_TAX_SUBSIDY_ELIGIBILITY. This formula derives the applicable wage tax subsidies for an employee assignment, and returns values for the relevant subsidies.

- Wage Tax Subsidy for Low Wages - NL_CALC_LOW_WAGES_TAX_SUBSIDY. This formula derives the wage tax subsidy for low wages for an employee.

- Wage Tax Subsidy for Paid Parental Leave - NL_CALC_PAID_PARENTAL_LEAVE_TAX_SUBSIDY. This formula derives a value for the wage tax subsidy for paid parental leave for an employee.

- Wage Tax Subsidy for Education - NL_CALC_EDUCATION_TAX_SUBSIDY. This formula derives a value for the wage tax subsidy for education for an employee.

- Wage Tax Subsidy for the Long Term Unemployed - NL_CALC_LONG_TERM_UNEMPLOYED_TAX_SUBSIDY. This formula derives a value for the wage tax subsidy for long term unemployed for an employee.

## Formulas to Enable Additional Part-Time Percentages for the Netherlands

Oracle HRMS enables you to create FastFormulas to calculate each of the additional part-time percentage values for an assignment, overriding the current general part-time percentage and assignment level information. The FastFormulas will have no input values, returning a parameter called Part_Time_Percentage, in the format 999.9999.

If you want all part-time percentage values to be set from information at the assignment level, you do not need to define any overriding FastFormulas.

If you want to override all the part-time percentage values using the same calculation, you define one formula called NL_PART_TIME_PERCENTAGE.

If you need to override any of the part-time percentage values, you must create the appropriate formula from the following:

- Standard SI part-time percentage- NL_STANDARD_SI_PART_TIME_PERCENTAGE

- Pseudo SI part-time percentage- NL_PSEUDO_SI_PART_TIME_PERCENTAGE

- Standard SI part-time percentage for reporting- NL_STANDARD_SI_REPORTING_PART_TIME_PERCENTAGE

- Pseudo SI part-time percentage for reporting-
  NL_PSEUDO_SI_REPORTING_PART_TIME_PERCENTAGE

  **Note:** Pseudo SI Calculation uses the Pseudo SI Part-time Percentage. All the part-time percentages are limited according to the maximum limits set in the corresponding global values.

## Sample Formulas for Payment Method (Saudi)

Oracle provides sample formulas that you can use to define payment method as per your bank's requirement.

A payment type is required to identify the method of payment for example, SA Direct Deposit - SAR. This payment type should reference the generic magnetic tape procedure.

**Formulas:** The following formulas are created for Electronic File Transfer (EFT):

***FORMULAS:***

| Formula | Description |
|---|---|
| SA_EFT_HEADER | This formula is the header section of the payment output file. It calls the formula function SA_GET_CUSTOMER_FORMULA_HEADER which in turn fetches the various required fields for the header part. |
| SA_EFT_HEADER_CUSTOMER | This formula actually formats the fields fetched by the SA_EFT_HEADER formula as per the requirement. This formula is a sample and you can format the fields as per your requirement. |
| SA_EFT_BODY | This formula is the body section of the payment output file. It calls the formula function SA_GET_CUSTOMER_FORMULA_BODY which in turn fetches the various required fields for the body part. |

| Formula | Description |
|---|---|
| SA_EFT_BODY_CUSTOMER | This formula actually formats the fields fetched by the SA_EFT_BODY formula as per the requirement. This formula is a sample and you can format the fields as per your requirement. |
| SA_EFT_FOOTER | This formula is the footer section of the payment output file. It calls the formula function SA_GET_CUSTOMER_FOOTER which in turn fetches the various required fields for the footer section. |
| SA_EFT_FOOTER_CUSTOMER | This formula actually formats the fields fetched by the SA_EFT_FOOTER formula as per the requirement. This formula is a sample and you can format the fields as per your requirement. |

**Formula Functions:**The following formula functions are also created:

***FORMULA FUNCTIONS:***

| Formula Function | Description |
|---|---|
| SA_USER_CUSTOMER_FORMULA_HEADER | This formula function fetches the values for the fields, which are passed to it by the calling formula SA_EFT_HEADER. |
| SA_USER_CUSTOMER_FORMULA_BODY | This formula function fetches the values for the fields, which are passed to it by the calling formula SA_EFT_BODY. |
| SA_USER_CUSTOMER_FORMULA_FOOTER | This formula function fetches the values for the fields, which are passed to it by the calling formula SA_EFT_FOOTER. |

**Customer Formula (Sample)** Oracle Payroll provides a sample formula for you, to set up your own formulas for Header, Body and Footer. This sample is based on superset of data that supports EFT.

The following inputs are allowed in the header/footer and the body section of the payment file, to write your own formula:

**Header/Footer:**

- CREATION_DATE, this refers to the effective date on which the payment file is created as per Hijrah calendar (YYYYMMDD).
- PROCESS_DATE, this refers to the date on which the payment file is submitted as per Hijrah calendar (YYYYMMDD).
- COUNT 1, this refers to the total number of employees.
- SUM1, this refers to the total amount. You would multiply the total amount with hundred (100) to display the last two decimal points in your report.

**Body:**

- AMOUNT, this refers to the amount for each employee. You would multiply this amount with hundred (100) to display the last two decimal points in your report.
- FIRST_NAME, this refers to the first name of the employee.
- LAST_NAME, this refers to the last name of the employee.
- EMP_NO, this refers to the employee number.
- ASG_NO, this refers to the assignment number of the employee.
- LOCAL_NATIONALITY, this refers to the local nationality of the employee.

The formula that you have defined should return all the WRITE_TEXT, REPORT1_TEXT and REPORT2_TEXT variables that are set up in the formulas. WRITE_TEXT, REPORT1_TEXT, and REPORT2_TEXT are the outputs that are available for you.

At the maximum you have they can have five WRITE_TEXT, five REPORT1_TEXT and five REPORT2_TEXT variables. Each variable should not exceed 255 characters in size.

You use the following contexts for each section of your own formula and use database items for these values only.

**Header / Footer:**

- DATE_EARNED
- ORG_PAY_METHOD_ID
- BUSINESS_GROUP_ID
- PAYROLL_ID
- PAYROLL_ACTION_ID

**Body:**

- ASSIGNMENT_ID
- BUSINESS_GROUP_ID
- PER_PAY_METHOD_ID
- DATE_EARNED
- PAYROLL_ID
- PAYROLL_ACTION_ID
- ASSIGNMENT_ACTION_ID
- ORGANIZATION_ID
- TAX_UNIT_ID

Oracle may change this formula in future releases. This formula is strictly for example or prototype uses only, and is not intended to provide a ready-made solution. You can make a copy of this formula but you should not change your copied version. Always write your own formula to meet your bank's requirements. This formula may contain certain hard-coded values for simplified use.

The prototype formula supplied is only designed to work for payment method and should not be used for any other purpose. Any use of this formula is subject to the terms of Oracle license agreement for the HRMS programs and documentation.

**Sample for the Header is as follows:**

********** */

/* Formula Name: SA_EFT_HEADER

Description : This formula formats and writes header section of the Magnetic File. It also writes data onto the audit report.

********** */

```
/* Initialise database items */
DEFAULT FOR ORG_SA_BANK_NAME  IS ' '
DEFAULT FOR ORG_SA_ACCOUNT_NUMBER  IS ' '
INPUTS ARE CREATION_DATE (Text)
  ,PROCESS_DATE (Text)
    ,COUNT1 (Text)
  ,SUM1 (Text)
 TRANSFER_SUM1 = SUM1
TRANSFER_COUNT1 = COUNT1
/* Customer to modify section below this */
```

```
/* File Header */
WRITE_TEXT1 = '0' + /* Header Rec */
   '000000000000' + /* Key Header */
     'G' + /* Month Type */
CREATION_DATE +        /* File Creation Date */
 PROCESS_DATE + /* Salary Process Date */
'10000'      + /* Total Amount for All Employees*/
'100'        + /*  Total Employees*/
LPAD(ORG_SA_ACCOUNT_NUMBER,15,'0') + /* Company Account Number */
 RPAD(' ',68,' ') + /*Filler*/
CHR(10)
/* Report File Header */
REPORT1_TEXT1 = '            ' + 'Header Record:' + '0           ' +
CHR(10) + '       '+
 'Key Header :'   + LPAD('0',12,'0') + '
+ 'File Creation Date :' + LPAD(CREATION_DATE,12,'0') + CHR(10)
/*REPORT1_TEXT2 = 'Header Record:' + '0            ' +
'Month Type :' + LPAD('0',12,'0') + '        '
+ 'Salary Process Date :' + LPAD('0',12,'0') + CHR(10)
REPORT1_TEXT3 = 'Header Record:' + '0            ' +
'File Creation Date:' + LPAD('0',12,'0') +
CHR(10)
REPORT1_TEXT4 = 'Header Record:' + '0            ' +
'Salary Process Date:' + LPAD('0',12,'0') +
CHR(10)*/
REPORT1_TEXT2 = '           ' + 'Month Type :' + LPAD('GREGORIAN',
12,'0') + '       '
+ 'Salary Process Date :' + LPAD(PROCESS_DATE,12,'0') + CHR(10)
REPORT1_TEXT3 = ' '
REPORT1_TEXT4 = ' ''
REPORT1_TEXT5 =  '                    '+
'Total Employees :' + LPAD(COUNT1,12,'0')+
CHR(10)
/*Customer to modify section above this */
RETURN  WRITE_TEXT1, REPORT1_TEXT1,
REPORT1_TEXT2,REPORT1_TEXT3,REPORT1_TEXT4,REPORT1_TEXT5,TRANSFER_COUNT1,
TRANSFER_SUM1
```

## Sample Payroll Formulas Enabled for Proration (UK Only)

The following sample formulas show how you can create payroll formulas to be used in different situations requiring proration.

Oracle may change or upgrade these formulas in future releases of HRMS programs. These formulas are strictly for example or prototype purposes only, and are not intended to provide a ready-made solution to be used in your environment. You can make a copy of these formulas, subject to the terms of the license agreement for the programs, but you should not change your copied version for prototyping purposes. You should always write a new formula of your own to meet your particular requirements. These formulas may contain certain hard-coded values to simplify the use of formulas for proration.

A typical example of proration would be when a new employee starts work in the middle of a monthly payroll period and your payroll department makes a pro-rata payment to reflect the proportion of monthly pay to which the employee is entitled.

The prototype formula supplied is only designed to work for payroll calculations involving proration and should not be used for any other purpose. Any use of the formula is subject to the terms of the Oracle license agreement for the HRMS programs and documentation.

#### FastFormula to use if Employee is Paid using Salary Administration

```
/*************************************************

Formula name : FF_PRORATION_SAL_MANAGEMENT

Formula to calculate salary in proration

DISCLAIMER: Oracle may change or upgrade this FastFormula in the future
releases. This FastFormula is strictly for example or prototype purposes
only. This FastFormula is not intended to provide a ready-made solution
to the users. This formula may contain certain hard-coded values added
to simplify the concept of usage of FastFormulas in proration. Users
should make a copy of this formula and not change this formula. Users
need to write their own new formula(s) to meet requirements.

*************************************************

default for ASG_SALARY_BASIS_GRADE_ANNUALIZATION_FACTOR is 0

default for prorate_start is '01-JAN-1990' (date)

default for prorate_end is '01-JAN-1900' (date)

default for PAY_PROC_PERIOD_START_DATE IS '01-JAN-1950' (date)

default for PAY_PROC_PERIOD_END_DATE IS '01-JAN-1950' (date)

default for NI_NEW_TAX_YEAR IS '01-JAN-1950' (date)

default for annual_salary is 0.0

inputs are annual_salary (number),
```

```
prorate_start (date),

prorate_end (date)

/**

Prorate_start and prorate_end dates are passed from the payroll engine.
These dates basically represent the dates on which the changes occur in
the salary amount during the pay period. annual_salary is an input value
created in the element to which this formula will be tied. Here the
assumption is that the user enters the annual salary amount through
salary management.

**/

l_amount=annual_salary

l_string = ''

if(ASG_SALARY_BASIS_GRADE_ANNUALIZATION_FACTOR was not defaulted) then

(

/**

 This calculation makes sure that l_amount has the annual salary.

**/

l_amount =

amount*ASG_SALARY_BASIS_GRADE_ANNUALIZATION_FACTOR

)

/**

l_tax_year_start_date derives the value from the DBI

NI_NEW_TAX_YEAR.

The following calculation sets the start date of the tax year to 01-APR-
YYYY and end date of the tax year to 31-MAR-(YYYY+1)

**/

l_tax_year_start_date = NI_NEW_TAX_YEAR

l_tax_year_start_date = TRUNC(l_tax_year_start_date,'month')

l_tax_year =
```

```
TO_NUMBER(TO_CHAR(l_tax_year_start_date,'YYYY')) + 1

l_tax_year_end_date =

TO_DATE('31/03/'||TO_CHAR(1_tax_year),'DD/MM/YYYY')

/**

The following calculation is for the case when no proration occurs
during the pay period and element entry start date and element entry end
date is not equal to the pay period start and end dates respectively.



**/

IF(prorate_start was defaulted) then

(

IF(PAY_PROC_PERIOD_END_DATE<>ENTRY_END_DATE) THEN

(

IF(PAY_PROC_PERIOD_START_DATE<>ENTRY_START_DATE) THEN

(

/**

Since prorate_start date is defaulted, it means no proration occured,
therefore we simply return the annual salary amount divided by 12.

**/

l_amount = ROUND(amount/12,2)

result1 = l_amount

return result1

)

)

)

l_post_calc = 'N'

/**
```

```
l_post_calc is a flag used to do the calculation differently depending
on the values of this flag.

**/

/**

The following if condition takes care of the case when

Either

The element entry starts on the same day as pay period start date

OR

Prorate start date IS EQUAL TO element entry start date AND

Prorate start date IS NOT EQUAL TO one day prior to pay period start
date.

For example, in a monthly payroll running for the month of June 2000.

Either

Element entry start date is )01-JUN-2000 (employee starts on the first
day of the payroll period)

OR

Prorate start date is 14-JUN-2000 (employee starts in the middle of the
month or gets a salary change in the middle of the month)

Prorate start date IS NOT 31-MAY-2000

**/

if((prorate_start = ENRTY_START_DATE and

prorate_start <> ADD_DAYS(PAY_PROC_PERIOD_START_DATE,-1

)) or

ENTRY_START_DATE = PAY_PROC_PERIOD_START_DATE) then

(

/**

This code is executed when it is a new entry. Thus the post calc must be
performed.
```

```
**/

l_post_calc = 'Y'

)

else

(

if(prorate_end<>ENTRY_END_DATE) then

(

if(prorate_end = PAY_PROC_PERIOD_END_DATE) then

)

/**

This code is executed when it is a normal last proration entry. Thus the
post calc must be performed.

**/

l_post_calc = 'Y'

)

)

)

/**

The following code sets the right proration start and end dates*/

**/

if(prorate_start was defaulted) then

(

/**

Prorate start is defaulted when no proration occurs

**/

if(ENTRY_START_DATE=PAY_PROC_PERIOD_START_DATE) then
```

```
(
/**
Element entry start date is the same as the payroll period start date
**/
prorate_start=ENTRY_START_DATE
if(ENTRY_END_DATE=PAY_PROC_PERIOD_END_DATE) then
(
prorate_end=ENTRY_END_DATE
)
else
(
prorate_end=PAY_PROC_PERIOD_END_DATE
)
r)
else if(ENTRY_END_DATE=PAY_PROC_PERIOD_END_DATE) then
(
prorate_end=ENTRY_END_DATE
prorate_start=PAY_PROC_PERIOD_START_DATE
)
)
/*Do we have to do daily calc*/
if((prorate_start>PAY_PROC_PERIOD_START_DATE
and prorate_end<PAY_PROC_PEROD_END_DATE)
or
(prorate_start=PAY_PROC_PERIOD_START_DATE
```

```
and prorate_end<PAY_PROC_PERIOD_END_DATE

and prorate_start=ENTRY_START_DATE)) then

(

l_post_calc='D'

)

/*Now perform the calculations*/

if(l_post_calc='N') then

(

/*Perform in the pre mode, i.e.start of year to current date*/

l_days=days_between(prorate_end,

l_tax_year_start_date) + 1

l_amount1=l_amount * 1_days/365

l_months=TRUNC(months_between(PAY_PROC_PERIOD_START_DATE,

l_tax_year_start_date

))

l_amount2=1_amount * l_months/12

result1=1_amount1-l_amount2

)

else

)

if(l_postcalc+'Y') then

)

/*perform in post mode, i.e.current date to end of year*/

l_days=days_between(l_tax_year_end_date,

prorate_start) + 1
```

```
l_amount1=l_amount*l_days/365

l_months=TRUNC(months_between(l_tax_year_end_date,

PAY_PROC_PERIOD_END_DATE

l_amount2=l_amount*l_months/12

result1=l_amount1-l_amount2

)

else

(

/**

This code is executed when l_post_calc=D. Perform in daily mode.

**/


l_days=days_between(prorate_end,prorate-start) + 1

l_amount1=0

l_months=0

l_amount2=0

result1=l_amount*l_days/365

)

)

/**

The following code is just to convert all the non-character variables
into the character variables, so that the values of the variables could
be displayed in the messages available either in SOE form or messages.

**/

prorate_start_res=to_char(prorate_start,'DD-MON-YYYY')

prorate_end_res=to_char(prorate_end,'DD-MON-YYYY')

l_days_res=TO_CHAR(l_days)
```

```
l_amount1_res=TO_CHAR(l_amount1)

l_months_res=TO_CHAR(l_months)

l_amount2_res=TO_CHAR(l_amount2)

return result1,

l_post_calc,

l_days_res,

l_amount1_res,

l_months_res,

l_amount2_res,

prorate_start_res,

prorate_end_res
```

**FastFormula to use if Employee is Paid using a Spinal Point/Pay Scale**

```
/*************************************************************

Formula Name = UK_PRORATION_SPINAL_POINT

Formula Type = Oracle Payroll

Description: This formula is executed from within the payroll run by
processing the element UK Salary Spinal Point. It calculates the rate of
pay and returns this value. It uses the version of the function
RATES_HISTORY which requires a date input to be passed. This formula is
just a prototype.

DISCLAIMER: Oracle may change or upgrade this fast formula in the future
releases. This FastFormula is strictly for example or prototype
purposes. This FastFormula is not intended to provide a ready-made
solution to the users. This formula may contain certain hard-coded
values added to simplify the concept of usage of FastFormulas in
proration. Users should make a copy of this formula and not change this
formula. Users need to write their own new formula(s) to meet
requirements.

*************************************************************/

DEFAULT FOR date_worked IS '01-JAN-1950'(date)

DEFAULT FOR PAY_PROC_PERIOD_END_DATE IS '01-JAN-1950'(date)

DEFAULT FOR PAY_PROC_PERIOD_START_DATE IS '01-JAN-1950'(date)
```

```
DEFAULT FOR prorate_start IS '01-JAN-1950'(date)

DEFAULT FOR prorate_end IS '01-JAN-1950'(date)

inputs are prorate_start(date),

prorate_end(date)

/**

Prorate_start and prorate_end dates are passed from the payroll engine.
These dates basically represent the dates on which the changes occur in
the pay scale/grade rate during the pay period.

**/

amount=0

message=''

IF(prorate_start WAS DEFAULTED)then

(

/**

prorate_start date is defaulted when no proration occurs

**/

l_date_worked=PAY_PROC_PERIOD_END_DATE

prorate_start=PAY_PROC_PERIOD_START_DATE

prorate_end=PAY_PROC_PERIOD_END_DATE

)

else

)

l_date_worked=prorate_end

)

message1='Date defaulted to'||TO_CHAR(date_worked,'DD-MON-YYYY')

/**
```

```
The following function returns the value of the pay scale on the
l_date_worked which is last date of hte payroll period if no proration
occurs, otherwise it is the proration end date.

**/

if rates_history(l_date_worked,

'UK Spinal Point,

'E',

'P',

amount,

message)= -1 then

(

return message, message1

)

else

(

l_days = days_between(prorate_end,prorate_start)+ 1

/**

l_days stores the number of days between prorate start and prorate end
dates

**/

message1 = 'The value      is'||TO_CHAR(ROUND(amount,2))||for='||TO_CHAR
(l_days||'days from '||TO_CHAR(prorate_start,'DD-MON-YYYY')
||'to'||TO_CHAR(prorate_end,'DD-MON-YYYY')

/**

The message is just for informational purposes.

**/

l_amount=amount*(12*1_days/365)

/**
```

```
The above calculation is hard-coded to simplify the calculation. Please
write your own calculation logic. The assumption was that the value in
the pay scale is monthly. Therefore to get annual amount it is
multiplied by 12. Then it is divided by 365 days to get the amount for a
single day. Once it is multiplied by l_days, we get the amount for the
days we want.

**/

/**

It is another assumption that a year contains 365 days. Please change
this assumption to take account of the leap year. This formula will not
work properly in a leap year.

**/

return l_amount, message1

)
```

### FastFormula to Use if Employee is Paid Using a Grade Rate

```
/*************************************************************

Formula Name = UK_PRORATION_GRADE_RATE

Formula Type = Oracle Payroll

Description: This formula is executed from within the payroll run by
processing the element UK Salary Grade Rate. It calculates the rate of
pay and returns this value. It uses the version of the function
RATES_HISTORY which requires a date input to be passed. This formula is
just a prototype.

DISCLAIMER: Oracle may change or upgrade this FastFormula in the future
releases. This FastFormula is strictly for example or prototype purposes
only. This FastFormula is not intended to provide a ready-made solution
to the users. This formula may contain certain hard-coded values added
to simplify the concept of usage of FastFormulas in proration. Users
should make a copy of this formula and not change this formula. Users
need to write their own new formula(s) to meet requirements.

*************************************************************/

DEFAULT FOR date_worked IS '01-JAN-1950'(date)

DEFAULT FOR PAY_PROC_PERIOD-END_DATE IS '01-JAN-1950'(date)

DEFAULT FOR PAY_PROC_PERIOD-START-DATE IS '01-JAN-1950'(date)

DEFAULT FOR prorate_start IS '01-JAN-1950'(date)

DEFAULT FOR prorate_end IS '01-JAN-1950'(date)
```

```
inputs are prorate_start(date),

prorate_end(date)

/**

Prorate_start and prorate_end dates are passed from the payroll engine.
These dates basically represent the dates on which the changes occur in
the pay scale/grade rate during the pay period.

**/

amount = 0

message = ''

IF(prorate_start WAS DEFAULTED)then

(

/**

prorate_start date is defaulted when no proration occurs

**/

l_date_worked=PAY_PROC_PERIOD_END_DATE

prorate_start=PAY_PROC_PERIOD_START_DATE

prorate_end=PAY_PROC_PERIOD_END_DATE

)

else

(

l_date_worked=prorate_end

)

message1='Date defaulted to'||TO_CHAR(date_worked,'DD-MON-YYYY')

/**

The following function returns the value of the pay scale on the
l_date_worked which is last date of the payroll period if no proration
occurs, therwise it is the proration end date.
```

```
**/

if rates_history(l_date_worked,

'UK Grade Rate',

'E',

'H',

amount,

message) = -1 then

(

return message, message1

)

else

(

l_days = days_between(prorate_end,prorate_start) + 1

/**

l_days stores the number f days between prorate start and prorate end
dates

**/

message1 = 'The value      is'||TO_CHAR(ROUND(amount,2))||for='||TO_CHAR
(1_days||'days from '||TO_CHAR(prorate_start,'DD-MON-YYYY')
||'to'||TO_CHAR(prorate_end,'DD-MON-YYYY')

/**

The message is just for informational purposes.

**/

l_amount=amount * ((l_days*1600)/365)

/**
```

```
The above calculation is hard-coded to simplify the calcuation. Please
write your own calculatioin logic. The assumption was that the value in
the grade rate is hourly. Therefore to get the annual amount it is
multiplied by 16000 hours (assuming that there are 1600 hours in a
year). Then it is divided by 365 days to get the amount for a single
day. Once it is multiplied by l_days, we get the amount for the days we
want.

**/

/**

It is another assumption that a year contains 365 days. Please change
this asusmption to take account of leap year. This formula will not work
properly in a leap year.

**/

return l-amount,message1

)
```

**FastFormula to use for a Deduction**

```
/*****************************************************************

Formula name: UK_PRORATION_DEDUCTIONS

Purpose: Formula prototype to calculate prorated deductions.

DISCLAIMER: Oracle may change or upgrade this FastFormula in future
releases. This FastFormula is strictly for example or prototype purposes
only. This FastFormula is ot intended to provide a ready-made solution
for the user. This formula may contain certain hard-coded values added
to simplify the concept of usage of FastFormulas in proration. Users
should make a copy of this formula and not change this formula. Users
need to write their own new formula(s) to meet requirements.

*****************************************************************/

default for prorate_start is '01-JAN-1900'(date)

default for prorate_end is '01-JAN-1900'(date)

default for amount is 0.0

inputs are

prorate_start(date),

prorate_end(date),

annual_deduction(number)
```

```
/**

Prorate-start and prorate_end dates are passed from the payroll engine.
These dates basically represent the dates on which the changes occur in
the deduction amount during the pay perod. annual_deduction is an input
value created in the element to which this formula will be tied. Here
the assumption is that the user enters the annual deduction amount in
the input value.

**/

l_amount=annual_deduction

message='Proration Start Date'||TO_CHAR(prorate_start, 'DD-MON-YYYY')

IF(prorate_start was defaulted)then

(

/**

prorate_start date is defaulted when no proration occurs. Therefore we
should just return the annual deduction amount divided by 12.

**/

l_amount=l_amount/12

return l_amount

)

else

(

l_days=days_between(prorate_end, prorate_start) + 1

/**

l_days stores the number of days between prorate start and prorate end
dates

**/

l_days_in_fiscal_year=365



/**
```

```
It is another assumption that a year contains 365 days. Please change
this assumption to take care of leap year. This formula will not work
properly in a leap year.

**/

l_amount=(l_amount*l_days)/l_days_in_fiscal_year

/**

In the above calculation, since l_amount contains the annual deduction
amount, it is divided by 365 days of the year to get the deduction
amount per day. This amount is then multiplied by the number of days in
question to get the proper deduction amount

**/

return l_amount, message

)
```

## Sample Rates History Formulas (UK Only)

The following sample formulas show how you can create payroll formulas to be used in different situations requiring historic rates.

Oracle may change this formula in future releases. This formula is strictly for example or prototype uses only, and is not intended to provide a ready-made solution. You can make a copy of this formula but you should not change your copied version. Always write a formula of your own to meet your own requirements. This formula may contain certain hard-coded values for simplified use.

#### Sample FastFormula to Derive a Historic Rate from a Single Element

```
/*

Formula name : SAMPLE_OVERTIME_PAY_USING_AN_ELEMENT_RATE

Description : Sample formula showing the use of historic rate to
calculate overtime

In this sample, the GET_HISTORIC_RATE function will reference the
'Overtime' element

*/

DEFAULT FOR date_worked IS '1951/01/01 00:00:00' (date)

DEFAULT FOR hours_worked IS 0

INPUTS ARE date_worked(date)
           , number_of_hours_worked
```

```
hourly_rate =
  GET_HISTORIC_RATE
   ('Overtime'       /* Name of the rate to be calculated */
   , date_worked     /* Date as of which the rate is required */
   , 'H'             /* Time dimension required, 'H' for Hourly */
   )

overtime pay = hourly_rate * number_of_hours_worked


RETURN overtime_pay
       , hourly rate
```

#### Sample FastFormula to Derive the Combined Historic Rate from More than One Element

```
/*

Formula name : SAMPLE_OVERTIME_PAY_USING_A_RATE_TYPE


Description : Sample formula showing the use of historic rate to
calculate the a pension deduction

In this sample, the GET_HISTORIC_RATE will identify all elements
attached.

*/
DEFAULT FOR employee_contribution IS 5.0

INPUTS employee_contribution     /* Percentage */
effective_date = SESSION_DATE


annual_salary_rate =
   GET_HISTORIC_RATE
    ('Pensionable Salary'    /* Name or the rate to be calculated. */
    , effective_date         /* Date as of which the rate is required.
*/
    , 'A'                    /* Time Dimension required, 'A' for Annual.
*/
    , 'R'                    /* Specify whether the rate name is a Rate
Type */
    )

pension_deduction = (annual_salary_rate * (employee_contribution / 100 )

RETURN ,pension_deduction
       ,annual_salary_rate
```

#### Parameters for the Historic Rates Function

The sample formulas can call the GET_HISTORIC_RATE formula function. The contexts, parameters and return values for the Rates History function are as follows:

**GET_HISTORIC_RATE**

GET_HISTORIC_RATE (rate_name, date, time_dimension, rate_type_or_element, contract_type, contract_type_usage).

The value returned by the function represents the rate as of the given date. The value returned is of type Number.

You can only use this function in formula types that have a context of Assignment_ID, for example, Oracle Payroll, QuickPaint.

| Parameter Name | Description |
|---|---|
| P_RATE_NAME | The name of the rate type or element that you want to process. The actual meaning of the parameter depends on what was passed in the P_RATE_TYPE_OR_ELEMENT parameter. By default, the rate name must be an element type. |
| P_EFFECTIVE_DATE | The date from which you want to calculate the rate. |
| P_TIME_DIMENSION | The time dimension in which you want to return the retrieved rate. If the p_rate_name references an element and you omit the time dimension in the GET_HISTORIC_RATE call, then the default time dimension returned is the same as the source time dimension that you defined on the element extra information Historic Rate - Element Attribution. You must supply a valid time dimension when the rate name references a rate type. |
| P_RATE_TYPE_OR_ELEMENT | Identifies whether the rate name is an element or rate type. By default, the rate name is an element type. Valid values for this parameter are:<br>• 'E' if P_RATE_NAME is an element name<br>• 'R' if P_RATE_NAME is a rate type name |
| P_CONTRACT_TYPE | The name of a default or override contract type. This parameter specifies the name of the contract type that the GET_HISTORIC_RATE function uses. |

| Parameter Name | Description |
|---|---|
| P_CONTRACT_TYPE_USAGE | Specifies whether the contract in P_CONTRACT_TYPE has a default assignment level contract type, or overrides an assignment level contract level type. You define assignment level contracts on the Extra Details of Service window. Valid values for this parameter are:<br>• 'DEFAULT' which causes the GET_HISTORIC_RATE function to use the contract type named in P_CONTRACT_TYPE only when an assignment level contract does not exist at the effective date.<br>• 'OVERRIDE' which causes the GET_HISTORIC_RATE function to always use the contract type named in P_CONTRACT_TYPE, even if an assignment level contract exists. |

## Sample Deduction Formula Calling the Arrearage Function (UK Only)

The following sample formula shows how you can create payroll formulas for pre-tax or voluntary deductions if you want to maintain an arrears balance. This formula does not handle iterative processing for pre-tax deductions. If you are writing a formula for a pre-tax deduction and you want it to handle iterative processing too, use the seeded GAYE formula as the example on which to base your own formula.

Oracle may change this formula in future releases. This formula is strictly for example or prototype uses only, and is not intended to provide a ready-made solution.

```
/*
============================================================================
===
* SECTION (1):Default values for database items
*
============================================================================
===
*/
Default for NET_PAY_ASG_RUN Is 0
/*
Uncomment this code if your deduction has a Clear Arrears input value.
Default for Clear_Arrears   Is 'N'
```

```
*/

/*
============================================================================
===

* SECTION (2): Element Input Values

*
============================================================================
=== */

Inputs are Amount,

Percentage

/* If your deduction has a Clear Arrears input value, add Clear_Arrears
(text) to the Inputs statement. */

/*
============================================================================
===

* SECTION (3): Initialize Local Variables

*
============================================================================
===

*/

/*
============================================================================
=

* SECTION (3.1): Default Iterative Arrearage Function's Out Values

*
============================================================================
=

*/

l_Actual_Usercalc_Amt = 0 /* Act.Ded.Amt for the Period
*/

l_Max_Amount          = 0 /* Max Amt that can be taken, on the earnings
*/

l_Min_Amount          = 0 /* Min Amt usually set to zero
*/

l_Not_Taken           = 0 /* Amt that could'nt be taken due to insuff.
earnings*/

l_To_Arrears          = 0 /* Amt that is added to Arrears, to be rec.
later */

l_Arrears_Taken       = 0 /* Amt that was taken from the Arrears
*/

l_Error_Message       = ' ' /* Error message from the Arrearage Function
*/

l_Warning_Message     = ' ' /* Warning message from the Arrearage
Function*/

l_Return_Value        = 0   /* Iterative Arrearage function return value
*/

l_mesg_1 = ' ' l_mesg_2 = ' '

/*
============================================================================
=
```

```
* SECTION (3.2): Default Local variables with DB items
*
==========================================================================
=
*/
l_clr_add_amt         = 0   /* Additional Amount          */
l_clr_rep_amt         = 0   /* Replacement Amount         */
l_deduction_amount    = 0   /* Actual Deduction Amount    */
l_arrears_allowed     = ' '
l_partial_allowed     = ' '
/*
==========================================================================
===
* SECTION (4): Formula Calulation
*
==========================================================================
===
*/
/*
==========================================================================
=
* SECTION (4.1): Calculate the actual deduction amount
*
==========================================================================
=
*/
l_Actual_UserCalc_Amt = Amount
l_deduction_amount    = Amount
l_mesg_1 = l_mesg_1 ||' Act.Ded.Amt='||to_char(l_Actual_UserCalc_Amt)
/*
==========================================================================
=
* SECTION (4.2): Get Outstanding Arrears and calculate the remaining
amount
* i.e. the Limit Amount and Guarantee Net Amount, if any.
*
==========================================================================
=
*/
l_Return_Value = PQP_GET_ARREARAGE_OPTIONS(l_arrears_allowed
,l_partial_allowed
,l_Error_Message)
l_Remaining_Amt  = 0 /*If Voluntary deduction has limits, specify the
rem. amt */
l_MaxArrears_Amt = 0 /*Out standing Arrears for the element if Arrears
allowed */
l_Guaranteed_Net = 0
```

```
/*
=============================================================================
=
* SECTION (4.3): Call the Arrearage Function
*
=============================================================================
=
*/
l_temp_ToArrears = 0
l_temp_NotTaken  = 0
/* Uncomment this code only if the element has a Clear Arrears
* input value that has a value Y or N.
If the deduction has a limit, set l_remaining_amt to the limit minus the
YTD deduction balance.
*/
/*
If Clear_Arrears = 'N' Then
(
l_MaxArrears_Amt = <ARREARS_BALANCE_NAME_YTD>
l_remaining_amt  = l_MaxArrears_Amt
)
Else
(
l_To_Arrears     = -1 * <ARREARS_BALANCE_NAME_YTD>
l_MaxArrears_Amt = 0
)
*/
l_Deduction_Amount = PQP_Arrearage
(NET_PAY_ASG_RUN     /* p_net_asg_run      */
,l_MaxArrears_Amt    /* p_maxarrears       */
,l_Deduction_Amount  /* p_dedn_amt         */
,l_temp_ToArrears    /* p_to_arrears      In Out */
,l_temp_NotTaken     /* p_not_taken       In Out */
,l_Arrears_Taken     /* p_arrears_taken   In Out */
,l_Remaining_Amt     /* p_remaining_amount */
,l_Guaranteed_Net    /* p_guaranteed_net   */
)
If l_temp_ToArrears <> 0
/* If the deduction has a Clear Arrears input value, uncomment the
following line*/
/* and Clear_Arrears <> 'Y' */
Then
l_To_Arrears = l_To_Arrears + l_temp_ToArrears
```

```
If l_temp_NotTaken <> 0 Then
l_Not_Taken  = l_temp_NotTaken
l_mesg_2 = l_mesg_2 ||'Arr.Ded.Amt='||to_char(l_Deduction_Amount)
l_mesg_2 = l_mesg_2 ||' Arrs='||to_char(l_To_Arrears)
l_mesg_2 = l_mesg_2 ||' NotT='||to_char(l_Not_Taken)
/*
==============================================================================
=
* SECTION (4.6): Formula Run Results values
*
==============================================================================
=
*/
Return l_Deduction_Amount,
l_Not_Taken,
l_To_Arrears,
l_mesg_1,
l_mesg_2
```

# Legislative Functions

## Canadian Legislative Functions

Oracle HRMS for Canada provides the following functions for use in formulas:

### CALC_YEARS_OF_SERVICE

This function returns the years of service of an employee assignment. It uses the following fast formula contexts:

- ASSIGNMENT_ID
- DATE_EARNED

Return Data Type: Number

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_data_type | This parameter determines starting date to be used to calculate the period of service. | I | HD |

### GET_RATE_ID_FOR_JOB

This function returns the rate_id of a WCB account related to a particular job.

Return Data Type: Number

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_job | Job_id of the job that has the associated WCB account attached. | I | |
| P_account_number | The Account Number of the WCB Account. | I | |

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| P_jurisdiction | Jurisdiction Code associated with the WCB Account. | I | |

## GET_RATE_ID_FOR_WCB_CODE

This function returns the rate of a WCB account related to a particular job.

Return Data Type: Number

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| P_account_number | Account Number of the WCB account. | I | |
| P_code | The Rate Code of the WCB Account. | I | |
| P_jurisdiction | Jurisdiction Code associated with the WCB Account. | I | |

## GET_WCB_RATE

This function returns the rate of a WCB account.

Return Data Type: Number

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| P_rate_id | Rate ID of a WCB Account. | I | |

## VALIDATE_PMED_ACCOUNT_NUMBER

This function validates the PMED account number being passed, and it returns S (Success) or E(Error) based upon the validation. It uses the BUSINESS_GROUP_ID fast formula context.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| P_account_number | PMED Account Number. | I | |

## VALIDATE_USER_TABLE_COLUMN

This function validates the column of a USER TABLE being passed, and it returns S (Success) or E(Error) based upon the validation. It uses the BUSINESS_GROUP_ID fast formula context.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_user_table_column | Column Name of the USER TABLE. | I | |

## VALIDATE_USER_TABLE_NAME

This function validates the name of a USER TABLE being passed, and it returns S (Success) or E(Error) based upon the validation. It uses the BUSINESS_GROUP_ID fast formula context.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_user_table_name | Name of the USER TABLE. | I | |

## VALIDATE_WCB_ACCOUNT_NUMBER

This function validates the Account Number of the WCB Account being passed, and it returns S(Success) or E(Error) based upon the validation. It uses the BUSINESS_GROUP_ID fast formula context.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_account_number | Account Number on the WCB Account. | I | |

## VALIDATE_WCB_RATE_CODE

This function validates the Rate Code of the WCB Account being passed, and it returns S(Success) or E(Error) based upon the validation. It uses the BUSINESS_GROUP_ID fast formula context.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_rate_code | Rate Code on the WCB Account. | I | |

## GET_EMP_ADDRESS_INFO

This function returns the current address of an employee.

Return Data Type: Number

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_person_id | Person_id of the employee for whom we need the current address. | I | |
| p_address1 | Returns the Address line 1 of the employee address. | O | |
| p_address2 | Returns the Address line 2 of the employee address. | O | |

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_address3 | Returns the Address line 3 of the employee address. | O | |
| p_city | Returns the City of employee address. | O | |
| p_postal_code | Returns the pin code of the employee address. | O | |
| p_country | Returns the country of the employee address. | O | |
| p_province | Returns the province of the employee address. | O | |

## VALIDATE_GRE_DATA

This function validates the data for the T4 transmitter GRE passed and all the other T4 GREs that belong to the same transmitter GRE.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_trans | T4 Transmitter GRE. | I | |
| P_year | The Year for which the transmitter GRE needs to be validated. | I | |

## CAEOY_GET_FOOTNOTE_AMOUNT

This function returns the T4A Footnote amount archived against the footnote code once the Federal yearend process for T4A is completed. It expects the ASSIGNMENT_ACTION_ID context to be set.

Return Data Type: Number

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_footnote_code | The Footnote for which we need to fetch the amount. | I | |

## DD_CONVERT_UPPERCASE

This function removes the accents from a string and returns it with the corresponding string without accents.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| p_input_string | Strings with accents so that the accents can be removed and returned. | I | |

## CHECK_AGE_UNDER18_OR_OVER70

This function returns a flag that identifies whether an employee is under 18 or over 70 as of the Date Paid (Effective Date of pay_payroll_action) of the run. This function also expects the PAYROLL_ACTION_ID context to be set.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| PER_DATE_OF_BIRTH | The Date of Birth of the employee for whom the check needs to be done. | I | |
| p_input_string | Strings with accents so that the accents can be removed and returned. | I | |

## CHECK_AGE_UNDER18

This function returns a flag that identifies if an employee is under 18 as of the Date Paid (Effective Date of pay_payroll_action) of the run. This function also expects the PAYROLL_ACTION_ID context to be set.

Return Data Type: Text

| Parameter Name | Description | In/Out/Both | Valid Values |
|---|---|---|---|
| PER_DATE_OF_BIRTH | The Date of Birth of the employee for whom the check needs to be done. | I | |

## Private Functions

- CA_GARN_BC_EXEMPT
- CAEOY_GET_PENSION_PLAN_REGNO
- CALCULATE_PERIOD_EARNINGS
- CHECK_AGE_UNDER18
- CONVERT_PERIOD_TYPE
- CONVERT_2_XML
- GET_DB_VALUE
- GET_EMPLOYEE_ITEM
- GET_FILE_CREATION_NUMBER
- GET_TRANSMITTER_ITEM
- MULTI_ASG_PRORATION_REGULAR
- STANDARD_HOURS_WORKED
- WORK_SCHEDULE_TOTAL_HOURS

## Hungarian Legislative Functions

Oracle HRMS for Hungary provides the following functions for use in formulas:

### HU_ABS_GET_BLIND_DAYS

This formula function returns the number of days in the period for which the employee is blind.

### HU_ABS_GET_CHILD_INFO

This formula function returns the child information such child or children aged under 16 in the start of the period, child or children turning 16 in the given period and date of birth of any children turning 16 in the given period.

### HU_ABS_GET_JOB_DAYS

This formula function returns the number of days for the job with the Additional Holiday set to Yes for the calculation year.

### HU_ABS_GET_PREV_EMP_SICKNESS_LEAVE

This formula function returns the amount of sickness leave taken in the previous employment for the calculation year.

### HU_ABS_GET_WORKING_DAYS

This formula function returns the number of working days for the assignment for the given period depending on the work pattern attached to the assignment.

### HU_ENTRY_IN_ACCRUAL_PLAN

This formula function returns the valid values for the holiday entitlements such as HU1 for Base Holiday, HU2 for Additional Holiday for bringing up children, HU3 for Other Additional Holiday, and HU4 for Sickness Holiday.

### HU_PAYROLL_PERIODS

This formula function returns the number of payroll periods per year.

### HU_PERSON_DOB

This formula function returns the date of birth of the employee.

## Japanese Legislative Functions

The following functions have been registered for use in Japanese legislative formulas:

### CHECK_FORMAT

(expr, fmt)

The CHECK_FORMAT function returns TRUE or FALSE to verify that the text string operand expr matches the fmt type as follows:

| Format Type | Validation |
|---|---|
| 0:0-9 | Numbers only, non-omissible |
| 9:0-9 | Numbers only, omissible |
| A:A-Z | Capital alphabet only, non-omissible |
| P:A-Z | Capital alphabet only, omissible |
| a:a-z | Small alphabet only, non-omissible |
| p:a-z | Small alphabet only, omissible |
| L:0-9, A-Z | Numbers and capital alphabet only, non-omissible |
| C:0-9, A-Z | Numbers and capital alphabet only, omissible |
| l:0-9, a-z | Numbers and small alphabet only, non-omissible |
| c:0-9, a-z | Numbers and small alphabet only, omissible |

Example:

```
CHECK_FORMAT ('123456ABC', '999999PPL') returns TRUE
```

### CHECK_DATE_FORMAT

(date, date fmt)

The CHECK_DATE_FORMAT function returns TRUE or FALSE to verify that the date operand matches a given date fmt.

Example:

CHECK_DATE_FORMAT ('19990623', 'YYYYMMDD') returns TRUE

### GET_LAST_ASSACT

(date1, date2)

The GET_LAST_ASSACT function returns latest assignment_action_id with 'SEQUENCED' classification_name between date1 and date2. Action_types with 'SEQUENCED' classification_name are as follows:

| ACTION_TYPE | MEANING | CLASSIFICATION_NAME |
|---|---|---|
| B | Balance Adjustment | SEQUENCED |
| F | Advance Pay | SEQUENCED |
| I | Balance Initialization | SEQUENCED |
| O | RetroPay | SEQUENCED |
| Q | QuickPay Run | SEQUENCED |
| R | Run | SEQUENCED |
| V | Reversal | SEQUENCED |
| Z | Purge | SEQUENCED |

### ORG_EXISTS

(organization_id, org_class)

The ORG_EXISTS function returns TRUE or FALSE to verify that the organization ID is in the organization class..

Example:

```
ORG_EXISTS(12345, 'JP_KENPO')
```

## Mexican Legislative Functions

Oracle HRMS for Mexico provides the following functions for use in formulas:

## CALCULATE_ISR_TAX

This function calculates the ISR Tax for Mexico.

SOURCE: PAY_MX_TAX_FUNCTIONS.CALCULATE_ISR_TAX

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | PAYROLL_ACTION_ID |
| 2 | ASSIGNMENT_ACTION_ID |
| 3 | BUSINESS_GROUP_ID |
| 4 | ASSIGNMENT_ID |
| 5 | TAX_UNIT_ID |
| 6 | DATE_EARNED |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_SUBJECT_AMOUNT | I | Amount Subject to Tax | |
| 2 | P_ISR_RATES_TABLE | I | ISR Tax Rates table | |
| 3 | P_SUBSIDY_TABLE | I | ISR Tax Subsidy table | |
| 4 | P_CREDIT_TO_SALARY_TABLE | I | ISR Subsidy for Employment Table | |

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 5 | P_ISR_CALCULATED | O | ISR Calculated amount | |
| 6 | P_ISR_CREDITABLE_SUBSIDY | O | Prorated ISR Subject amount | |
| 7 | P_ISR_NON_CREDITABLE_SUBSIDY | O | Not used | |
| 8 | P_CREDIT_TO_SALARY | O | Amount considered as Credit to Salary | |
| 9 | P_CREDIT_TO_SALARY_PAID | O | Amount Paid as Credit to Salary | |

## CALCULATE_ISR_TAX_113

This function calculates the ISR Tax for Mexico.

SOURCE: PAY_MX_TAX_FUNCTIONS.CALCULATE_ISR_TAX

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | PAYROLL_ACTION_ID |
| 2 | ASSIGNMENT_ACTION_ID |
| 3 | BUSINESS_GROUP_ID |
| 4 | ASSIGNMENT_ID |
| 5 | TAX_UNIT_ID |
| 6 | DATE_EARNED |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_CALC_MODE | I | Article to be used in ISR Tax Calculation | ARTICLE113<br>ARTICLE142 |
| 2 | P_SUBJECT_AMOUNT | I | Amount Subject to Tax | |
| 3 | P_ISR_RATES_TABLE | I | ISR Tax Rates table | |
| 4 | P_SUBSIDY_TABLE | I | ISR Tax Subsidy table | |
| 5 | P_CREDIT_TO_SALARY_TABLE | I | ISR Subsidy for Employment Table | |
| 6 | P_ISR_CALCULATED | O | ISR Calculated amount | |
| 7 | P_ISR_CREDITABLE_SUBSIDY | O | Prorated ISR Subject amount | |
| 8 | P_ISR_NON_CREDITABLE_SUBSIDY | O | Not used | |
| 9 | P_CREDIT_TO_SALARY | O | Amount considered as Credit to Salary | |
| 10 | P_CREDIT_TO_SALARY_PAID | O | Amount paid as Credit to Salary | |

## CALCULATE_ISR_TAX_114

This function calculates/adjusts the ISR Tax for Mexico.

SOURCE: PAY_MX_TAX_FUNCTIONS.CALCULATE_ISR_TAX

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | PAYROLL_ACTION_ID |
| 2 | ASSIGNMENT_ACTION_ID |
| 3 | BUSINESS_GROUP_ID |
| 4 | ASSIGNMENT_ID |
| 5 | TAX_UNIT_ID |
| 6 | DATE_EARNED |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_RUN_TYPE | I | Tax Adjustment Type | ADJTAX<br>MTDTAXADJ |
| 2 | P_CALC_MODE | I | Article to be used in ISR Tax Calculation | ARTICLE113 |
| 3 | P_SUBJECT_AMOUNT | I | Amount Subject to Tax | |
| 4 | P_ISR_RATES_TABLE | I | ISR Tax Rates table | |
| 5 | P_SUBSIDY_TABLE | I | ISR Tax Subsidy table | |
| 6 | P_CREDIT_TO_SALARY_TABLE | I | ISR Subsidy for Employment Table | |

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 7 | P_ISR_CALCULATED | O | ISR Calculated amount | |
| 8 | P_ISR_CREDITABLE_SUBSIDY | O | Prorated ISR Subject amount | |
| 9 | P_ISR_NON_CREDITABLE_SUBSIDY | O | Not used | |
| 10 | P_CREDIT_TO_SALARY | O | Amount considered as Credit to Salary | |
| 11 | P_CREDIT_TO_SALARY_PAID | O | Amount paid as Credit to Salary | |

## CALCULATE_ISR_TAX_ANNUAL_ADJ

This function calculates/adjusts the ISR Tax for Mexico Annual Tax Adjustment.

SOURCE: PAY_MX_TAX_FUNCTIONS.CALCULATE_ISR_TAX

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | PAYROLL_ACTION_ID |
| 2 | ASSIGNMENT_ACTION_ID |
| 3 | BUSINESS_GROUP_ID |
| 4 | ASSIGNMENT_ID |
| 5 | TAX_UNIT_ID |
| 6 | DATE_EARNED |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_CALC_MODE | I | Calculation Mode for Annual Tax Adjustment | ARTICLE177 |
| 2 | P_SUBJECT_AMOUNT | I | Amount Subject to Tax | |
| 3 | P_ISR_RATES_TABLE | I | ISR Tax Rates table | |
| 4 | P_SUBSIDY_TABLE | I | ISR Tax Subsidy table | |
| 5 | P_CREDIT_TO_SALARY_TABLE | I | ISR Subsidy for Employment Table | |
| 6 | P_ISR_CALCULATED | O | ISR Calculated amount | |
| 7 | P_ISR_CREDITABLE_SUBSIDY | O | Prorated ISR Subject amount | |
| 8 | P_ISR_NON_CREDITABLE_SUBSIDY | O | Not used | |
| 9 | P_CREDIT_TO_SALARY | O | Amount considered as Credit to Salary | |
| 10 | P_CREDIT_TO_SALARY_PAID | O | Amount paid as Credit to Salary | |

## CHECK_EE_EMPLOYMENT_CRITERIA

This function determines if an employee meets the continuous employment criteria. Returns 'Y' if employee has worked continuously between the given Start date and End

date.

Only those employees who have maintained continuous employment with the employer for the period specified by the Adjustment Start Date and Adjustment End Date fields are considered eligible for Annual Tax Adjustment.

SOURCE: PAY_MX_TAX_FUNCTIONS.CHECK_EE_EMPLOYMENT_CRITERIA

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ACTION_ID |
| 2 | DATE_EARNED |

PARAMETERS: None

## CHECK_EE_LEGAL_EMPLOYER_CHANGE

This function determines if an employee has changed legal employer in the middle of the year or not. Returns Y if Employee has changed Legal Employer in the middle of the Year.

According to SAT Statutory rules, employees who have changed legal employer in the middle of the year are not eligible for Annual Tax Adjustment. Customers must modify their existing formula text, associated to Annual Tax Adjustment Assignment Set, by adding the formula function call. For more information refer to the Oracle HRMS Payroll Management Guide (Mexico).

SOURCE: PAY_MX_TAX_FUNCTIONS.CHECK_EE_LEGAL_EMPLOYER_CHANGE

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ACTION_ID |
| 2 | DATE_EARNED |

PARAMETERS: None

## CHECK_EE_SAL_CRITERIA

This function determines if the employee meets the salary criteria for Annual Tax Adjustment. Returns Y if persons gross YTD earnings are less than a specified cap.

According to SAT Statutory rules, only employees who have earned less than $400,000 Pesos are eligible for Annual Tax Adjustment.

SOURCE: PAY_MX_TAX_FUNCTIONS.CHECK_EE_SAL_CRITERIA

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ACTION_ID |
| 2 | DATE_EARNED |

PARAMETERS: None

## CONVERT_INTO_MONTHLY_SALARY

This function determines the Monthly Salary based on the given Earnings, Payroll Period Type and Number of Days in a Month specified for GRE.

SOURCE: PAY_MX_TAX_FUNCTIONS.CONVERT_INTO_MONTHLY_SALARY

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |
| 3 | PAYROLL_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_PERIODIC_EARNINGS | I | Period Earnings | |

## CONVERT_PERIOD_TYPE

This function converts the given amount per one frequency/period into amount per another frequency/period, based on the Assignment Work Schedule, Standard Hours, Payroll Period and Frequency.

SOURCE: PAY_MX_FF_UDFS.CONVERT_PERIOD_TYPE

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |
| 3 | PAYROLL_ID |

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_ASST_WORK_SCHEDULE | I | Assignment Work Schedule | |
| 2 | P_ASST_STD_HOURS | I | Assignment Standard Hours | |
| 3 | P_FIGURE | I | Amount to be converted | |

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 4 | P_FROM_FREQ | I | Convert from Frequency | |
| 5 | P_TO_FREQ | I | Convert to Frequency | |
| 6 | P_PERIOD_START_DATE | I | Period Start Date | |
| 7 | P_PERIOD_END_DATE | I | Period End Date | |
| 8 | P_ASST_STD_FREQ | I | Assignment Working Hours Frequency Code | |

## DAYS_IN_PAY_PERIOD

This function determines the number of days in a period based on Payroll frequency.

SOURCE: PAY_MX_UTILITY.GET_DAYS_IN_PAY_PERIOD

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |
| 3 | PAYROLL_ID |

PARAMETERS: None

## getDaysInBiMonth

This function determines the total number of days in current and previous months.

SOURCE: PAY_MX_UTILITY.GET_DAYS_IN_BIMONTH

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly): None

PARAMETERS: None

## getDaysInMonth

This function determines the number of days in a month, based on details setup at GRE or Legal Employer. If it is not set, this function will return the actual number of days in the current month.

SOURCE: PAY_MX_UTILITY.GET_DAYS_IN_MONTH

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |
| 3 | PAYROLL_ID |

PARAMETERS: None

## GetIDW

This function determines the Fixed, Variable, and Total IDW values for the current assignment as on 'Date Earned' context.

SOURCE: PAY_MX_FF_UDFS.GET_IDW

RETURN DATA TYPE: Number

Return Data Type: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ID |
| 2 | TAX_UNIT_ID |

| SEQUENCE | CONTEXT |
|---|---|
| 3 | DATE_EARNED |
| 4 | PAYROLL_ACTION_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_MODE | I | IDW Calculation Mode | REPORT |
| 2 | P_FIXED_IDW | O | Fixed Portion of IDW | |
| 3 | P_VARIABLE_IDW | O | Variable Portion of IDW | |
| 4 | P_EXECUTE_OLD_IDW_CODE | I | Parameter to indicate execution of new code | Y |

## GETIDW_BYDATE

This function determines the Fixed, Variable, and Total IDW values for the current assignment as on a given date.

SOURCE: PAY_MX_FF_UDFS.GET_IDW_BYDATE

RETURN DATA TYPE: Number

Return Data Type: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ID |

| SEQUENCE | CONTEXT |
|---|---|
| 2 | TAX_UNIT_ID |
| 3 | DATE_EARNED |
| 4 | PAYROLL_ACTION_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_MODE | I | IDW Calculation Mode | REPORT |
| 2 | P_FIXED_IDW | O | Fixed Portion of IDW | |
| 3 | P_VARIABLE_IDW | O | Variable Portion of IDW | |
| 4 | P_EXECUTE_OLD_IDW_CODE | I | Parameter to indicate execution of new code | Y |
| 5 | P_CALCULATION_DATE | I | Date as on which IDW to be calculated | |

## GET_BASE_PAY_FOR_TAX_CALC

This function determines the Base Pay of an Employee.

SOURCE: PAY_MX_FF_UDFS.GET_BASE_PAY_FOR_TAX_CALC

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | ASSIGNMENT_ID |
| 3 | TAX_UNIT_ID |
| 4 | PAYROLL_ID |
| 5 | DATE_EARNED |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_MONTH_OR_ PAY_PERIOD | I | Parameter to indicate the Base Pay calculation for a Month or Pay Period | MONTH<br>PAY_PERIOD |

## GET_EMPLOYER_PAY_EMP_SSQUOTA

This function returns Y if 'ER pay SS Quotas if EE Sal equal to Min Wage' setup for a given GRE is set to Yes.

SOURCE: PAY_MX_FF_UDFS.GET_EMPLOYER_PAY_EMP_SSQUOTA

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | TAX_UNIT_ID |

PARAMETERS: None

## GET_HIRE_DATE

This function determines the Adjusted Service Date / Original Hire Date of an Employee.

SOURCE: HR_MX_UTILITY.GET_HIRE_DATE

RETURN DATA TYPE: Date

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ID |
| 2 | DATE_EARNED |

PARAMETERS: None

## GET_MIN_WAGE

This function determines the Minimum Wage for a given Economic Zone.

SOURCE: PAY_MX_UTILITY.GET_MIN_WAGE

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | DATE_EARNED |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_TAX_BASIS | I | Not used | NONE |

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| | P_ECON_ZONE | I | Economic Zone of GRE | NONE<br>A<br>B |

## GET_MX_ECON_ZONE

This function determines the Economic Zone setup for a given GRE.

SOURCE: PAY_MX_UTILITY.GET_MX_ECON_ZONE

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | TAX_UNIT_ID |
| 2 | DATE_EARNED |

PARAMETERS: None

## GET_MX_EE_HEAD_COUNT

This function returns the Employee Headcount setup for a Legal Employer and Jurisdiction combination.

SOURCE: PAY_MX_TAX_FUNCTIONS.GET_MX_EE_HEAD_COUNT

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |

| SEQUENCE | CONTEXT |
|---|---|
| 3 | DATE_EARNED |
| 4 | JURISDICTION_CODE |

PARAMETERS: None

## GET_MX_STATE_TAX_RULES

This function returns the State Tax Rate setup for a Legal Employer and Jurisdiction combination.

SOURCE: PAY_MX_TAX_FUNCTIONS.GET_MX_STATE_TAX_RULES

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |
| 3 | DATE_EARNED |
| 4 | JURISDICTION_CODE |

PARAMETERS: None

## GET_MX_TAX_INFO

This function determines the seeded Tax Information for a given GRE and Information Type.

SOURCE: PAY_MX_TAX_FUNCTIONS.GET_MX_TAX_INFO

Return Data Type: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |
| 3 | DATE_EARNED |
| 4 | JURISDICTION_CODE |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_LEGISLATION_INFO_TYPE | I | Information Type | 'MX Minimum Wage Information' |
| 2 | P_LEGISLATION_INFO1 | B | Economic Zone | GMW<br>MWA<br>MWB |
| 3 | P_LEGISLATION_INFO2 | O | Minimum Wage | |
| 4 | P_LEGISLATION_INFO3 | O | Not used | |
| 5 | P_LEGISLATION_INFO4 | O | Not used | |
| 6 | P_LEGISLATION_INFO5 | O | Not used | |
| 7 | P_LEGISLATION_INFO6 | O | Not used | |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_LEGISLATION_INFO_TYPE | I | Information Type | 'MX Social Security Information' |
| 2 | P_LEGISLATION_INFO1 | B | Social Security Insurance Code | FA<br>AQ<br>BC<br>DD<br>DC<br>WRI<br>PEN<br>RET<br>SEP<br>INF |
| 3 | P_LEGISLATION_INFO2 | O | Days Basis of Quotation | |
| 4 | P_LEGISLATION_INFO3 | O | Insurance CAP | |
| 5 | P_LEGISLATION_INFO4 | O | EE Quota percentage | |
| 6 | P_LEGISLATION_INFO5 | O | ER Quota percentage | |
| 7 | P_LEGISLATION_INFO6 | O | Display Sequence | |

## GET_PREVIOUS_PERIOD_BAL

This function determines the previous period wages and taxes.

SOURCE: PAY_MX_TAX_FUNCTIONS.GET_PREVIOUS_PERIOD_BAL

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ID |
| 2 | ASSIGNMENT_ACTION_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_MODE | I | Database Item name for which the value to be determined | |

## GET_PROCESS_PARAMETERS

This function determines the Legislative Parameter value for a specified Payroll Action.

SOURCE: PAY_MX_UTILITY.GET_PROCESS_PARAMETERS

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | PAYROLL_ACTION_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_PARAMETER_NAME | I | Legislative Parameter name for which the value to be determined | |

## GET_SENIORITY

This function returns the Seniority of an Employee as on the 'Date Earned' context, rounded to nearest integer.

- Fractions from 0 to 6 months, Seniority = 0
- Fractions from 6.1 to 12 months, Seniority = 1
- 2 years 3 months = 2 seniority years
- 2 years 6 months 1 day = 3 seniority years

SOURCE: HR_MX_UTILITY.GET_SENIORITY

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |
| 3 | PAYROLL_ID |
| 4 | PERSON_ID |
| 5 | DATE_EARNED |

PARAMETERS: None

## GET_SENIORITY_SOCIAL_SECURITY

This function returns the Seniority of an Employee as on 'Date Earned' context, rounded to Two/Five decimal points, based on the value set for Profile Option 'PAY: No of decimal places for Seniority (Two/Five)'.

SOURCE: HR_MX_UTILITY.GET_SENIORITY_SOCIAL_SECURITY

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | PERSON_ID |
| 2 | DATE_EARNED |

PARAMETERS: None

## GET_SUBJECT_EARNINGS

This function determines the portion of Earnings Subject to ISR and State Taxes.

SOURCE: PAY_MX_TAX_FUNCTIONS.GET_PARTIAL_SUBJ_EARNINGS

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | DATE_EARNED |
| 2 | ASSIGNMENT_ACTION_ID |
| 3 | BUSINESS_GROUP_ID |
| 4 | JURISDICTION_CODE |
| 5 | ELEMENT_TYPE_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_TAX_TYPE | I | Tax Type | ISR<br>STATE |
| 2 | P_EARNINGS_ AMT | I | Earnings Amount | |
| 3 | P_GROSS_EAR NINGS | I | Gross Earnings | |
| 4 | P_DAILY_SALA RY | I | Daily Salary | |
| 5 | P_CLASSIFICAT ION_NAME | I | Secondary Classification name of Earnings | |

## GET_SUBJECT_EARNINGS_ANN

This function determines the portion of Earnings Subject to ISR and State Taxes, considering YTD Earnings also, in case of Earnings that are paid Annually.

SOURCE: PAY_MX_TAX_FUNCTIONS.GET_PARTIAL_SUBJ_EARNINGS

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | DATE_EARNED |
| 2 | ASSIGNMENT_ACTION_ID |
| 3 | BUSINESS_GROUP_ID |
| 4 | JURISDICTION_CODE |

| SEQUENCE | CONTEXT |
|---|---|
| 5 | ELEMENT_TYPE_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_TAX_TYPE | I | Tax Type | ISR<br>STATE |
| 2 | P_EARNINGS_ AMT | I | Earnings Amount | |
| 3 | P_YTD_EARNI NGS_AMT | I | Year to Date Earnings Amount | |
| 4 | P_GROSS_EAR NINGS | I | Gross Earnings | |
| 5 | P_YTD_GROSS_ EARNINGS | I | Year to Date Gross Earnings | |
| 6 | P_DAILY_SALA RY | I | Daily Salary | |
| 7 | P_CLASSIFICAT ION_NAME | I | Secondary Classification name of Earnings | |

## GET_SUBJECT_EARNINGS_FOR_PERIOD

This function determines the portion of Earnings Subject to ISR and State Taxes, considering PTD Earnings also, in case of Earnings that are paid Periodically.

SOURCE: PAY_MX_TAX_FUNCTIONS.GET_SUBJ_EARNINGS_FOR_PERIOD

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | DATE_EARNED |
| 2 | ASSIGNMENT_ACTION_ID |
| 3 | BUSINESS_GROUP_ID |
| 4 | JURISDICTION_CODE |
| 5 | ELEMENT_TYPE_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_TAX_TYPE | I | Tax Type | ISR<br>STATE |
| 2 | P_EARNINGS_ AMT | I | Earnings Amount | |
| 3 | P_PTD_EARNIN GS_AMT | I | Period to Date Earnings Amount | |
| 4 | P_GROSS_EAR NINGS | I | Gross Earnings | |
| 5 | P_YTD_GROSS_ EARNINGS | I | Year to Date Gross Earnings | |
| 6 | P_DAILY_SALA RY | I | Daily Salary | |
| 7 | P_CLASSIFICAT ION_NAME | I | Secondary Classification name of Earnings | |

## GET_TAX_BALANCE

This function determines the balance value for given DBI.

SOURCE: PAY_MX_FF_UDFS.GET_TAX_BALANCE

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ACTION_ID |
| 2 | TAX_UNIT_ID |
| 3 | DATE_EARNED |
| 4 | PAYROLL_ACTION_ID |
| 5 | BUSINESS_GROUP_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_ENTITY_NAME | I | Database Item name for which the value to be determined | |

## GET_TAX_SUBSIDY_PERCENT

This function returns the Tax Subsidy Percentage value setup for a GRE.

SOURCE: HR_MX_UTILITY.GET_TAX_SUBSIDY_PERCENT

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |

PARAMETERS: None

## GET_UMA_VALUE

This function returns the UMA (Unidad de Medida y Actualización – Unit of Measure and Update) value as on 'Date Earned' context.

To comply with legislative requirements, UMA value is used in the calculation of Mexico ISR, Employer State Tax and Social Security Quota components (instead of the Minimum Wage value).

SOURCE: PAY_MX_UTILITY.GET_UMA_VALUE

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | DATE_EARNED |

PARAMETERS: None

## GET_WRIP

This function returns the Work Risk Insurance Percentage value setup for a GRE.

SOURCE: HR_MX_UTILITY.GET_WRIP

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |
| 2 | TAX_UNIT_ID |

PARAMETERS: None

## IS_ASG_EXEMPT_FROM_ISR

This function determines if the employee is exempt from ISR withholding and should therefore be excluded from the Annual Tax Adjustment process. The value is derived from the ISR Exempt input value of the Mexico Tax element. Returns Y if Assignment is set for Exempt from ISR Tax calculation.

SOURCE: PAY_MX_TAX_FUNCTIONS.IS_ASG_EXEMPT_FROM_ISR

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ID |
| 2 | DATE_EARNED |

PARAMETERS: None

## FUNCTION

This function determines if the employee should be excluded from the Annual Tax Adjustment process. The function derives this based on the value entered for the Exempt Tax Adjustment field. Returns Y if an Employee is set for Exempt from Annual Tax Adjustment.

SOURCE: PAY_MX_TAX_FUNCTIONS.FUNCTION

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | ASSIGNMENT_ID |
| 2 | DATE_EARNED |

PARAMETERS: None

## MX_ENTRY_IN_LOOKUP

This function returns Y, if entry lookup value is valid for a specified lookup.

SOURCE: HR_MX_UTILITY.CHK_ENTRY_IN_LOOKUP

RETURN DATA TYPE: Text

CONTEXTS USED (which need not be passed explicitly): None

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_LOOKUP_TYPE | I | Lookup Type | |
| 2 | P_ENTRY_VAL | I | Lookup Value | |
| 3 | P_EFFECTIVE_DATE | I | Effective Date | |
| 4 | P_MESSAGE | O | Error Message | |

## STANDARD_HOURS_WORKED

This function returns the standard hours worked.

SOURCE: PAY_MX_FF_UDFS.STANDARD_HOURS_WORKED

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly): None

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_STD_HOURS | I | Total Standard Hours | |
| 2 | P_RANGE_START | I | Start Date | |
| 3 | P_RANGE_END | I | End Date | |

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 4 | P_STD_FREQ | I | Standard Frequency | |

### WORK_SCH_TOTAL_HOURS_OR_DAYS

This function returns the Work Schedule Total Hours or Days.

SOURCE: PAY_MX_FF_UDFS.WORK_SCH_TOTAL_HOURS_OR_DAYS

RETURN DATA TYPE: Number

CONTEXTS USED (which need not be passed explicitly):

| SEQUENCE | CONTEXT |
|---|---|
| 1 | BUSINESS_GROUP_ID |

***PARAMETERS***

| SEQ | NAME | TYPE (IN/OUT /BOTH) | DESCRIPTION | VALID VALUES |
|---|---|---|---|---|
| 1 | P_WS_NAME | I | Work Schedule name | |
| 2 | P_RANGE_START | I | Start Date | |
| 3 | P_RANGE_END | I | End Date | |
| 4 | P_MODE | I | Mode | Hours<br>Days |

## South African Legislative Functions

Oracle HRMS for South Africa provides two prorating functions for use in formulas:

### Prorate_Working_Days

- Proration is based on an average of 21.67 working days in a month. The number of days the employee has worked is derived from Assignment start to end date or the Payroll Period End Date. The number of working days in a month excludes Saturdays and Sundays but includes public holidays.

### Prorate_Calendar_Days

- Proration is based on the number of calendar days in a month. The number of days the employee has worked is derived from Assignment start to end date or the Payroll Period End Date.

See:Functions, *Using Oracle FastFormula*

## UK Only Functions

The following functions have been registered for use in the UK only formulas.

### CALCULATE_TIME_WORKED

Calculates the time worked between a start date and end date for an assignment.

### CLASS1A_YTD

Calculate car benefit, year to date

### COUNT_ASSIGNMENTS

Count of assignments an employee has

### DIRECTOR_WEEKS

Number of weeks an employee has been a director

### GET_BACS_PROCESS_DATE

Return the BACS processing date

### GET_FTE_VALUE

The GET_FTE_VALUE identifies the FTE value for a given assignment on a specified date in the past from the PER_ASSIGNMENT_BUDGET_VALUES table.

This function uses the following Input Parameters:

- P_ASSIGNMENT_ID (number) - this is a required input and identifies which assignment to retrieve FTE for.
- P_QUERY_DATE (date) - this is a required input and specifies the date at which to retrieve FTE.

### NI_ABLE_DIR_YTD

NIable pay for a director, year to date

### NI_ABLE_PER_PTD

NIable pay for a person with multiple assignments

### NI_CO_RATE_FROM_CI_RATE

Find the NI contracted out rate from the CI rate

### PAYMENT_YTD

Calculate car payment, year to date

### PERIOD_TYPE_CHECK

Test whether the period type is valid

### PQP_GB_GET_ABSENCE_SSP_FOR_DATE_RANGE

Returns the amount of SSP payable for an absence within a date range.

### PQP_GB_GET_ABSENCE_SMP_FOR_DATE_RANGE

Returns the amount of SMP payable for an absence within a date range.

### PQP_GB_GAP_GET_FIRST_PAID_DAY

Returns the date on which a person starts to receive a given level of OSP payment.

### PQP_GB_GAP_GET_LAST_PAID_DAY

Returns the date on which a person last received a given level of OSP payment.

### PQP_GB_GAP_GET_FIRST_ENTITLED_DAY

Returns the date on which a person's entitlement for a given level of OSP payment

starts.

### PQP_GB_GAP_GET_LAST_ENTITLED_DAY

Returns the date on which a person's entitlement for a given level of OSP payment ceases.

### SESSION_DATE

Return the session date

### UK_TAX_YR_END

Find the end of the tax year

### UK_TAX_YR_START

Find the start of the tax year

### USER_RANGE_BY_LABEL

Lower bound of range from user table using row label

### USER_VALUE_BY_LABEL

Value from user table using row label

### VALIDATE_BACS_DATE

Return the previous BACS process date to a given date

### VALIDATE_USER_VALUE

Verify that a given value is in a user table.

# Glossary

### 360-Degree Appraisal

Part of the Performance Management Appraisal function and also known as a Group Appraisal. This is an employee appraisal undertaken by managers with participation by reviewers.

### Absence

A period of time in which an employee performs no work for the assigned organization.

### Absence Case

Two or more absences for the same person that you associate manually because they share a common factor, such as the same underlying cause.

### Absence Type

Category of absence, such as medical leave or vacation leave, that you define for use in absence windows.

### Accrual

The recognized amount of leave credited to an employee which is accumulated for a particular period.

### Accrual Band

A range of values that determines how much paid time off an employee accrues. The values may be years of service, grades, hours worked, or any other factor.

### Accrual Period

The unit of time, within an accrual term, in which PTO is accrued. In many plans, the same amount of time is accrued in each accrual period, such as two days per month. In other plans, the amount accrued varies from period to period, or the entitlement for the full accrual term is given as an up front amount at the beginning of the accrual term.

### Accrual Plan

See: *PTO Accrual Plan*, page Glossary-31

### Accrual Term

The period, such as one year, for which accruals are calculated. In most accrual plans, unused PTO accruals must be carried over or lost at the end of the accrual term. Other plans have a rolling accrual term which is of a certain duration but has no fixed start and end dates.

### Action

In AME, an Action is the *Then* part of an Approval Rule that specifies how the application must progress a transaction's approval process in a particular way depending on the conditions met.

### Action Type

In AME, an action type is the generic container for specific actions. It enables you to specify the action to take if a transaction meets the condition of an approval rule. The action type, thus, generates the appropriate approvers for a transaction. As an AME administrator you can make particular action types available for specified transaction types.

### Activity Rate

The monetary amount or percentage associated with an activity, such as $12.35 per pay period as an employee payroll contribution for medical coverage. Activity rates can apply to participation, eligibility, coverages, contributions, and distributions.

### Actual Premium

The per-participant premium an insurance carrier charges the plan sponsor for a given benefit.

### Administrative Enrollment

A type of scheduled enrollment caused by a change in plan terms or conditions and resulting in a re-enrollment.

### AdvancePay

A process that recalculates the amount to pay an employee in the current period, to make an authorized early payment of amounts that would normally be paid in future payroll periods.

### Agency

An external organization that assists an enterprise in their recruitment process. Agencies act on behalf of the candidates to help them search and apply for jobs. They provide candidates to the fill up job openings in an enterprise or sometimes handle the complete placement process for a vacancy.

### Agency Candidate

An agency candidate is a person whose profile is created in iRecruitment by a recruiting agency. This profile includes personal and professional information.

### Agency User

An external person who belongs to a recruiting agency and accesses iRecruitment to conduct recruiting activities such as creating candidates and applying on behalf of the candidates.

### Alert

An email notification that you can set up and define to send a recipient or group of recipients a reminder or warning to perform a certain task or simply a notification to inform the recipient of any important information.

### Align

To define a relationship between objectives. Workers can align their own objectives with objectives that other workers have shared with them. Aligned objectives are also known as *supporting objectives*.

### AME

Oracle Approvals Management Engine. A highly extensible approvals rules engine that enables organizations implementing Oracle Applications to simply and effectively define business rules that determine who must approve a transaction originating within an application. You can devise simple or complex rules, as your organization requires, which then form part of your overall business flow. A central repository holds all the rules to facilitate management and sharing between business processes.

### API

Application Programmatic Interfaces, used to upload data to the Oracle Applications database. APIs handle error checking and ensure that invalid data is not uploaded to the database.

### Applicant

An applicant is a person who submits an application for employment to an organization.

### Applicability

In HRMS budgeting, a term describing whether a budget reallocation rule pertains to donors or receivers.

### Applicant/Candidate Matching Criteria

Matching functionality in the iRecruitment system that systematically identifies which

candidates and applicants possess the skills, knowledge and abilities to be considered for a specific vacancy. The following columns are used for matching:

- Skills
- FT/PT
- Contractor/Employee
- Work at Home
- Job Category
- Distance to Location
- Key Words
- Salary

### Apply for a Job

An SSHR function that enables an employee to, apply, search and prepare applications for an internally advertised vacancy.

### Appraisal

An appraisal is a process where an employee's work performance is rated and future objectives set.

See also: *Assessment*, page Glossary-5.

### Appraisee

The person who is the subject of an appraisal.

### Appraiser

A person, usually a manager, who appraises an employee.

### Appraising Manager

The person who initiates and performs an Employee-Manager or 360 Degree Appraisal. An appraising manager can create appraisal objectives.

### Approval Rule

In AME, a business rule that determines a transaction's approval process. You construct rules using *conditions* and *actions*. For example, you can write a business rule with the conditions that if the total cost of a transaction is less than 1000 USD, and the transaction is for travel expenses, then the action must be to obtain approval from the immediate supervisor of the person triggering the transaction.

### Approver Groups

In AME, an approver group is a collection of approvers you define, which you can include as part of actions when you set up your approval rules.

### Arrestment

Scottish court order made out for unpaid debts or maintenance payments.

See also: *Court Order* , page Glossary-12

### Assessment

An information gathering exercise, from one or many sources, to evaluate a person's ability to do a job.

See also: *Appraisal*, page Glossary-4.

### Assignment

A worker's assignment identifies their role within a business group. The assignment is made up of a number of assignment components. Of these, organization is mandatory, and payroll is required (for employees only) for payment purposes.

### Assignment Number

A number that uniquely identifies a worker's assignment. A worker with multiple assignments has multiple assignment numbers.

### Assignment Rate

A monetary value paid to a contingent worker for a specified period of time. For example, an assignment rate could be an hourly overtime rate of $10.50.

### Assignment Set

A grouping of employees and applicants that you define for running QuickPaint reports and processing payrolls.

See also: *QuickPaint Report*, page Glossary-32

### Assignment Status

For workers, used to track their permanent or temporary departures from your enterprise and, for employees only, to control the remuneration they receive. For applicants, used to track the progress of their applications.

### Attribute

In AME, attributes are the business facts of a transaction, such as the total amount of a transaction, percentage of a discount, an item's category, or a person's salary and so on. These business variables form part of the conditions of an approval rule, and determine how the transaction must progress for approvals.

### Authoria

A provider of health insurance and compensation information, that provides additional information about benefits choices.

### BACS

Banks Automated Clearing System. This is the UK system for making direct deposit payments to employees.

### Balance Adjustment

A correction you make to a balance. You can adjust user balances and assignment level predefined balances only.

### Balance Dimension

The period for which a balance sums its balance feeds, or the set of assignments/transactions for which it sums them. There are five time dimensions: Run, Period, Quarter, Year and User. You can choose any reset point for user balances.

### Balance Feeds

These are the input values of matching units of measure of any elements defined to feed the balance.

### Balances

Positive or negative accumulations of values over periods of time normally generated by payroll runs. A balance can sum pay values, time periods or numbers.

See also: *Predefined Components* , page Glossary-30

### Bargaining Unit

A bargaining unit is a legally organized group of people which have the right to negotiate on all aspects of terms and conditions with employers or employer federations. A bargaining unit is generally a trade union or a branch of a trade union.

### Base Summary

A database table that holds the lowest level of summary. Summary tables are populated and maintained by user-written concurrent programs.

### Beneficiary

A person or organization designated to receive the benefits from a benefit plan upon the death of the insured.

### Benefit

Any part of an employee's remuneration package that is not pay. Vacation time,

employer-paid medical insurance and stock options are all examples of benefits.

See also: *Elements*, page Glossary-16

### Block

The largest subordinate unit of a window, containing information for a specific business function or entity. Every window consists of at least one block. Blocks contain fields and, optionally, regions. They are delineated by a bevelled edge. You must save your entries in one block before navigating to the next.

See also: *Region*, page Glossary-33, *Field*, page Glossary-18

### Budget Measurement Type (BMT)

A subset of Workforce Measurement Type. It consists of a number of different units used to measure the workforce. The most common units are headcount and full time equivalent.

### Budget Value

In Oracle Human Resources you can enter staffing budget values and actual values for each assignment to measure variances between actual and planned staffing levels in an organization or hierarchy.

### Business Group

The business group represents a country in which your enterprise operates. It enables you to group and manage data in accordance with the rules and reporting requirements of each country, and to control access to data.

### Business Group Currency

The currency in which Oracle Payroll performs all payroll calculations for your Business Group. If you pay employees in different currencies to this, Oracle Payroll calculates the amounts based on exchange rates defined in the system.

### Business Number (BN)

In Canada, this is the employer's account number with Revenue Canada. Consisting of 15 digits, the first 9 identify the employer, the next 2 identify the type of tax account involved (payroll vs. corporate tax), and the last 4 identify the particular account for that tax.

### Business Rule

See Configurable Business Rules, page Glossary-10

### Cafeteria Benefits Plan

See: *Flexible Benefits Program*, page Glossary-18

### Calendar Exceptions

If you are using the Statutory Absence Payments (UK) feature, you define calendar exceptions for an SSP qualifying pattern, to override the pattern on given days. Each calendar exception is another pattern which overrides the usual pattern.

### Calendars

In Oracle Human Resources you define calendars that determine the start and end dates for budgetary years, quarters and periods. For each calendar you select a basic period type. If you are using the Statutory Absence Payments (UK) feature, you define calendars to determine the start date and time for SSP qualifying patterns.

### Canada/Quebec Pension Plan (CPP/QPP) Contributions

Contributions paid by employers and employees to each of these plans provide income benefits upon retirement.

### Candidate

(iRecruitment) A candidate is a person who has either directly provided their personal and professional information to a company's job site or provided their resume and details to a manager or recruiter for entering in the iRecruitment system.

### Candidate Offers

An SSHR function used by a line manager to offer a job to a candidate. This function is supplied with its own responsibility.

### Career Path

This shows a possible progression from one job or position from any number of other jobs or positions within the Business Group. A career path must be based on either job progression or position progression; you cannot mix the two.

### Carry Over

The amount of unused paid time off entitlement an employee brings forward from one accrual term to the next. It may be subject to an expiry date i.e. a date by which it must be used or lost.

See also: *Residual*, page Glossary-34

### Cascade

A process managers at each level in a hierarchy use to allocate their own objectives to workers who report directly to them. This technique enables the allocation of enterprise objectives in some form to all workers.

### Cash Analysis

A specification of the different currency denominations required for paying your

employees in cash. Union contracts may require you to follow certain cash analysis rules.

### Ceiling

The maximum amount of unused paid time off an employee can have in an accrual plan. When an employee reaches this maximum, he or she must use some accrued time before any more time will accrue.

### Certification

Documentation required to enroll or change elections in a benefits plan as the result of a life event, to waive participation in a plan, to designate dependents for coverage, or to receive reimbursement for goods or services under an FSA.

### Child/Family Support Payments

In Canada, these are payments withheld from an employee's compensation to satisfy a child or family support order from a Provincial Court. The employer is responsible for withholding and remitting the payments to the court named in the order.

### Collective Agreement

A collective agreement is a form of contract between an employer or employer representative, for example, an employer federation, and a bargaining unit for example, a union or a union branch.

### Collective Agreement Grade

Combination of information that allows you to determine how an employee is ranked or graded in a collective agreement.

### Communications

Benefits plan information that is presented in some form to participants. Examples include a pre-enrollment package, an enrollment confirmation statement, or a notice of default enrollment.

### Compensation

The pay you give to employees, including wages or salary, and bonuses.

See also: *Elements*, page Glossary-16

### Compensation Category

A group of compensation items. Compensation Categories determine the type of compensation that you award under a plan.

### Compensation Object

For Standard and Advanced Benefits, compensation objects define, categorize, and help to manage the benefit plans that are offered to eligible participants. Compensation

objects include programs, plan types, plans, options, and combinations of these entities.

### Competency

Any measurable behavior required by an organization, job or position that a person may demonstrate in the work context. A competency can be a piece of knowledge, a skill, an attitude, or an attribute.

See also: *Unit Standard Competency*, page Glossary-40

### Competency Assessment Template

The entity that configures the Competencies section of an appraisal.

See also: *Objective Assessment Template*, page Glossary-25

### Competency Evaluation

A method used to measure an employees ability to do a defined job.

### Competency Profile

Where you record applicant and employee accomplishments, for example, proficiency in a competency.

### Competency Requirements

Competencies required by an organization, job or position.

See also: *Competency*, page Glossary-10, *Core Competencies*, page Glossary-11

### Competency Type

A group of related competencies.

### Condition

In AME, a Condition is the *If* part of an Approval Rule that specifies the conditions a transaction must meet to trigger an approval action. A condition consists of an attribute, which is a business variable, and a set of attribute values that you can define. When a transaction meets the specified attribute values, then the application triggers the appropriate action.

### Configurable Business Rule

In HRMS position control and budgeting, predefined routines (also called process rules) that run when you apply an online transaction, and validate proposed changes to positions, budgets, or assignments. You set their default status level (typically Warning) to Warning, Ignore, or Error.

### Configurable Forms

Forms that your system administrator can modify for ease of use or security purposes by means of Custom Form restrictions. The Form Customization window lists the forms and their methods of configuration.

### Consideration

(iRecruitment) Consideration means that a decision is registered about a person in relation to a vacancy so that the person can be contacted.

### Consolidation Set

A grouping of payroll runs within the same time period for which you can schedule reporting, costing, and post-run processing.

### Contact

A person who has a relationship to an employee that you want to record. Contacts can be dependents, relatives, partners or persons to contact in an emergency.

### Content

When you create a spreadsheet or word processing document using Web ADI, the content identifies the data in the document. Content is usually downloaded from the Oracle application database.

### Contingent Worker

A worker who does not have a direct employment relationship with an enterprise and is typically a self-employed individual or an agency-supplied worker. The contingent worker is not paid via Oracle Payroll.

### Contract

A contract of employment is an agreement between an employer and employee or potential employee that defines the fundamental legal relationship between an employing organization and a person who offers his or her services for hire. The employment contract defines the terms and conditions to which both parties agree and those that are covered by local laws.

### Contribution

An employer's or employee's monetary or other contribution to a benefits plan.

### Core Competencies

Also known as *Leadership Competencies* or *Management Competencies*. The competencies required by every person to enable the enterprise to meet its goals.

See also: *Competency*, page Glossary-10

### Costable Type

A feature that determines the processing an element receives for accounting and costing purposes. There are four costable types in Oracle HRMS: costed, distributed costing, fixed costing, and not costed.

### Costing

Recording the costs of an assignment for accounting or reporting purposes. Using Oracle Payroll, you can calculate and transfer costing information to your general ledger and into systems for project management or labor distribution.

### Court Order

A ruling from a court that requires an employer to make deductions from an employee's salary for maintenance payments or debts, and to pay the sums deducted to a court or local authority.

See also: *Arrestment*, page Glossary-5

### Credit

A part of the Qualifications Framework. The value a national qualifications authority assigns to a unit standard competence or a qualification. For example, one credit may represent 10 hours of study, a unit standard competence may equate to 5 credits, and a qualification may equate to 30 credits.

### Criteria Salary Rate

Variable rate of pay for a grade, or grade step. Used by Grade/Step Progression.

### Current Period of Service

An employee's period of service is current if their most recent hire date is on or before the effective date, and either the employee does not have a termination date for their latest employment, or their termination date is later than the effective date.

The table below provides an example using an effective date of 12 October 2004:

| Effective Date | Hire Date | Termination Date | Current Period of Service? |
|---|---|---|---|
| 12 Oct 2004 | 23 Jan 1994 | 16 Aug 2003 | No |
| 12 Oct 2004 | 14 Oct 2004 | ANY | No |
| 12 Oct 2004 | 14 Mar 2000 | NONE | Yes |
| 12 Oct 2004 | 11 Sep 2001 | 15 Oct 2004 | Yes |

**Additional Information:** In Oracle HRMS an employee cannot transfer from one business group to another. To move from one business group to another, the business group they are leaving must terminate the

employee, and the business group they are joining must re-hire the employee. Therefore the definition of period of service, above, does not take account of any service prior to the most recent business group transfer.

### Current Period of Placement

A contingent worker's period of placement, page Glossary-28 is current if their most recent placement start date is on or before the effective date, and either the contingent worker does not have a placement end date for their latest placement or their placement end date is later than the effective date.

| Effective Date | Place Date | End Placement Date | Current Period of Placement? |
|---|---|---|---|
| 12 Oct 2004 | 23 Jan 1994 | 16 Aug 2003 | No |
| 12 Oct 2004 | 14 Oct 2004 | ANY | No |
| 12 Oct 2004 | 14 Mar 2000 | NONE | Yes |
| 12 Oct 2004 | 11 Sep 2001 | 15 Oct 2004 | Yes |

### Database Item

An item of information in Oracle HRMS that has special programming attached, enabling Oracle FastFormula to locate and retrieve it for use in formulas.

### Date Earned

The date the payroll run uses to determine which element entries to process. In North America (and typically elsewhere too) it is the last day of the payroll period being processed.

### Date Paid

The effective date of a payroll run. Date paid dictates which tax rules apply and which tax period or tax year deductions are reported.

### Date To and Date From

These fields are used in windows not subject to DateTrack. The period you enter in these fields remains fixed until you change the values in either field.

See also: *DateTrack*, page Glossary-13, *Effective Date*, page Glossary-15

### DateTrack

When you change your effective date (either to past or future), DateTrack enables you to enter information that takes effect on your new effective date, and to review information as of the new date.

See also: *Effective Date*, page Glossary-15

### Default Postings

(iRecruitment) Default text stored against business groups, organizations, jobs, and/or positions. The default postings are used to create job postings for a vacancy.

### Dependent

In a benefit plan, a person with a proven relationship to the primary participant whom the participant designates to receive coverage based on the terms of the plan.

### Deployment

The temporary or permanent employment of an employee in a business group.

See also: *Secondment*, page Glossary-36

### Deployment Factors

See: *Work Choices*, page Glossary-42

### Deployment Proposal

The entity that controls the permanent transfer or temporary secondment of an employee from a source business group to a destination business group. The HR Professional in the destination business group creates the deployment proposal using the Global Deployments function.

### Derived Factor

A factor (such as age, percent of fulltime employment, length of service, compensation level, or the number of hours worked per period) that is used in calculations to determine Participation Eligibility or Activity Rates for one or more benefits.

### Descriptive Flexfield

A field that your organization can configure to capture additional information required by your business but not otherwise tracked by Oracle Applications.

See also: *Key Flexfield* , page Glossary-22

### Deviation

A change to the standard approver list is a deviation.

### Developer Descriptive Flexfield

A flexfield defined by your localization team to meet the specific legislative and

reporting needs of your country.

See also: *Extra Information Types*, page Glossary-18

### Direct Deposit

The electronic transfer of an employee's net pay directly into the account(s) designated by the employee.

### Discoverer Workbook

A grouping of worksheets. Each worksheet is one report.

> **Additional Information:** See also My Oracle Support Knowledge Document 2277369.1, *Oracle E-Business Suite Support Implications for Discoverer 11gR1*.

### Discoverer Worksheet

A single report within a workbook. A report displays the values of predefined criteria for analysis.

### Distribution

Monetary payments made from, or hours off from work as allowed by, a compensation or benefits plan.

### Download

The process of transferring data from the Oracle HRMS application to your desktop (the original data remains in the application database).

### Effective Date

The date for which you are entering and viewing information. You set your effective date in the Alter Effective Date window.

See also: *DateTrack*, page Glossary-13

### EIT

See: *Extra Information Type*, page Glossary-18

### Electability

The process which determines whether a potential benefits participant, who has satisfied the eligibility rules governing a program, plan, or option in a plan, is able to elect benefits. Participants who are *eligible* for benefits do not always have *electable* benefit choices based on the rules established in a benefit plan design.

### Element Classifications

These control the order in which elements are processed and the balances they feed.

Primary element classifications and some secondary classifications are predefined by Oracle Payroll. Other secondary classifications can be created by users.

### Element Entry

The record controlling an employee's receipt of an element, including the period of time for which the employee receives the element and its value.

See also: *Recurring Elements*, page Glossary-33, *Nonrecurring Elements*, page Glossary-25

### Element Link

The association of an element to one or more components of an employee assignment. The link establishes employee eligibility for that element. Employees whose assignment components match the components of the link are eligible for the element.

See also: *Standard Link*, page Glossary-38

### Elements

Components in the calculation of employee pay. Each element represents a compensation or benefit type, such as salary, wages, stock purchase plans, and pension contributions.

### Element Set

A group of elements that you define to process in a payroll run, or to control access to compensation information from a configured form, or for distributing costs.

### Eligibility

The process by which a potential benefits participant satisfies the rules governing whether a person can ever enroll in a program, plan, or option in a plan. A participant who is *eligible* for benefits must also satisfy *electability* requirements.

### Eligibility Profile

A set of eligibility criteria grouped together. Eligibility profiles help determine eligibility for compensation and benefits and are re-usable. Eligibility profiles can be linked to a compensation object (such as a program, plan, or option), a collective agreement, a grade ladder, or a work schedule to restrict eligibility for these.

### Employee

A worker who has a direct employment relationship with the employer. Employees are typically paid compensation and benefits via the employer's payroll application.

Employees have a system person type of Employee and one or more assignments with an assignment type of Employee.

### Employee Histories

An SSHR function for an employee to view their Learning History, Job Application

History, Employment History, Absence History, or Salary History. A manager can also use this function to view information on their direct reports.

### Employment Category

A component of the employee assignment. Four categories are defined: Full Time - Regular, Full Time - Temporary, Part Time - Regular, and Part Time - Temporary.

### Employment Equity Occupational Groups (EEOG)

In Canada, the Employment Equity Occupational Groups (EEOG) consist of 14 classifications of work used in the Employment Equity Report. The EEOGs were derived from the National Occupational Classification system.

### Employment Insurance (EI)

Benefit plan run by the federal government to which the majority of Canadian employers and employees must contribute.

### Employment Insurance Rate

In Canada, this is the rate at which the employer contributes to the EI fund. The rate is expressed as a percentage of the employee's contribution. If the employer maintains an approved wage loss replacement program, they can reduce their share of EI premiums by obtaining a reduced contribution rate. Employers would remit payroll deductions under a different employer account number for employees covered by the plan.

### Enrollment Action Type

Any action required to complete enrollment or de-enrollment in a benefit.

### Entitlement

In Australia, this is all unused leave from the previous year that remains to the credit of the employee.

### ESS

Employee Self Service. A predefined SSHR responsibility.

### Event

An activity such as a training day, review, or meeting, for employees or applicants. Known as *class* in OLM.

### Ex-Applicant

Someone who has previously applied for a vacancy or multiple vacancies, but all applications have ended, either because the applicant has withdrawn interest or they have been rejected. Ex-Applicants can still be registered users.

### Expected Week of Childbirth (EWC)

In the UK, this is the week in which an employee's baby is due. The Sunday of the expected week of childbirth is used in the calculations for Statutory Maternity Pay (SMP).

### Extra Information Type (EIT)

A type of developer descriptive flexfield that enables you to create an unlimited number of information types for six key areas in Oracle HRMS. Localization teams may also predefine some EITs to meet the specific legislative requirements of your country.

See also: *Developer Descriptive Flexfield*, page Glossary-14

### Field

A view or entry area in a window where you enter, view, update, or delete information.

See also: *Block*, page Glossary-7, *Region*, page Glossary-33

### Flex Credit

A unit of "purchasing power" in a flexible benefits program. An employee uses flex credits, typically expressed in monetary terms, to "purchase" benefits plans and/or levels of coverage within these plans.

### Flexible Benefits Program

A benefits program that offers employees choices among benefits plans and/or levels of coverage. Typically, employees are given a certain amount of flex credits or moneys with which to "purchase" these benefits plans and/or coverage levels.

### Flexible Spending Account

(FSA) Under US Internal Revenue Code Section 125, employees can set aside money on a pretax basis to pay for eligible unreimbursed health and dependent care expenses. Annual monetary limits and use-it-or-lose it provisions exist. Accounts are subject to annual maximums and forfeiture rules.

### Form

A predefined grouping of functions, called from a menu and displayed, if necessary, on several windows. Forms have blocks, regions and fields as their components.

See also: *Block*, page Glossary-7, *Region*, page Glossary-33, *Field*, page Glossary-18

### Format Mask

A definition of a person-name format. The format mask comprises standard name components, such as title, first name, and last name, in an order appropriate to its purpose and legislation.

### Format Type

A format-mask classification that identifies the mask's purpose. Oracle HRMS defines the Full Name, Display Name, List Name, and Order Name format types. You can also define your own format types for use in custom code.

### Full Time Equivalent (FTE)

A Workforce Measurement Type (WMT) that measures full time equivalent. Although the actual value and calculation may vary, this value is taken from the Assignment Budget Value (ABV) in Oracle HRMS. If the Assignment Budget Value in Oracle HRMS is not set up then a FastFormula is used to determine the value to be calculated.

### Global Value

A value you define for any formula to use. Global values can be dates, numbers or text.

### Goods or Service Type

A list of goods or services a benefit plan sponsor has approved for reimbursement.

### Grade

A component of an employee's assignment that defines their level and can be used to control the value of their salary and other compensation elements.

### Grade Comparatio

A comparison of the amount of compensation an employee receives with the mid-point of the valid values defined for his or her grade.

### Grade Ladder

The key component of Grade/Step Progression. You use a grade ladder to categorize grades, to determine the rules for how an employee progresses from one grade (or step) to the next, and to record the salary rates associated with each grade or step on the ladder.

### Grade Rate

A value or range of values defined as valid for a given grade. Used for validating employee compensation entries.

### Grade Scale

A sequence of steps valid for a grade, where each step corresponds to one point on a pay scale. You can place each employee on a point of their grade scale and automatically increment all placements each year, or as required.

See also: *Pay Scale*, page Glossary-27

### Grade Step

An increment on a grade scale. Each grade step corresponds to one point on a pay scale.

See also: *Grade Scale*, page Glossary-19

### Grandfathered

A term used in Benefits Administration. A person's benefits are said to be grandfathered when a plan changes but they retain the benefits accrued.

### Group

A component that you define, using the People Group key flexfield, to assign employees to special groups such as pension plans or unions. You can use groups to determine employees' eligibility for certain elements, and to regulate access to payrolls.

### Group Certificate

In Australia, this is a statement from a legal employer showing employment income of an employee for the financial year..

### Headcount(HEAD)

A Workforce Measurement Type (WMT) that measures headcount. Although the actual value and calculation may vary, this value is taken from the Assignment Budget Value (ABV) in Oracle HRMS. If the Assignment Budget Value in Oracle HRMS is not set up then a FastFormula is used to determine the value to be calculated.

### Hierarchy

An organization or position structure showing reporting lines or other relationships. You can use hierarchies for reporting and for controlling access to Oracle HRMS information.

### High Availability

iRecruitment functionality that enables enterprises to switch between two instances to continuously support the candidate job site.

### Imputed Income

Certain forms of indirect compensation that US Internal Revenue Service Section 79 defines as fringe benefits and taxes the recipient accordingly. Examples include employer payment of group term life insurance premiums over a certain monetary amount, personal use of a company car, and other non-cash awards.

### Incumbent

In Oracle HRMS, the term Incumbent refers to an active worker (employee or

contingent worker).

### Individual Compensation Distribution

A tool that enables managers assign one-time or recurring awards, bonuses, and allowances to qualified employees such as housing allowances, spot bonuses, and company cars. Also enables employees to enter voluntary contributions, such as savings plans, charitable organizations, and company perquisites.

### Info Online

A generic framework to integrate Oracle applications with partner applications, enabling users to access information from third-party providers, My Oracle Support and Oracle Learning Management.

### Initiator

A person who starts a 360 Degree appraisal (Employee or Self) on an individual. An initiator and the appraisee are the only people who can see all appraisal information.

### Input Values

Values you define to hold information about elements. In Oracle Payroll, input values are processed by formulas to calculate the element's run result. You can define up to fifteen input values for an element.

### Instructions

An SSHR user assistance component displayed on a web page to describe page functionality.

### Integrating Application

In AME, an application that uses Oracle Approvals Management Engine to manage the approval processes of its transactions.

### Integrator

Defines all the information that you need to download or upload from a particular window or database view using Web ADI.

### Interface

A Web ADI term for the item that specifies the columns to be transferred from the Oracle applications database to your desktop or vice versa.

### Involuntary

Used in turnover to describe employees who have ceased employment with the enterprise not of their own accord, for example, through redundancy.

### Job

A job is a generic role within a business group, which is independent of any single organization. For example, the jobs "Manager" and "Consultant" can occur in many organizations.

### Job Posting

An advertisement for a specific vacancy. This is the public side of the vacancy for which a candidate would apply.

### Key Flexfield

A flexible data field made up of segments. Each segment has a name you define and a set of valid values you specify. Used as the key to uniquely identify an entity, such as jobs, positions, grades, cost codes, and employee groups.

See also: *Descriptive Flexfield*, page Glossary-14

### Layout

Indicates the columns to be displayed in a spreadsheet or Word document created using Web ADI.

### Learning Management

Oracle's enterprise learning management system that administers online and offline educational content.

### Leave Loading

In Australia, an additional percentage amount of the annual leave paid that is paid to the employee.

### Leaver's Statement

In the UK, this Records details of Statutory Sick Pay (SSP) paid during a previous employment (issued as form SSP1L) which is used to calculate a new employee's entitlement to SSP. If a new employee falls sick, and the last date that SSP was paid for under the previous employment is less than eight calendar weeks before the first day of the PIW for the current sickness, the maximum liability for SSP is reduced by the number of weeks of SSP shown on the statement.

### Legal Employer

A business in Australia that employs people and has registered with the Australian Tax Office as a Group Employer.

### Legal Entity

A legal entity represents the designated legal employer for all employment-related

activities. The legal authorities in a country recognize this organization as a separate employer.

### Life Event

A significant change in a person's life that results in a change in eligibility or ineligibility for a benefit.

### Life Event Collision

A situation in which the impacts from multiple life events on participation eligibility, enrollability, level of coverage or activity rates conflict with each other.

### Life Event Enrollment

A benefits plan enrollment that is prompted by a life event occurring at any time during the plan year.

### Linked PIWs

In the UK, these are linked periods of incapacity for work that are treated as one to calculate an employee's entitlement to Statutory Sick Pay (SSP). A period of incapacity for work (PIW) links to an earlier PIW if it is separated by less than the linking interval. A linked PIW can be up to three years long.

### Linking Interval

In the UK, this is the number of days that separate two periods of incapacity for work. If a period of incapacity for work (PIW) is separated from a previous PIW by less than the linking interval, they are treated as one PIW according to the legislation for entitlement to Statutory Sick Pay (SSP). An employee can only receive SSP for the maximum number of weeks defined in the legislation for one PIW.

### LMSS

Line Manager Self Service. A predefined SSHR responsibility.

### Long Service Leave

Leave with pay granted to employees of a particular employer after a prescribed period of service or employment with that employer.

### Lookup Types

Categories of information, such as nationality, address type and tax type, that have a limited list of valid values. You can define your own Lookup Types, and you can add values to some predefined Lookup Types.

### Lower Earnings Limit (LEL)

In the UK, this is the minimum average weekly amount an employee must earn to pay National Insurance contributions. Employees who do not earn enough to pay National

Insurance cannot receive Statutory Sick Pay (SSP) or Statutory Maternity Pay (SMP).

### Manager

(iRecruitment) A manager accesses the iRecruitment system to document their hiring needs and conduct their recruiting activities online. Specifically, these activities include vacancy definition, searching for candidates, and processing applicants through the vacancy process.

### Manager-Employee Appraisal

Part of the Appraisal function. A manager appraisal of an employee. However, an appraising manager does not have to be a manager.

### Mapping

If you are bringing in data from a text file to Oracle HRMS using a spreadsheet created in Web ADI, you need to map the columns in the text file to the application's tables and columns.

### Maternity Pay Period

In the UK, this is the period for which Statutory Maternity Pay (SMP) is paid. It may start at any time from the start of the 11th week before the expected week of confinement and can continue for up to 18 weeks. The start date is usually agreed with the employee, but can start at any time up to the birth. An employee is not eligible to SMP for any week in which she works or for any other reason for ineligibility, defined by the legislation for SMP.

### Medicare Levy

An amount payable by most taxpayers in Australia to cover some of the cost of the public health system.

### Menus

You set up your own navigation menus, to suit the needs of different users.

### My Account

(iRecruitment) My Account is the total of either a candidate or applicant's personal and vacancy-specific information including the information needed to manage their progress through the recruitment process.

### NACHA

National Automated Clearing House Association. This is the US system for making direct deposit payments to employees.

### National Identifier

This is the alphanumeric code that is used to uniquely identify a person within their country. It is often used for taxation purposes. For example, in the US it is the Social Security Number, in Italy it is the Fiscal Code, and in New Zealand it is the IRD Number.

### National Occupational Classification (NOC) code

In Canada, the National Occupational Classification (NOC) System was developed to best reflect the type of work performed by employees. Occupations are grouped in terms of particular tasks, duties and responsibilities. The use of this standardized system ensures consistency of data from year to year within the same company as well as between companies. These codes are used in the Employment Equity Report.

### Net Accrual Calculation

The rule that defines which element entries add to or subtract from a plan's accrual amount to give net entitlement.

### Net Entitlement

The amount of unused paid time off an employee has available in an accrual plan at any given point in time.

### Nonrecurring Elements

Elements that process for one payroll period only unless you make a new entry for an employee.

See also: *Recurring Elements*, page Glossary-33

### North American Industrial Classification (NAIC) code

The North American Industrial Classification system (NAICs) was developed jointly by the US, Canada and Mexico to provide comparability in statistics regarding business activity across North America. The NAIC replaces the US Standard Industrial Classification (SIC) system, and is used in the Employment Equity Report.

### Not in Program Plan

A benefit plan that you define outside of a program.

### Objective Assessment Template

The entity that configures the Objectives section of the appraisal.

See also: **Competency Assessment Template**, page Glossary-10

### Objectives Library

A collection of reusable objectives. HR Professionals can either create individual objectives in the Objectives Library or import them from an external source.

### Off-Boarding

Descriptive term covering all HR processes and procedures involved in removing a worker from your organization, including termination, relocation, and long-term sickness.

### OLM

Oracle Learning Management.

### On-Boarding

Descriptive term covering all HR processes and procedures involved in hiring and integrating a worker in your organization, including recruitment, hiring, and orientation.

### Online Analytical Processing (OLAP)

Analysis of data that reveals business trends and statistics that are not immediately visible in operational data.

### Online Transactional Processing (OLTP)

The storage of data from day-to-day business transactions into the database that contains operational data.

### Open Enrollment

A type of scheduled enrollment in which participants can enroll in or alter elections in one or more benefits plans.

### Options

A level of coverage for a participant's election, such as Employee Only for a medical plan, or 2x Salary for a life insurance plan.

### Oracle FastFormula

Formulas are generic expressions of calculations or comparisons you want to repeat with different input values. With Oracle FastFormula you can write formulas using English words and basic mathematical functions. The output of FastFormulas is fed back into reports.

### Organization

A required component of employee assignments. You can define as many organizations as you want within your Business Group. Organizations can be internal, such as departments, or external, such as recruitment agencies. You can structure your organizations into organizational hierarchies for reporting purposes and for system access control.

### Organization Manager Hierarchy

An HRMS structure that contains supervisors and subordinates on a reporting chain who also own organizations. HRMS uses this hierarchy to filter the information you display in report modules.

### OSSWA

Oracle Self Service Web Applications.

### Outcome

For a unit standard competence, a behavior or performance standard associated with one or more assessment criteria. A worker achieves a unit standard competence when they achieve all outcomes for that competence.

### Overrides

You can enter overrides for an element's pay or input values for a single payroll period. This is useful, for example, when you want to correct errors in data entry for a nonrecurring element before a payroll run.

### Parameter Portlet

A portlet in which you select a number of parameters that may affect all your portlets on your page. These may include an effective date, the reporting period, the comparison type, the reporting manager, and the output currency for your reports. The parameter portlet is usually available at the top of the portal page.

### Pattern

A pattern comprises a sequence of time units that are repeated at a specified frequency. The Statutory Absence Payments (UK) feature, uses SSP qualifying patterns to determine employees entitlement to Statutory Sick Pay (SSP).

### Pattern Time Units

A sequence of time units specifies a repeating pattern. Each time unit specifies a time period of hours, days or weeks.

### Pay Scale

A set of progression points that can be related to one or more rates of pay. Employee's are placed on a particular point on the scale according to their grade and, usually, work experience.

See also: *Grade Scale*, page Glossary-19

### Pay Value

An amount you enter for an element that becomes its run item without formula calculations.

See also: *Input Values*, page Glossary-21

### Payment Type

There are three standard payment types for paying employees: check, cash and direct deposit. You can define your own payment methods corresponding to these types.

### Payroll

A group of employees that Oracle Payroll processes together with the same processing frequency, for example, weekly, monthly or bimonthly. Within a Business Group, you can set up as many payrolls as you need.

### Payroll Reversal

A payroll reversal occurs when you reverse a payroll run for a single employee, in effect cancelling the run for this employee.

### Payroll Rollback

You can schedule a payroll rollback when you want to reverse an entire payroll run, cancelling out all information processed in that run. To preserve data integrity, you can roll back only one payroll at a time, starting with the one most recently run.

### Payroll Run

The process that performs all the payroll calculations. You can set payrolls to run at any interval you want.

### People List

An SSHR line manager utility used to locate an employee.

### Performance Management Plan

The entity that defines the performance-management process for a specified period. A component of the Workforce Performance Management function.

### Performance Management Viewer (PMV)

A reporting tool that displays the report that corresponds to one or more PMF targets.

### Period of Incapacity for Work (PIW)

In the UK, this is a period of sickness that lasts four or more days in a row, and is the minimum amount of sickness for which Statutory Sick Pay can be paid. If a PIW is separated by less then the linking interval, a linked PIW is formed and the two PIWs are treated as one.

### Period of Placement

The period of time a contingent worker spends working for an enterprise. A contingent worker can have only one period of placement at a time; however, a contingent worker

can have multiple assignments during a single period of placement.

### Period Type

A time division in a budgetary calendar, such as week, month, or quarter.

### Personal Public Service Number (PPS)

The Irish equivalent to National Insurance number in the UK, or the Social Security number in the US.

### Personal Tax Credits Return (TD1)

A Revenue Canada form which each employee must complete. Used by the employee to reduce his or her taxable income at source by claiming eligible credits and also provides payroll with such important information as current address, birth date, and SIN. These credits determine the amount to withhold from the employee's wages for federal/provincial taxes.

### Person Search

An SSHR function which enables a manager to search for a person. There are two types of search, Simple and Advanced.

### Person Type

There are eight system person types in Oracle HRMS. Seven of these are combinations of employees, ex-employees, applicants, and ex-applicants. The eighth category is 'External'. You can create your own user person types based on the eight system types.

### Personal Scorecard

A collection of objectives for a single worker arising from a single Performance Management Plan.

### Personnel Actions

*Personnel actions* is a public sector term describing business processes that define and document the status and conditions of employment. Examples include hiring, training, placement, discipline, promotion, transfer, compensation, or termination. Oracle HRMS uses the term *self-service actions* synonymously with this public sector term. Oracle Self Service Human Resources (SSHR) provides a configurable set of tools and web flows for initiating, updating, and approving self-service actions.

### Plan Design

The functional area that allows you to set up your benefits programs and plans. This process involves defining the rules which govern eligibility, available options, pricing, plan years, third party administrators, tax impacts, plan assets, distribution options, required reporting, and communications.

### Plan Sponsor

The legal entity or business responsible for funding and administering a benefits plan. Generally synonymous with employer.

### Position

A specific role within the Business Group derived from an organization and a job. For example, you may have a position of Shipping Clerk associated with the organization Shipping and the job Clerk.

### Predefined Components

Some elements and balances, all primary element classifications and some secondary classifications are defined by Oracle Payroll to meet legislative requirements, and are supplied to users with the product. You cannot delete these predefined components.

### Process Rule

See Configurable Business Rules, page Glossary-10

### Professional Information

An SSHR function which allows an employee to maintain their own professional details or a line manager to maintain their direct reports professional details.

### Proficiency

A worker's perceived level of expertise in a competency, in the opinion of an assessor, over a given period. For example, a worker may demonstrate the communication competency at Novice or Expert level.

### Progression Point

A pay scale is calibrated in progression points, which form a sequence for the progression of employees up the pay scale.

See also: *Pay Scale*, page Glossary-27

### Prospect Pool

(iRecruitment) The prospect pool contains all registered users who have given permission for their information to be published.

### Provincial/Territorial Employment Standards Acts

In Canada, these are laws covering minimum wages, hours of work, overtime, child labour, maternity, vacation, public/general holidays, parental and adoption leave, etc., for employees regulated by provincial/territorial legislation.

### Provincial Health Number

In Canada, this is the account number of the provincially administered health care plan

that the employer would use to make remittances. There would be a unique number for each of the provincially controlled plans i.e. EHT, Quebec HSF, etc.

### PTO Accrual Plan

A benefit in which employees enroll to entitle them to accrue and take paid time off (PTO). The purpose of absences allowed under the plan, who can enroll, how much time accrues, when the time must be used, and other rules are defined for the plan.

### QPP

(See Canada/Quebec Pension Plan)

### QA Organization

Quality Assurance Organization. Providers of training that leads to Qualifications Framework qualifications register with a QA Organization. The QA Organization is responsible for monitoring training standards.

### Qualification Type

An identified qualification method of achieving proficiency in a competence, such as an award, educational qualification, a license or a test.

See also: *Competence*, page Glossary-10

### Qualifications Framework

A national structure for the registration and definition of formal qualifications. It identifies the unit standard competencies that lead to a particular qualification, the awarding body, and the field of learning to which the qualification belongs, for example.

### Qualifying Days

In the UK, these are days on which Statutory Sick Pay (SSP) can be paid, and the only days that count as waiting days. Qualifying days are normally work days, but other days may be agreed.

### Qualifying Pattern

See: *SSP Qualifying Pattern*, page Glossary-37

### Qualifying Week

In the UK, this is the week during pregnancy that is used as the basis for the qualifying rules for Statutory Maternity Pay (SMP). The date of the qualifying week is fifteen weeks before the expected week of confinement and an employee must have been continuously employed for at least 26 weeks continuing into the qualifying week to be entitled to SMP.

### Quebec Business Number

In Canada, this is the employer's account number with the Ministere du Revenu du Quebec, also known as the Quebec Identification number. It consists of 15 digits, the first 9 identify the employer, the next 2 identify the type of tax account involved (payroll vs. corporate tax), and the last 4 identify the particular account for that tax.

### Questionnaire

A function which records the results of an appraisal.

### QuickPaint Report

A method of reporting on employee and applicant assignment information. You can select items of information, paint them on a report layout, add explanatory text, and save the report definition to run whenever you want.

See also: *Assignment Set*, page Glossary-5

### QuickPay

QuickPay allows you to run payroll processing for one employee in a few minutes' time. It is useful for calculating pay while someone waits, or for testing payroll formulas.

### Ranking

(iRecruitment) A manually entered value to indicate the quality of the applicant against other applicants for a specific vacancy.

### Rates

A set of values for employee grades or progression points. For example, you can define salary rates and overtime rates.

### Rate By Criteria

A function that enables the calculation of pay from different rates for each role a worker performs in a time period.

### Rating Scale

Used to describe an enterprise's competencies in a general way. You do not hold the proficiency level at the competence level.

### Record of Employment (ROE)

A Service Canada form that must be completed by an employer whenever an interruption of earnings occurs for any employee. This form is necessary to claim Employment Insurance benefits.

### Recruitment Activity

An event or program to attract applications for employment. Newspaper advertisements, career fairs and recruitment evenings are all examples of recruitment activities. You can group several recruitment activities together within an overall activity.

### Recurring Elements

Elements that process regularly at a predefined frequency. Recurring element entries exist from the time you create them until you delete them, or the employee ceases to be eligible for the element. Recurring elements can have standard links.

See also: *Nonrecurring Elements*, page Glossary-25, *Standard Link*, page Glossary-38

### Recruiting Area

A recruiting area consists of a set of countries, business groups, and locations. Define recruiting areas using the Generic Hierarchy function of Oracle HRMS. In iRecruitment, when managers create vacancies, they can select multiple locations as vacancy locations using recruiting areas.

### Referenced Rule

In HRMS budgeting, any predefined configurable business rule in the Assignment Modification, Position Modification, or Budget Preparation Categories you use as the basis for defining a new rule.

See Configurable Business Rules, page Glossary-10

### Region

A collection of logically related fields in a window, set apart from other fields by a rectangular box or a horizontal line across the window.

See also: *Block*, page Glossary-7, *Field*, page Glossary-18

### Registered Pension Plan (RPP)

This is a pension plan that has been registered with Revenue Canada. It is a plan where funds are set aside by an employer, an employee, or both to provide a pension to employees when they retire. Employee contributions are generally exempt from tax.

### Registered Retirement Savings Plan (RRSP)

This is an individual retirement savings plan that has been registered with Revenue Canada. Usually, contributions to the RRSP, and any income earned within the RRSP, is exempt from tax.

### Registered User

(iRecruitment) A person who has registered with the iRecruitment site by entering an e-mail address and password. A registered user does not necessarily have to apply for

jobs.

### Reporting Group

A collection of programs and plans that you group together for reporting purposes, such as for administrative use or to meet regulatory requirements.

### Report Parameters

Inputs you make when submitting a report to control the sorting, formatting, selection, and summarizing of information in the report.

### Report Set

A group of reports and concurrent processes that you specify to run together.

### Requisition

The statement of a requirement for a vacancy or group of vacancies.

### Request Groups

A list of reports and processes that can be submitted by holders of a particular responsibility.

See also: *Responsibility*, page Glossary-34

### Residual

The amount of unused paid time off entitlement an employee loses at the end of an accrual term. Typically employees can carry over unused time, up to a maximum, but they lose any residual time that exceeds this limit.

See also: *Carry Over*, page Glossary-8

### Responsibility

A level of authority in an application. Each responsibility lets you access a specific set of Oracle Applications forms, menus, reports, and data to fulfill your business role. Several users can share a responsibility, and a single user can have multiple responsibilities.

See also: *Security Profile*, page Glossary-36, *User Profile Options*, page Glossary-41, *Request Groups*, page Glossary-34, *Security Groups*, page Glossary-34

### Resume

A document that describes the experience and qualifications of a candidate.

### RetroPay

A process that recalculates the amount to pay an employee in the current period to account for retrospective changes that occurred in previous payroll periods.

### Retry

Method of correcting a payroll run or other process *before* any post-run processing takes place. The original run results are deleted and the process is run again.

### Revenue Canada

Department of the Government of Canada which, amongst other responsibilities, administers, adjudicates, and receives remittances for all taxation in Canada including income tax, Employment Insurance premiums, Canada Pension Plan contributions, and the Goods and Services Tax (legislation is currently proposed to revise the name to the Canada Customs and Revenue Agency). In the province of Quebec the equivalent is the Ministere du Revenu du Quebec.

### Reversal

Method of correcting payroll runs or QuickPay runs *after* post-run processing has taken place. The system replaces positive run result values with negative ones, and negative run result values with positive ones. Both old and new values remain on the database.

### Reviewer (SSHR)

A person invited by an appraising manager to add review comments to an appraisal.

### RIA

Research Institute of America (RIA), a provider of tax research, practice materials, and compliance tools for professionals, that provides U.S. users with tax information.

### Rollback

Method of removing a payroll run or other process *before* any post-run processing takes place. All assignments and run results are deleted.

### Rollup

An aggregate of data that includes subsidiary totals.

### Run Item

The amount an element contributes to pay or to a balance resulting from its processing during the payroll run. The Run Item is also known as calculated pay.

### Salary Basis

The period of time for which an employee's salary is quoted, such as hourly or annually. Defines a group of employees assigned to the same salary basis and receiving the same salary element.

### Salary Rate

The rate of pay associated with a grade or step. Used by Grade/Step Progression.

### Scheduled Enrollment

A benefits plan enrollment that takes place during a predefined enrollment period. Scheduled enrollments can be administrative, or open.

### Search by Date

An SSHR sub-function used to search for a Person by Hire date, Application date, Job posting date or search by a Training event date.

### Secondment

The temporary transfer of an employee to a different business group.

### Security Group

Security groupsenable HRMS users to partition data by Business Group. Only used for Security Groups Enabled security.

See also: *Responsibility*, page Glossary-34, *Security Profile*, page Glossary-36, *User Profile Options*, page Glossary-41

### Security Groups Enabled

Formerly known as Cross Business Group Responsibility security. This security model uses security groups and enables you to link one responsibility to many Business Groups.

### Security Profile

Security profiles control access to organizations, positions and employee and applicant records within the Business Group. System administrators use them in defining users' responsibilities.

See also: *Responsibility*, page Glossary-34

### Self Appraisal

Part of the Appraisal function. This is an appraisal undertaken by an employee to rate their own performance and competencies.

### Separation Category

Separation category groups the leaving reasons. HRMS*i* refers to Termination Category as Separation Category.

See also: *termination category*, page Glossary-39

### Site Visitor

(iRecruitment) A person who navigates to the iRecruitment web site and may view job

postings. This person has not yet registered or logged in to the iRecruitment system. This individual may search for postings on the web site and also has the ability to log in or register with the iRecruitment site.

### SMP

See: *Statutory Maternity Pay*, page Glossary-38

### Social Insurance Number (SIN)

A unique number provided by Human Resources Development Canada (HRDC) to each person commencing employment in Canada. The number consists of 9 digits in the following format (###-###-###).

### Source Deductions Return (TP 1015.3)

A Ministere du Revenu du Quebec form which each employee must complete. This form is used by the employee to reduce his or her taxable income at source by claiming eligible credits and also provides payroll with such important information as current address, birth date, and SIN. These credits determine the amount of provincial tax to withhold from the employee's wages.

### Special Information Types

Categories of personal information, such as skills, that you define in the Personal Analysis key flexfield.

### Special Run

The first run of a recurring element in a payroll period is its normal run. Subsequent runs in the same period are called special runs. When you define recurring elements you specify Yes or No for special run processing.

### SSHR

Oracle Self-Service Human Resources. An HR management system using an intranet and web browser to deliver functionality to employees and their managers.

### SSP

See: *Statutory Sick Pay*, page Glossary-38

### SSP Qualifying Pattern

In the UK, an SSP qualifying pattern is a series of qualifying days that may be repeated weekly, monthly or some other frequency. Each week in a pattern must include at least one qualifying day. Qualifying days are the only days for which Statutory Sick Pay (SSP) can be paid, and you define SSP qualifying patterns for all the employees in your organization so that their entitlement to SSP can be calculated.

### Standard HRMS Security

The standard security model. Using this security model you must log on as a different

user to see a different Business Group.

### Standard Link

Recurring elements with standard links have their element entries automatically created for all employees whose assignment components match the link.

See also: *Element Link*, page Glossary-16, *Recurring Elements*, page Glossary-33

### Statement of Commissions and Expenses for Source Deduction Purposes (TP 1015.R.13.1)

A Ministere du Revenu du Quebec form which allows an employee who is paid partly or entirely by commissions to pay a constant percentage of income tax based on his or her estimated commissions for the year, less allowable business expenses.

### Statement of Earnings (SOE)

A summary of the calculated earnings and deductions for an assignment in a payroll period.

### Statement of Remuneration and Expenses (TD1X)

In Canada, the Statement of Remuneration and Expenses allows an employee who is paid partly or entirely by commission to pay a constant percentage of income tax, based on his or her estimated income for the year, less business-related expenses.

### Statutory Adoption Pay

In the UK, Statutory Adoption Pay (SAP) is payable to a person of either sex with whom a child is, or is expected to be, placed for adoption under UK law.

### Statutory Maternity Pay

In the UK, you pay Statutory Maternity Pay (SMP) to female employees who take time off work to have a baby, providing they meet the statutory requirements set out in the legislation for SMP.

### Statutory Sick Pay

In the UK, you pay Statutory Sick Pay (SSP) to employees who are off work for four or more days because they are sick, providing they meet the statutory requirements set out in the legislation for SSP.

### Statutory Paternity Pay

In the UK, Statutory Paternity Pay Birth (SPPB) is payable to a person supporting the mother at the time of birth. In cases of adoption, the primary carer receives Statutory Adoption Pay, while the secondary carer receives Statutory Paternity Pay Adoption (SPPA).

### Suitability Matching

An SSHR function which enables a manager to compare and rank a persons competencies.

### Superannuation Guarantee

An Australian system whereby employers are required to contribute a percentage of an eligible employee's earnings to a superannuation fund to provide for their retirement.

### Supplier

An internal or external organization providing contingent workers for an organization. Typically suppliers are employment or recruitment agencies.

### Supporting Objective

An objective aligned with another objective. Supporting objectives contribute to the achievement of the objectives they support.

### Tabbed Regions

Parts of a window that appear in a stack so that only one is visible at any time. You click on the tab of the required region to bring it to the top of the stack.

### Task Flows

A sequence of windows linked by buttons to take you through the steps required to complete a task, such as hiring a new recruit. System administrators can create task flows to meet the needs of groups of users.

### Tax Point

The date from which tax becomes payable.

### Template Letter

Form letter or skeleton letter that acts as the basis for creating mail merge letters. The template letter contains the standard text, and also contains field codes, which are replaced by data from the application during the mail merge process.

### Terminating Employees

You terminate an employee when he or she leaves your organization. Information about the employee remains on the system but all current assignments are ended.

### Termination Category

When employees leave an enterprise, the decision is either made by the employee or by the enterprise. When the decision is made by the employee the termination is Voluntary. When the decision is made by the enterprise, the termination is Involuntary.

HRMSi elsewhere refers to Termination Category as Separation Category.

### Termination Rule

Specifies when entries of an element should close down for an employee who leaves your enterprise. You can define that entries end on the employee's actual termination date or remain open until a final processing date.

### Tips

An SSHR user assistance component that provides information about a field.

### Total Compensation Statement

A module to communicate compensations, rewards, and benefits to employees and contingent workers.

### Transaction Type

In AME, an integrating application may divide its transactions into several categories, where each category requires a distinct set of approval rules. Each set of rules is a transaction type. Different transaction types can use the same attribute name to represent values that the application fetches from different places. This enables several transaction types to share approval rules, thus facilitating a uniform approval policy across multiple transaction types.

### Transcentive

A third-party compensation management solutions provider, that provides additional information about benefits choices.

### Unit Standard

A nationally registered document that describes a standard of performance. The standard is typically defined and maintained by industry representatives.

### Unit Standard Competency

A competency that is defined in a Unit Standard and linked to a Qualifications Framework qualification.

### Upload

The process of transferring the data from a spreadsheet on your desktop, created using Web ADI, back to the Oracle HRMS application.

### User Assistance Components

SSHR online help comprising tips and instructions.

### User Balances

Users can create, update and delete their own balances, including dimensions and balance feeds.

See also: *Balances*, page Glossary-6

### User Profile Options

Features that allow system administrators and users to tailor Oracle HRMS to their exact requirements.

See also: *Responsibility*, page Glossary-34, *Security Profile*, page Glossary-36

### User-based Security

With this type of security, the application generates the security permissions for a current user when that user logs on to a system. The system uses the security profile (can be position, supervisor, or organization-based, for example) to generate security permissions for the current user, for example, based on the user's position. An alternative to user-based security is a security profile with defined security rules, for example, to specify that the top-level position for a position-based security profile is Position A, irrespective of the current user's position.

### View

An example of an interface that you can use to download data from the Oracle HRMS application to a spreadsheet using Web ADI.

### Viewer (SSHR)

A person with view only access to an appraisal. An appraising manager or an employee in a 360 Degree Self appraisal can appoint view only access to an appraisal.

### Viewer (Web ADI)

A desktop application, such as a spreadsheet or word processing tool, that you use to view the data downloaded from Oracle HRMS via Web ADI.

### Voluntary

Term used in turnover to describe employees who have ceased employment with the enterprise of their own accord, for example, by resigning.

### Waiting Days

In the UK, statutory Sick Pay is not payable for the first three qualifying days in period of incapacity for work (PIW), which are called waiting days. They are not necessarily the same as the first three days of sickness, as waiting days can be carried forward from a previous PIW if the linking interval between the two PIWs is less than 56 days.

**WCB Account Number**

In Canada, this is the account number of the provincially administered Workers' Compensation Board that the employer would use to make remittances. There would be a unique number for each of the provincially controlled boards i.e. Workplace Safety & Insurance Board of Ontario, CSST, etc.

**Work Choices**

Also known as Work Preferences, Deployment Factors, or Work Factors. These can affect a person's capacity to be deployed within an enterprise, such willingness to travel or relocate. You can hold work choices at both job and position level, or at person level.

**Worker**

An employee, page Glossary-16 or a contingent worker, page Glossary-11

**Workers' Compensation Board**

In Canada, this is a provincially governed legislative body which provides benefits to employees upon injury, disability, or death while performing the duties of the employer. Workers' Compensation Board premiums are paid entirely by the employer.

**Workflow**

An Oracle application which uses charts to manage approval processes and in addition is used in SSHR to configure display values of sections within a web page and instructions.

**Workforce Measurement Type (WMT)**

Groups of different units combined to measure the workforce. The most common units are headcount and full time equivalent.

**Workforce Measurement Value (WMV)**

A WMT value, for example, headcount or FTE.

**Workforce Performance Management**

The Oracle HRMS functions that support enterprise-directed objective setting, management, and assessment.

**Work Structures**

The fundamental definitions of organizations, jobs, positions, grades, payrolls and other employee groups within your enterprise that provide the framework for defining the work assignments of your employees.

# Index

## D

## E



## F

## G

## N



## O



## P

## Q



## R



## S

## T



## U

## V



## W