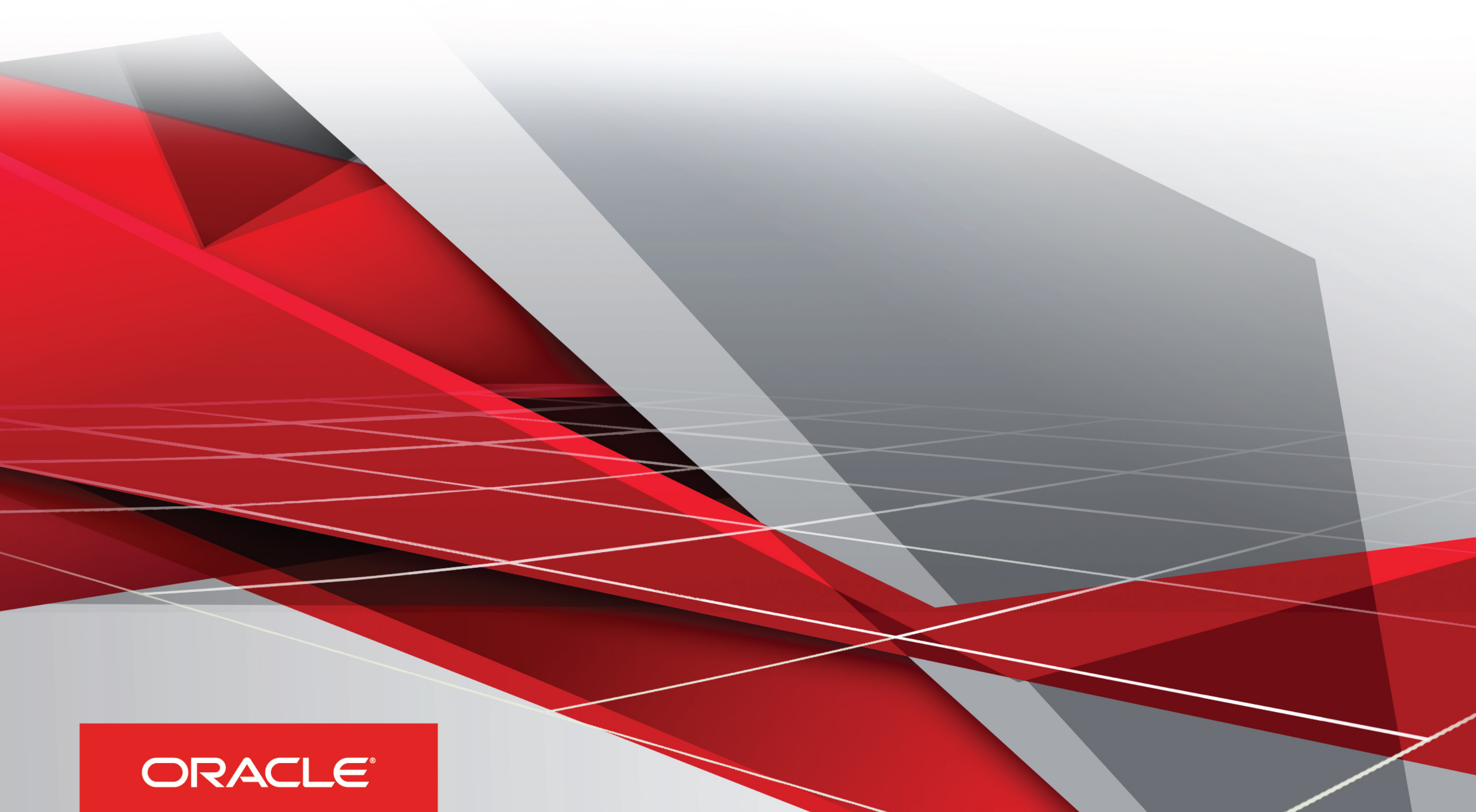
Oracle

Global Human Resources Cloud Using Fast Formula

Release 10

This guide also applies to on-premise implementations



Oracle® Global Human Resources Cloud Using Fast Formula

Part Number E61317-02

Authors: Srinivas Vellikad, Dan Douglass, Holly Zimmerman, Carla Fabrizio, Angela Brown, Tim Bisset, Janet McCandless

Contents

Preface

This Preface introduces information sources available to help you use Oracle Applications.

Oracle Applications Help

Use the help icon to access Oracle Applications Help in the application.

Note

If you don't see any help icons on your page, click the Show Help button in the global area. Not all pages have help icons.

You can also access Oracle Applications Help at https://fusionhelp.oracle.com/.

Oracle Applications Guides

To find other guides for Oracle Applications, go to:

- Oracle Applications Help, and select **Documentation Library** from the **Navigator** menu.
- Oracle Help Center at http://docs.oracle.com/

Other Information Sources

My Oracle Support

Oracle customers 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.

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

Oracle Enterprise Repository for Oracle Fusion Applications

Oracle Enterprise Repository for Oracle Fusion Applications (http://fusionappsoer.oracle.com) provides details on assets (such as services, integration tables, and composites) to help you manage the lifecycle of your software.

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.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides!

- Send e-mail to: oracle_fusion_applications_help_ww_grp@oracle.com.
- Click your user name in the global area of Oracle Applications Help, and select **Send Feedback to Oracle**.

1 Overview

Using Formulas: Explained

Fast formulas are generic expressions of calculations or comparisons that you want to repeat with different input variables. Each formula usage summarized in this topic corresponds to one or more formula types, requiring specific formula inputs and outputs. You can use the Manage Fast Formulas task in the Setup and Maintenance work area, or work areas relevant to the formula type, such as Payroll Calculation.

Note
Requirements for specific formula inputs and outputs are explained in separate chapters of the Oracle Global HR Cloud: Using Fast Formula guide.

Calculate Payrolls

You can write payroll calculations and skip rules for elements to represent earnings and deductions.

With fast formulas you can:

- Associate more than one payroll formula with each element to perform different processing for employee assignments with different statuses.
- Define elements and formulas for earnings and deductions with highly complex calculations requiring multiple calls to the database.
- Associate a skip rule formula with an element to define the circumstances in which it's processed.
- Customize the predefined proration formula to control how payroll runs prorate element entries when they encounter an event, such as a mid-period change in an element entry value.

Define Custom Calculations for Benefits Administration

You can use formulas to structure your benefit plans. Formulas provide a flexible alternative to the delivered business rules. Use formulas to configure:

- Date calculations, such as enrollment start and end dates, rate or coverage start and end dates, waiting periods and enrollment periods, or action item due dates
- Calculations of rate and coverage amount, minimum and maximum, or upper and lower limits
- Certification requirements
- Partial month and proration calculations
- Eligibility and participation evaluation

For example, you can write a formula to calculate benefits eligibility for those cases where the provided eligibility criterion does not accommodate your particular requirements.

Note
For more information, see Benefits Fast Formula Reference Guide (1456985.1) on My Oracle Support at https://support.oracle.com.

Validate Element Inputs or User-Defined Tables

Use lookups or maximum and minimum values to validate user entries.

For more complex validations you can write a formula to check the entry. You can also use a formula to validate entries in user tables.

Edit the Rules for Populating Work Relationship or Payroll Relationship Groups

You can define criteria to dynamically populate a payroll relationship group or work relationship group.

When you create a payroll relationship group or work relationship group formula type, you can choose to use an expression editor or a text editor. The expression editor makes it easy to build criteria to define the group. For more complex conditions, such as validations, you can select the text editor.

Define Custom Configuration for Compensation

To extend the existing flexibility of compensation plan configuration write formulas to customize:

- Start and end dates for compensation allocations under individual compensation plans
- Person selection, hierarchy determination, column default values, and currency selection for workforce compensation plans
- The source of items displayed in total compensation statements

2 Writing Formulas

Writing a Fast Formula Using Expression Editor: Worked Example

This example demonstrates how to create a fast formula that groups executive workers for reporting and processing. All executive workers are in department EXECT_10000. Once the formula is created, it will be added to the object group parameters so that only those workers in department EXECT_10000 are used in processing.

Before you create your formula, you may want to determine the following:

Decisions to Consider	In This Example
Is the formula for a specific legislative data group?	Yes, InVision
What is the formula type for this formula?	Payroll Relationship Group

Creating a Fast Formula Using the Expression Editor

1. On the Payroll Calculation Tasks page, click **Manage Fast Formulas** to open the Manage Fast Formulas page.
2. On the Manage Fast Formula page, click the **Create** icon to create a new formula.
3. On the Create Fast Formula page, complete the fields as shown in this table.

Field	Value
Formula Name	Executive Payroll Relationship Group
Type	Payroll Relationship Group
Description	Executive Workers
Legislative Data Group	Vision LDG
Effective As-of Date	1-Jan-2010
Type of Editor	Expression Builder

Field	Value
	Note For more complex conditions to create a group, you can select Text. However, once you save the formula, you can't change the type of editor.

4. Click **Continue**.
5. In the Formula Details section, click **Add After** to add a row to enter the fields in this table.

Conjunction	Database Item Name	Data Type	Operand	Literal Value
	DEPARTMENT	Character	=	'EXECT_10000'
And	SELECT_EMP	Character	=	'YES'

6. Click **Compile**.
7. Click **Save**.

Related Topics

- Formula Operators: Explained

Writing a Fast Formula Using Formula Text: Worked Example

This example demonstrates how to create a fast formula using the text editor to return the range of scheduled hours for managers and a different range for other workers.

Before you create your formula, you may want to determine the following:

Decisions to Consider	In This Example
Is the formula for a specific legislative data group?	No, this is a global formula that can be used by any legislative data group.
What is the formula type for this formula?	Range of Scheduled Hours
Are there any contexts used in this formula?	No

Decisions to Consider	In This Example
Are there any database item defaults?	Yes, ASG_JOB
Are there any input value defaults?	No
What are the return values?	MIN_HOURS, MAX_HOURS, FREQUENCY

Creating a Fast Formula Using the Text Editor to Determine a Manager's Scheduled Hours

1. On the Overview page in the Setup and Maintenance work area, search for the Manage Fast Formulas Task.
2. Click **Go to Task**.
3. On the Manage Fast Formula page, click the **Create** icon to create a new formula.
4. On the Create Fast Formula page, complete the fields as shown in this table.

Field	Value
Formula Name	Manager Range of Scheduled Hours
Formula Type	Range of Scheduled Hours
Description	Manager's Range of Hours
Effective Start Date	1-Jan-2010

5. Click **Continue**.
6. Enter the following formula details in the Formula Text section:

```
/* DATABASE ITEM DEFAULTS BEGIN */
DEFAULT FOR asg_job IS ' '
/* DATABASE ITEM DEFAULTS END */
JOB_1 = ASG_JOB
IF JOB_1 = 'Manager' then
(MIN_HOURS = 25
MAX_HOURS = 40
FREQUENCY = 'H')
else
(MIN_HOURS = 20
MAX_HOURS = 35
FREQUENCY = 'H')
return MIN_HOURS, MAX_HOURS, FREQUENCY
```

7. Click **Compile**.
8. Click **Save**.

Related Topics

- Using Formula Components: Explained
- Formula Operators: Explained

Formula Performance Improvements: Explained

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

Variable Names and Aliases

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

Inputs Statements

Use INPUTS statements rather than database items whenever possible. It speeds up the running of your payroll by eliminating the need to access the database for the input variables.

An example of inefficient formula without INPUTS statement is:

```
SALARY = SALARY_ANNUAL_SALARY / 12
RETURN SALARY
```

An example of efficient use of INPUTS statements is:

```
INPUTS ARE ANNUAL_SALARY
SALARY = ANNUAL_SALARY / 12
RETURN SALARY
```

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 the formula process more quickly. Doing this, however, causes unnecessary database calls which slows processing.

An example of an inefficient use of database items is:

```
S = SALARY
A = AGE
IF S < 20000 THEN
IF A < 20 THEN
TRAINING_ALLOWANCE = 30
ELSE
TRAINING_ALLOWANCE = 0
```

An example of an efficient use of database items is:

```
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 example only fetches AGE if salary is less than 20000.

Balance Dimensions

Wherever possible, only use balance dimensions for single assignments in formulas. Multiple assignments require more calculation time, leading to slower processing time.

Normally, only a small number of workers have multiple assignments. The presence of a small number doesn't lead to any significant increase in overall processing time. However, there could be a problem if you unnecessarily link balance dimensions for multiple assignments into general formulas.

Formula Compilation Errors: Explained

Compilation errors display in the Manage Fast Formulas page after you compile the formula. The compiler aborts the compilation process when it encounters an error. Error messages display the line number and type of error encountered.

Common Compilation Errors

This table lists the type and description of several common formula compilation errors.

Formula Error	Description
Syntax Error	The formula text violates the grammatical rules for the formula language. An example is using IF1 instead of IF for an IF statement.
Incorrect Statement Order	ALIAS, DEFAULT, or INPUT statements come after other statements.
Misuse of ASSIGNMENT Statement	Occurs when any of these conditions exist: • An **ASSIGNMENT** assigns a value to a database item. • A context is assigned a value externally to a **CHANGE_CONTEXTS** statement. • The formula assigns a value to a non-context variable within a **CHANGE_CONTEXTS** statement. CHANGE_CONTEXTS statements can be used in a formula.
Misuse of ALIAS Statement	You can only use an ALIAS statement for a database item.
Missing DEFAULT Statement	A database item that specifies defaulting must have a DEFAULT statement.
Misuse of DEFAULT Statement	A DEFAULT statement is specified for a variable other than an input or database item.

Formula Error	Description
Uninitialized Variable	The compiler detects that a variable is uninitialized when used. The compiler can't do this in all cases. This error often occurs when the formula includes a database item that requires contexts that the formula type doesn't support. The formula treats the database item as a local variable. For example, balance database items require the PAYROLL_ REL_ACTION_ID PAYROLL_ ASSIGNMENT_ID and CALC_ BREAKDOWN_ ID contexts. Generally you can only use them in formulas of type Oracle Payroll.
Missing Function Call	The compiler does not recognize a function call. The combination of return type, function name, and parameter types does not match any available function.
Incorrect Operator Usage	An instance of a formula operator use doesn't match the permitted uses of that operator. For example, the + operator has two permitted uses. The operands are both of data type NUMBER, or both of data type TEXT.
Inconsistent Data Type Usage	The formula uses a formula variable of more than one data type. Or the formula uses a database item or context with the wrong data type. For example, Variable A is assigned a NUMBER value at the start of the formula, but is assigned a TEXT value later in the formula.
EXIT Statement Not Within WHILE Loop	A condition that eventually becomes false or an EXIT call for exiting the loop doesn't exist.
Misuse of Context	The formula uses a variable as a context, or a context as a variable. For example, a formula assigns a value to AREA1 as an ordinary variable, but later uses AREA1 as a context in a GET_CONTEXT call.

Formula Execution Errors: Explained

Fast formula execution 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.

Formula Execution Errors

This table lists the type and description of each formula execution error.

Formula Error	Description
Uninitialized Variable	Where the formula compiler can't fully determine if a variable or context is initialized, it generates code to test if the variable is initialized. When the formula executes, this code displays an error if the variable or context isn't initialized.
Divide by Zero	Raised when a numeric value is divided by zero.
No Data Found	Raised when a non-array type database item unexpectedly fails to return any data. If the database item can't return data, then it should provide a default value. You can do this by creating a default statement. An error in formula function code can also cause this error message.
Too Many Rows	Raised when a non-array type database item unexpectedly returns more than a single row of data. The cause is an incorrect assumption made about how the data is being accessed. An error in the formula function code can also cause this error message.
NULL Data Found	Raised when a database item unexpectedly returns a NULL data value. If the database item can return a NULL value, then it provides a default value. **Note** Some database items can't return a NULL value. If it can, then you can provide a default value for that DBI.
Value Exceeded Allowable Range	Raised for a variety of reasons, such as exceeding the maximum allowable length of a string.
Invalid Number	Raised when a formula attempts to convert a non numeric string to a number.

Formula Error	Description
User Defined Function Error	Raised from within a formula function. The error message text is provided as part of the formula error message.
External Function Call Error	A formula function returned an error, but didn't provide any additional information to the formula code. The function might have sent error information to the logging destination for the executing code.
Function Returned NULL Value	A formula function returned a NULL value.
Too Many Iterations	A single WHILE loop, or a combination of WHILE loops, has exceeded the maximum number of permitted iterations. The error is raised to terminate loops that can never end. This indicates a programming error within the formula.
Array Data Value Not Set	The formula attempted to access an array index that has no data value. This error occurs in the formula code.
Invalid Type Parameter for WSA_EXISTS	An invalid data type was specified in the WSA_EXISTS call.
Incorrect Data Type For Stored Item	When retrieving an item using WSA_GET, the actual data type doesn't match that of the stored item. This error occurs within the calling formula.
Called Formula Not Found	The called formula couldn't be resolved when attempting to call a formula from a formula. This issue could be due to an error in the calling formula, or because of installation issues.
Recursive Formula Call	An attempt was made to call a formula from itself. The call could be made directly or indirectly from another called formula. Recursive formula calling isn't permitted.
Input Has Different Types In Called and Calling Formulas	When calling a formula from a formula, the input data type within the called formula doesn't match the data type specified from the calling formula.
Output Has Different Types In Called and Calling Formulas	When calling a formula from a formula, the output data type within the called formula doesn't match the data type specified from the calling formula.

Formula Error	Description
Too Many Formula Calls	There are too many formula calls from other formulas.

FAQs for Writing Formulas

When do I run the Compile Formula process?

When you create or update multiple fast formulas at the same time, run the Compile Formula process on the Submit a Process or Report page from the Payroll Administration work area.

What's the difference between a formula compilation error and an execution error?

Compilation errors occur on the Manage Fast Formulas page when you compile the formula. An error message explains the nature of the error. Common compilation errors are syntax errors resulting from typing mistakes. You can view error messages on the dashboard or go to the messages tab directly after the process is run.

Execution 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.

3 Formula Components

Using Formula Components: Explained

When you're developing a formula, you must understand formula language, the rules that the application imposes on the formula, and the calculation requirements.

Create formulas using these components:

- Assignment statements
- Return statements
- Variables
- Input statements
- Expressions
- Conditions
- Comments

Note
Other topics explain additional components you can use in formulas. These include literals, database items, working storage area, calls to other formulas, functions, and operators.

To illustrate how each component is used in a formula, suppose you wanted to calculate the pay value for the WAGEelement by multiplying the number of hours an employee works each week by the hourly rate. The formula can be written as follows:

```
WAGE = HOURS_WORKED * HOURLY_RATE
RETURN WAGE
```

Assignment Statements

An assignment statement assigns a value to the **WAGE** element.

Return Statements

A return statement passes the **WAGE** value back to the payroll run. You can use a return statement to stop the formula execution without passing back any values.

Variables

There are three classes of variables:

- Input variables appear in INPUTS statements and bring values into a formula.
- Output variables appear in RETURN statements and return values from a formula. A variable can be both an input and output.
- Local variables are only used within one formula.

You can change a local variable within the formula by assigning a value to it using an assignment statement. To calculate the **WAGE** value, the formula needs to get the value for the **HOURS_WORKED**variable.

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 pass data back to the application. Below is an example showing the use of an **ANNUAL_LEAVE** variable.

```
/* Formula: Annual Leave Formula */
IF YEARS_SERVICE >= 10
THEN
ANNUAL_LEAVE = 25
ELSE
ANNUAL_LEAVE = 20 + FLOOR (YEARS_SERVICE/2)
RETURN ANNUAL_LEAVE
```

Input Statements

You can use **HOURS_WORKED** as an input value of the **WAGE** element. To pass the element input values to the formula during processing, define an input statement as follows:

```
INPUTS ARE HOURS_WORKED
WAGE = HOURS_WORKED * HOURLY_RATE
RETURN WAGE
```

Note

This is a payroll application example. The name used in the input statement must be the same as the name of the element input value. Multiple words must be joined by underscores. Other input statements that have nothing to do with elements would have their own rules for formula input variables. In this example, the **HOURS_WORKED** input variable is numeric. If the input variable is not numeric, you must specify the type. For example,

```
INPUTS ARE START_DATE (DATE)
```

Expressions

Each function or calculation is one expression. You can nest expressions to create more complex calculations. You can use brackets to control the order in which calculations are performed.

The formula evaluates expressions within brackets first. Within nested brackets, evaluation proceeds from the least inclusive set to the most inclusive set. If you don't use brackets, the following hierarchal order of execution is implied: multiplication and division then addition and subtraction.

Expressions combine constants and variables with operators (+, -, *, /), array methods, and functions to return a value of a certain data type. For example, the expression (3 + 2) returns a value of 5, and is a **NUMBER** data type. The format of an expression is:

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

You can combine a number of sub-expressions into a single expression. For example, you can combine the sub-expressions (3 + 2) and **MONTHS_BETWEEN**(start_date, end_date) into a single expression as follows:

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

You can also use expressions inside functions, such as:

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

Operands in an expression are usually of the same data type which is 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)) + BONUS
```

BONUS is the operand for the above expression. The return value is **GREATEST**. The arguments for **GREATEST** are separate expressions.

Conditions

You can use conditions to process expressions based on whether a certain condition occurs. For example,

```
TRAINING_ALLOWANCE = 0
IF (AGE < 20) THEN
TRAINING_ALLOWANCE = 30
```

This formula checks if the condition (AGE < 20) is true or false. If it's true, the formula processes the statement that follows the word **THEN**. If the condition isn't true, the formula ignores this statement.

Comments

Use comments to explain all or part of a formula. Also, you can change some formula lines into comments until they are ready to be used. Comments are designated by the comment delimiters of /* and */. Anything written inside these delimiters is a comment. You can place comments anywhere within a formula. The beginning of a formula should contain the following comments:

- The formula title and a short purpose statement.
- A description of the formula inputs.
- A list of variables and literals that may require updating.
- An explanation of the formula's calculation.
- The dates of any modifications, the name of the person modifying the formula, and the reason for the change.

Note
Do not put a comment within a comment. This causes a syntax error when the formula is compiled.

Related Topics

- Formula Functions: Explained

Formula Statements: Explained

You use formula statements to provide instructions that you want your formula to carry out. When working with statements, it's important to have knowledge of the different statement types, the required order, and how to group statements.

Statement Types

The table below describes the various statements that you can use in your formulas to provide instructions.

Statement	Statement Form	Description and Example
ALIAS	ALIAS name1 AS name2	Provides a different name for a database item or global value. Sometimes the database item names provided by the application are too long to conveniently use in a formula. Use the ALIAS statement to shorten the name of a database item. Once the ALIAS is created, use it instead of the database item name. Using an alias is more efficient than assigning the

Statement	Statement Form	Description and Example
		database item to a local variable with a short name.

ASSIGNMENT	variable = expression
	
	array[index] = expression
		Assigns an expression value to a variable or an array variable at an index position. A formula evaluates the expression on the right hand side of the statement. It 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.
		
		Within a CHANGE_ CONTEXTS statement, assign values to contexts only. Outside a CHANGE_ CONTEXTS statement, assign values to input, output, and local variables only.

CHANGE_ CONTEXTS	(context1 = expression1 [,context2 = expression2] ...	Changes one or more contexts within a formula. Within the CHANGE_ CONTEXTS statement, use ASSIGNMENT statements to assign the new values.
DEFAULT	DEFAULT FOR variable IS literal DEFAULT_ DATA_VALUE FOR variable IS literal	The DEFAULT FOR statement provides a value that the formula uses for a formula input or database item if: • The input doesn't provide a value. • The database item isn't found • The value of a non-array database item is NULL The DEFAULT_DATA_VALUE FOR statement provides a value for an array database item where individual data values are NULL. Some database items are defined to require a default value because they could return no data or NULL values from the database. **DEFAULT FOR HOURLY_RATE IS 3.00**

Statement	Statement Form	Description and Example
		INPUTS ARE HOURLY_RATE **X = HOURS_WORKED * HOURLY_RATE**
EXIT	EXIT	Immediately exits from the enclosing WHILE loop. You can't use the EXIT statement outside of a WHILE loop.
FORMULA CALLING	SET_ INPUT(input [,value]) EXECUTE(formula) The formula RATE_FORMULA is called to get a value for HOURLY_RATE. RATE_FORMULA.	Calls a formula from another formula. For example, formulas can call a small formula that performs a common calculation. Use this approach to avoid writing long formulas.

IF	IF condition THEN statements
	
	IF condition THEN statements ELSE statements
		Executes one or more statements if a condition evaluates as true. Use the IF ELSE statement to specify a set of statements to execute if the condition evaluates to false.

INPUT	INPUTS ARE input1 [, input2] ...	Lists the input variables for the formula. There is only one INPUT statement in a formula.
RETURN	RETURN [output1] [, output2] ...	Causes a formula to stop executing immediately. For its value to be returned to the caller, you must enter a formula output variable in the RETURN statement that stopped the formula. You can enter multiple return statements in a formula.
WHILE	WHILE condition LOOP statements In this example, 'A' is an array variable with a numerical index.	Executes a number of statements as long as a condition evaluates to true. To prevent endless looping, an error occurs if the WHILE statement loop performs an excessive number of iterations.

Statement	Statement Form	Description and Example
WORKING STORAGE	WSA_ DELETE([item]) - Deletes values from the storage area. WSA_ EXISTS(item[type]) - Determine if an item exists . WSA_GET(item, value) - Fetches values from the storage area. WSA_SET(item, value) - Sets values from the storage area. In the example a number of rates are set up.	Stores reference data, which you can set, fetch, or delete.

Ordering Statements

In your formulas you must place statements in the following order:

1. ALIAS statements, if any
2. DEFAULT statements, if any
3. INPUT statements, if any
4. Other statements

Grouping Statements

If you want to group more than one statement under IF/THEN statements, ELSE clauses, WHILE loops, or CHANGE_CONTEXTS, enclose the group of statements within brackets. In the absence of brackets, the preceding statement only applies to the first statement.

Correct example:

```
I = A.FIRST
WHILE (A.EXISTS(I)) LOOP
(
 A[I] = I
 I = A.NEXT(I,-1)
)
```

Incorrect example:

```
I = A.FIRST
WHILE (A.EXISTS(I)) LOOP
 A[I] = I
 I = A.NEXT(I,-1) /* This is not executed as part of the loop. */
```

Naming Variables: Explained

When you add variables to your formulas there are two acceptable naming schemes. In addition, it's important to avoid using reserved words as variable names.

Naming Schemes

Use one of these naming schemes:

- Variable names comprise one or more words, joined by underscores. The words must each start with a letter and can be followed by a combination of letters, and digits.
- Variable names begin and end with double quotes (''). Between the quotes, you can use any printable characters, such as ''This is a quoted variable name''. Please note that any word consisting of only digits could be mistaken for numbers.

Formulas aren't case sensitive. For example, the variable named EMPLOYEE_NAME is the same as the variable employee_name.

Restriction

The maximum length of a variable name is 255 characters.

Reserved Words

You must not use the following reserved words as the names of variables:

Source of Reserved Words	Reserved Words
Statements	**ALIAS** **AND** **ARE** **AS** **CHANGE_CONTEXTS** **DEFAULT** **DEFAULT_DATA_VALUE** **DEFAULTED** **ELSE** **EXIT** **FOR** **IF** **INPUTS** **IS** **LIKE** **LOOP** **NEED_CONTEXT** **NOT** **OR** **RETURN** **THEN** **USING** **WAS** **WHILE**
Array Types	**EMPTY_DATE_NUMBER** **EMPTY_NUMBER_NUMBER** **EMPTY_TEXT_NUMBER** **EMPTY_DATE_TEXT** **EMPTY_NUMBER_TEXT** **EMPTY_TEXT_TEXT**
Formula Data Types	**DATE**

Source of Reserved Words	Reserved Words
	DATE_NUMBER **DATE_TEXT** **NUMBER** **NUMBER_NUMBER** **NUMBER_TEXT** **TEXT** **TEXT_NUMBER** **TEXT_TEXT**
Array Methods	**COUNT** **DELETE** **EXISTS** **FIRST** **LAST** **NEXT** **PREVIOUS** **PRIOR**
Built-in Calls	**CONTEXT_IS_SET** **EXECUTE** **GET_CONTEXT** **GET_OUTPUT** **IS_EXECUTABLE** **SET_INPUT** **WSA_DELETE** **WSA_EXISTS** **WSA_GET** **WSA_SET**

Database Items: Explained

Database items exist in the application database and have computer code associated with them. The system uses this code to find data. All database items are read-only variables. You can't change database item values within a formula. If you attempt to write a value to a database item, you will receive a compilation error.

You can use database items in formulas and HCM extracts.

This topic explains the following types of database items:

- Static
- Dynamic
- Array

Static Database Items

Static database items are predefined. They include standard types of information, such as the sex, birth date, and work location of an employee. They also include data about other objects, such as the start and end dates of a payroll period.

Dynamic Database Items

The application creates dynamic database items when you create the following objects. In the case of flexfields, you must run the Generate Flexfield Database Items process to create the database items.

Object	Description
Elements	The element name is the database item name prefix.
Balances	The balance name followed by the balance dimension name is the database item name.
Formula global values	The global value name is the database item name.
Input values	The element and input value names are the database item name prefix.
Flexfields	The Generate Flexfield Database Items process creates database items for the contexts and segments of your registered HCM flexfields.

Array Database Items

Array database items have an index type of NUMBER with indexes starting at 1 and increasing by 1 without gaps. Here's an example:

```
/* 1 is the starting index for an array database item. */
I = 1
WHILE DBI.EXISTS(I) LOOP
(
 V = DBI[I] /* Do some processing with element at index I. */
 I = I + 1 /* Array database items indexes go up in steps of 1. */
)
```

Use the DEFAULT_DATA_VALUE FOR statement to set a default value in the case where an array database item could return a NULL value for an element. There can only be one DEFAULT_DATA_VALUE FOR statement for each array database item and it must appear at the start of the formula.

Here's an example of a DEFAULT_DATA_VALUE FOR statement:

```
DEFAULT_DATA_VALUE FOR A IS 0
INPUTS ARE B, C
```

Here's an example of an array database item usage error case:

```
 /* Array database item A. */
A[1] = 1
 A = B
A.DELETE(1)
A.DELETE
```

Generating Flexfield Database Items: Explained

You configure registered HCM flexfields to add contexts and segments for your business requirements. After you deploy the flexfield, you can generate database items for the flexfield for use in your formulas and extracts by submitting the Generate Flexfield Database Items process from the Payroll Checklist or Payroll Administration work areas.

You can generate DBIs for the following flexfields:

- Descriptive flexfields
- Extensible flexfields for single and multiple row routes
- Key flexfields

The process generates DBIs at the enterprise level only. As a best practice, when you submit the process, skip the legislative data group parameter so that the process generates DBIs for use by any legislative data group.

You can determine which DBIs to generate by specifying or skipping the flexfield and context parameters.

Flexfield Parameter	Context Parameter	Result
Specify parameter	Skip parameter	Generate DBIs for all the contexts and related segments for a specified flexfield
Skip parameter	Skip parameter	Generate DBIs for all registered flexfields and their contexts.

The process creates database item names with this following structure:

<FLEXFIELD_CODE><CONTEXT_CODE><SEGMENT_CODE>

When you include the database item in a formula or extract, the application returns a value for the database item, based on the flexfield context, for the segments column in the underlying flexfield table. After you generate DBIs, compile any formulas using these DBIs.

Periodically, you may need to update a flexfield structure, for example to add a segment to capture additional data. If you previously generated DBIs for a flexfield, submitting the process deletes and regenerates the associated DBIs. After the process regenerates the DBIs, be sure to compile any formulas using them.

Related Topics

- Flexfields: Overview
- Extract Components: How They Work Together

Generating Flexfield Database Items: Worked Example

This example demonstrates how to add segments and contexts to a registered HCM flexfield, and how to generate database items for the flexfield for later use in formulas and extracts.

The following table summarizes the key decisions for this scenario.

Decisions to Consider	In this Example
Which registered HCM flexfield requires database items?	Organization Information flexfield

Decisions to Consider	In this Example
What is the name of the flexfield code?	PER_ ORGANIZATION_ INFORMATION_ EFF
How many contexts include segments that require database items?	Two contexts: HCM_ CN_ PSU_ TERMINATION_ INFO HRX_ CN_ TRU_ TERMINATION_ INFO
Should I give all legislative data groups access to the generated database items for use in their formulas and extracts?	Yes

In this example, Joe plans to create formulas for an implementation in China based on information captured in the organization information flexfield. Joe configures the flexfield to add the contexts and segments. He writes a formula to calculate severance pay that returns results for the leave compensation factor based on the tax reporting unit.

Configure the Flexfield

1. In the Setup and Maintenance work area, go to the Manage Extensible Flexfields task, and search for the Organization Information EFF.
2. Click **Edit**.
3. Click **Manage Contexts**.
4. Click **Create** and create two contexts: HCM_CN_TRU_TERMINATION_INFO and HRX_CN_PSU_TERMINATION_INFO.
5. For each context, create two segments: LEAVE_COMPENSATION_FACTOR and MONTHLY_SALARY_PAID_DAYS.
6. Deploy the flexfield.

Submit the Generate Flexfield Database Process

1. In the Payroll Checklists or Payroll Administration work area, select the **Submit a Process or Report** task.
2. Complete the following parameters:

Page	Parameter	Data
Select Flow Pattern	Legislative Data Group	Skip this parameter
Select Flow Pattern	Process or Report	Generate Flexfield Database Items
Enter Parameters	Payroll Flow	Organization flexfield database items

Page	Parameter	Data
Enter Parameters	Name	PER_ ORGANIZATION_ INFORMATION_ EFF
Enter Parameters	Context	Skip this field to generate database items for all contexts.

3. Click **Next** and skip the Enter Interaction page.
4. Click **Next** and skip the Schedule page.
5. Click **Next** and review the submitted parameters on the Review page.
6. Click **Submit** to create the payroll flow.

   The submitted process creates database items for each context for the flexfield segments:

   - PER_ORGANIZATION_INFORMATION_HRX_CN_TRU_TERMINATION_LEAVE_COMPENSATION_FACTOR
   - PER_ORGANIZATION_INFORMATION_HRX_CN_TRU_TERMINATION_MONTHLY_SALARY_PAID_DAYS
   - PER_ORGANIZATION_INFORMATION_HRX_CN_PSU_TERMINATION_LEAVE_COMPENSATION_FACTOR
   - PER_ORGANIZATION_INFORMATION_HRX_CN_PSU_TERMINATION_MONTHLY_SALARY_PAID_DAYS

Create a Formula

1. Create a formula for calculating severance pay that returns a segment for LEAVE_COMPENSATION_FACTOR, if the context code is the one specified for the database item, HRX_CN_TRU_TERMINATION_INFO.

Formula Operators: Explained

Formula operators are expressions that may contain arithmetic operators. These operators determine how the formula manipulates variables and literals. For example, the plus operator (+) indicates that two items are added together. You can also use operators for string concatenation.

Types of Operators

The operator types are described in the following table.

Operator	Description	Example
+	Addition	A = B + 1

Operator	Description	Example
+ ||	String concatenation	A = 'Hello ' + 'World' B = 'Hello ' || 'World'
-	Subtraction	A = B - 1
-	Unary minus	A = -B
*	Multiplication	A = B * C
/	Division	A = B / C

Using Operators

The arithmetic operators, subtraction, multiplication, and division, can only be used with numeric operands. The addition operator can be used with numeric or text operands. The operands can be variables, literals, or sub-expressions. A formula error occurs if:

- The second operand of a division equals zero
- The result of multiplication is too large

  What is too large is determined by the normal limits in the database. For string concatenation, if the result is longer than 255 characters, a formula error is raised.

Expressions are evaluated in order from left to right. The unary minus has precedence over the other operators because it applies directly to a single sub-expression. The multiplication and division operators take precedence over addition and subtraction. For example, the expression 1 + 2 * 3 evaluates to 7 rather than 9. Brackets can be used to change precedence. For example, (1 + 2) * 3 evaluates to 9.

Literals: Explained

A literal is a piece of information that you manipulate or use in a formula. This topic explains the four types of literals: Numeric, Text, Date, and Array.

Numeric Literals

When you enter numeric literals, follow these rules:

- Don't use quotes to enclose the literal.
- Don't use commas or spaces in the number.
- Don't use exponents and floating point scientific notations.

- You can enter numbers that have a decimal component after a decimal point.
- To enter a negative number precede it with a minus sign (-).

Examples of numeric literals are:

- 63
- 3.55
- -2.3
- -.033
- -.2
- 10000

Text Literals

When you enter text literals, enclose them in single quotes. Text literals may contain spaces. To enter a single quote character in a text constant, enter two single quotes (for example, 'P O''Donnell'). Note that this is not the same as the double quote ('').

Examples of text literals are:

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

Date Literals

When you enter a date literal, enclose dates in single quotes and follow immediately with the word date, in brackets.

Use one of the following formats:

- YYYY-MM-DD''T'' HH:MI:SS.FFF''Z''
- YYYY-MM-DD HH24:MI:SS
- DD-MON-YYYY

Tip
Use one of the first two formats if you want to compile the formula under different language settings.

Examples of date literals are:

- '2010-11-04T00:00:00.000Z' (DATE)

- '1989-03-12 00:00:00' (DATE)
- '12-MAR-1989' (DATE)

Array Literals

An array holds multiple values that the formula can access using the corresponding index values. You define array literals only for an empty array of each type.

The array types are:

- Array of date values indexed by a numeric index (EMPTY_DATE_NUMBER)
- Array of number values indexed by a numeric index (EMPTY_NUMBER_NUMBER)
- Array of text values indexed by a numeric index (EMPTY_TEXT_NUMBER)
- Array of date values indexed by a text index (EMPTY_DATE_TEXT)
- Array of numeric values indexed by a text index (EMPTY_NUMBER_TEXT)
- Array of text values indexed by a text index (EMPTY_TEXT_TEXT)

Formula Variable Data Types: How They're Determined

Variable data type can be numeric, text or date. The data type determines the type of information the variable holds. You don't have to specify the variable type. Formulas determine the type from how you use the variable. For example, if you set a variable to 'J. Smith', the formula interprets it as a text variable.

Important
Inconsistent or incorrect use of variables, such as trying to add a number to a text string, causes formula compilation errors.

How Formulas Determine Variable Data Types

Formulas process the rules that determine the variable data type in the following order:

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

   ```
   INPUTS ARE SALARY_AMOUNT,
   START_DATE (DATE),
   FREQUENCY (TEXT)
   ```

   If you don't specify the variable data type in the statement, the formula assumes it's a number.

   The variable data type can be determined from a DEFAULT_FOR statement such as:

   ```
   DEFAULT FOR B IS 0 /* B is a NUMBER variable. */
   ```

In the case of array database items, the index type and the value type can be determined from a DEFAULT_DATA_VALUE FOR statement:

DEFAULT_DATA_VALUE FOR A IS EMPTY_NUMBER_NUMBER /* A is a NUMBER_NUMBER array variable. */

The variable data type can be determined from a DEFAULT_DATA_VALUE FOR statement such as:

DEFAULT_DATA_VALUE FOR A IS EMPTY_NUMBER_NUMBER /* A is a NUMBER_NUMBER array variable. */

In the case of array database items, the type can be determined from a DEFAULT FOR statement:

DEFAULT FOR B IS 0 /* B is a NUMBER variable. */

2. The formula searches the list of database items. If the variable is in the list, the data type is known.
3. If the variable appears in a context handling statement, then the formula searches the list of contexts. If the variable is in the list, then the formula knows the data type, otherwise it raises an error.
4. If the variable isn't a database item or a context, then the formula treats it as a local variable and determines the data type by the way you use the variable. For example:

   A = 'abc' /* A is a TEXT variable. */

Related Topics

- Formula Compilation Errors: Explained

Array Variables: Explained

You can use arrays for input, output, and local formula variables. These array variables can hold date, number, or text values. Arrays are similar to PL/SQL index-by tables.

Caution
Large arrays and excessive use of arrays will result in excessive memory consumption.

Array Indexes

Here are some aspects of array indexes that you should be aware of:

- The index types are either text or number.
- Text indexes are upper case unique.
- Gaps in index value sequences are permitted.
- Number indexes are truncated to remove any fractional part.
- You may iterate an array in an index either forwards or backwards.

Array Data Types

Array types are specified as DATA_TYPE_INDEX_TYPE. Here are the following types:

- NUMBER_NUMBER
- NUMBER_TEXT
- DATE_NUMBER
- DATE_TEXT
- TEXT_NUMBER
- TEXT_TEXT

Rules for Using Arrays

Formula functions can't return arrays or take array parameters. Contexts can't be array types. An attempt to reference an array value at a nonexistent index causes a formula execution error. However, an attempt to delete a value at a nonexistent index doesn't cause an error.

Array Methods

Array methods provide a way to get the first and last indexes and to get the next or prior index. These methods return the index data type. You can specify a default value for these methods, if the required indexes don't exist. A method is also provided to test the existence of an index.

Note

The array method syntax doesn't work directly with the array literal values. For example, you can't use a construct such as EMPTY_DATE_NUMBER.COUNT.

The following table provides examples of each array method:

Array Method	Description	Usage Example
<name> [<index value>]	Get the value for an index.	V = A[1]
<name> . FIRST(<default value>)	Get the first index for an array. The default value is returned if the array is empty.	I = A.FIRST(-1)
<name> . LAST(<default value>)	Get the last index for an array.	L = B.LAST(' ')
<name> . EXISTS(<index value>)	Conditional checking if a value exists at an index. The default value is returned if the array is empty.	IF A.EXISTS(1) THEN
<name> . NEXT(<index value> , <default index value>)	Get the next index given an index position. The default value is returned if there is no next index.	N = A.NEXT(1)

Array Method	Description	Usage Example
<name> . PRIOR(<index value> , <default index value>)	Get the prior index given the index position. The default value is returned if there is no prior index.	P = B. PRIOR('Two')
<name> , COUNT	Numeric method to count the array elements.	C = A.COUNT
<name , DELETE(<index value>)	Delete the element at an index position.	B. DELETE('three')
<name> , DELETE()	Delete all elements.	B.DELETE()

Iterating Through an Array

In the following example, A is an array variable with a NUMBER index. -1234 is known to be an invalid index for A so it's used as a default value when the FIRST and NEXT calls can't find an index.

```
/* -1234 is not a valid index for A in this instance, so use as default. */
NI = A.FIRST(-1234)
WHILE A.EXISTS(NI) LOOP
(
 VA = A[NI] /* Do some processing with element at index NI. */
 NI = A.NEXT(NI,-1234) /* Go to next index. */
)
```

The following example does the same thing for array variable B with a TEXT index.

```
/* 'No Index' is not a valid index for A in this instance, so use as default. */
TI = B.FIRST('No Index')
WHILE B.EXISTS(TI) LOOP
(
 VB = B[TI] /* Do some processing with element at index TI. */
 TI = B.NEXT(TI, 'No Index') /* Go to next index. */
)
The following example iterates backwards from through an array C with a NUMBER index.
/* -1234 is not a valid index for C in this instance, so use as default. */
NI = C.LAST(-1234)
WHILE C.EXISTS(NI) LOOP
(
 VC = C[NI] /* Do some processing with element at index NI. */
 NI = C.PRIOR(NI,-1234) /* Go to prior index. */
```

Formula Contexts: Explained

Formulas run within an application-specific execution context, which determines the context variables that are available to the formula. Context values act as SQL bind values when the formula fetches database item values from the database. Formulas can also pass context values into formula function calls.

Examples of contexts are:

- **EFFECTIVE_DATE** for the effective date the formula is running

- **PAYROLL_ID** for the running payroll
- **PERSON_ID** for identifying the person for who the formula is processing or evaluating

Context Value Setting

The application code calling a formula usually sets all the context values. For some complex applications, such as the payroll run, the code only sets the contexts necessary to meet general processing requirements.

For payroll formulas:

- A payroll run sets contexts for the legislative data group, date earned, the payroll being processed, the payroll relationship, payroll actions, and the person being processed.
- Additional, country-specific contexts are available. For example, the jurisdiction area and tax code context values are country-specific.

Formula Context-Handling Statements

If you use a variable in a context handling statement, the formula searches the list of contexts. The variable must appear in the contexts list; otherwise the formula displays an error. The data type is held with the context list entry.

Formula context handling statements are described below.

Statement	Purpose	Example
CHANGE_ CONTEXTS(assignment [. ..])	Changes context values within the context changing block. Inside this block, formula function calls, database items, and called formulas use the new context values. Outside the block, the formula uses the original values. You can nest context changing blocks to apply context changes in stages.	**/* * Nested Context changes: DBI1 depends upon SOURCE_ID and SOURCE_TEXT. */ CHANGE_CONTEXTS(SOURCE_TEXT = 'A') (/* SOURCE_TEXT = 'A' */ X = DBI1 /* Nesting used to change Contexts in stages. */ CHANGE_CONTEXT(SOURCE_ID = 2) (/* SOURCE_TEXT = 'A', SOURCE_ID = 2 */ Y = DBI1 /* Overriding a Context change. */ CHANGE_CONTEXTS(SOURCE_TEXT = 'B',SOURCE_ID = 3) (/* SOURCE_TEXT = 'B', SOURCE_ID = 3 */ Z = DBI1)))**
CONTEXT_ IS_ SET(context)	Tests whether or not a context value is set.	The following code tests whether or not the AREA3 context is set.

Statement	Purpose	Example
GET_ CONTEXT(context default value)	Returns a context's value if the context is set, otherwise it returns the default value specified in its second argument.	**/* AREA1 is a context of type TEXT. */ AREA1_VALUE = GET_CONTEXT(AREA1,' ')**

Working Storage Area: Explained

The working storage area is a mechanism for storing global values across formulas. Using the different call methods, you can test whether or not an item exists in the storage area, delete an item, set the value for an item, and get a value for an item. Access the values by name. The names are case-independent.

The working storage area methods are described in the below table.

Method	Description
WSA_ EXISTS(item [, type])	Test whether or not the item called **item** exists in the storage area. If **type** is specified, then the item must be of the same type. The valid values for type are one of the strings: • **DATE** • **DATE_NUMBER** • **DATE_TEXT** • **NUMBER** • **NUMBER_NUMBER** • **NUMBER_TEXT** • **TEXT** • **TEXT_NUMBER**, • **TEXT_TEXT**
WSA_ DELETE([item])	Delete the item called item. If you don't specify a name then all storage area data is deleted.
WSA_SET(item, value)	Set the value for the item called item. Any existing item of the same name is overwritten.
WSA_GET(item, default-value)	Retrieve a value for the item called item. If there's no item called item, then the method returns the default

Method	Description
	value. The data type of default-value is the expected data type for item.

Calling a Formula from a Formula: Explained

Another formula can call a formula. This enables some modularity in formula organization. The called formula name, and any formula input or output names, are specified as **TEXT** values. The names are case-independent. There are two alternative approaches to calling a formula: using a single call, or separate calls.

Consider the following aspects:

- Validation of the Called Formula
- Passing Contexts
- Alternative Methods to Call a Formula
  - Using Separate Calls
  - Using a Single Self-Contained Call
- Use Cases to Compare Methods

Validation of the Called Formula

When the formula runs, it checks to ensure the called formula can be run, and whether the specified input and output data types are correct. You can use the **IS_EXECUTABLE** call to determine whether an executable formula with a specified name exists. You must compile the formula to make it available for the specified legislative data group. Also, it must be valid on the effective date that the calling formula runs. Payroll code imposes extra restrictions based on formula type combinations.

Passing Contexts

Context values are inherited from the calling formula. You can also set or unset the context values explicitly in the nested formula call.

Alternative Methods to Call a Formula

There are two ways to call a formula from a formula:

- Using a series of separate calls
- Using a single self-contained call

Using Separate Calls

You can use three separate calls as follows:

1. Set the Inputs

   Use a **SET_INPUT** call for each formula input and context that you need to explicitly set for the formula call. You don't need to specify all formula inputs and contexts. To explicitly unset a context values, use the **SET_INPUT** call without passing the optional value parameter. Any extra inputs specified in **SET_INPUT** calls are ignored. xx Use a **SET_INPUT** call for each formula input and context that you need to explicitly set for the formula call. You don't need to specify all formula inputs and contexts. To explicitly unset a context value, use the **SET_INPUT** call without passing the optional value parameter. Any extra inputs specified in **SET_INPUT** calls are ignored.

2. Call the Formula

   Use an **EXECUTE** call to call a formula.

3. Get the Formula Outputs

   Use one or more **GET_OUTPUT** calls to fetch outputs from the last formula call.

When the formula runs, it will show an execution error if:

- The called formula is not executable.
- The called formula is already running.
- The data type of an input variable (specified using SET_INPUT) or an output variable (specified using GET_OUTPUT) doesn't match its actual data type within the called formula.

The following table summarizes the methods for calling a formula using separate calls.

Method	Description
SET_ INPUT(input [,value])	The value parameter is optional. If it's provided, the specified input is set to this value. If it's not provided, the input is passed as unset to the formula. The data type of the value is the expected data type for the input.
EXECUTE(formula)	Executes the called formula.
GET_ OUTPUT(output default-value)	Gets the value of the output parameter after calling a formula. If there's no formula output called 'output' or it's not set, the formula returns the value specified in the default value parameter. The data type of default value is the expected data type for output.

Note
Formula inputs set using **SET_INPUT** persist as long as no **EXECUTE** or **GET_OUTPUT** calls are made. Output values from a called formula persist as long as no **SET_INPUT** or new **EXECUTE** calls are made. When the calling formula exits, the process removes any saved input or output values.

Using a Single Self-Contained Call

The end result with this approach is the same as using separate calls except that:

- Input values are cleared at the start so that prior **SET_INPUT** call values are not used.

- Outputs are discarded at the end so that subsequent **GET_OUTPUT** calls just return the default values.

Use the **CALL_FORMULA** method as follows:

```
CALL_FORMULA(formula, [set statement, get statement])
```

A **SET** statement is a **SET_INPUT** call. A **GET** statement assigns a **GET_OUTPUT** call result to a variable in the calling formula. The execution order is:

1. SET_INPUT calls
2. EXECUTE call
3. GET_OUTPUT assignments

The compiler generates code to execute in this order even if SET and GET statements are interspersed.

Calling a Formula from a Formula: Examples

These examples illustrate how to call a formula from another formula using two methods: a series of separate calls or a single self-contained call. The examples include two versions of a wage formula, and a table comparing the two methods using a series of use cases.

The first two examples show different versions of the wage formula. The following points apply to both examples:

- The formula calls RATE_FORMULA to get a value for HOURLY_RATE.
- The RATE_FORMULA has a text input called UNIT.
- The formula call sets the UNIT input to 'Hourly'.
- The RATE_FORMULA returns the rate in the output variable called RATE.
- The GET_OUTPUT call returns 0.00 if the RATE_FORMULA doesn't return RATE.

Wage Formula Using Separate Calls

This example illustrates a formula call from separate calls.

```
SET_INPUT('UNIT', 'Hourly')
EXECUTE('RATE_FORMULA')
HOURLY_RATE = GET_OUTPUT('RATE',0.0)
WAGE = HOURS_WORKED * HOURLY_RATE
RETURN WAGE
```

Wage Formula Using a Self-Contained Call

This example illustrates a formula called from a self-contained call.

```
CALL_FORMULA
```

```
('RATE_FORMULA','Hourly' > 'UNIT'
/* SET_INPUT('UNIT', 'Hourly') */
,HOURLY_RATE < 'RATE' DEFAULT 0.0
/* HOURLY_RATE = GET_OUTPUT('RATE',0.0) */
)
WAGE = HOURS_WORKED * HOURLY_RATE
RETURN RATE
```

Sample Expressions to Compare Methods

The following table provides sample expressions you use in the two methods to:

- Call a formula
- Set inputs and context values
- Unset context values
- Get output values into a variable or array variable
- Provide a default output value

Note
SET_INPUT or > statements have no effect if the calling formula has no formula input or context of the same name.

Use Case	Using Separate Calls	Using a Self-Contained Call
Execute a formula where the formula GET_RATES is executed	**EXECUTE('GET_RATES')**	Use within a CALL_FORMULA statement
Set an input value in the called formula where you round up EXTRA_HOURS to 2 decimal places and set the input OVERTIME in the called formula. The called formula should contain the statement:	**SET_INPUT ('OVERTIME' ,ROUNDUP(EXTRA_HOURS,2))**	Use within a CALL_FORMULA statement
Leave a formula input value unset inside the called formula, where RATE is not a formula context.	A SET_INPUTS statement is not required, but you can use the following:	A SET statement is not required, but you can use the following:
Inherit a context value from the called formula. For example, both the calling and called formula support the AREA1	No statements are required to do this.	No statements are required to do this.

Use Case	Using Separate Calls	Using a Self-Contained Call
context. The called formula inherits the AREA1 context value from the calling formula.		
Set a context value inside a called formula, where the called formula supports the AREA1 context and AREA1 has to be set to 'London' in the called formula.	**SET_INPUT ('AREA1' ,'London')**	**'London' > 'AREA1'**
Call a formula with an unset context value, where the called formula supports the AREA1 context and AREA1 has to be unset in the called formula.	**SET_INPUT('AREA1')**	**> 'AREA1'**
Get a formula output from the called formula. Get BONUS_RATE output value into the RATE variable using the default value 0.0 if the BONUS_RATE output does not exist or was not set.	**RATE = GET_OUTPUT ('BONUS_RATE' ,0.0)**	**RATE <'BONUS_RATE' DEFAULT 0.0**
Get a formula output from a called formula into an array Get the BONUS_RATE output value into the RATES array variable at index position 'BONUS'. Use the default value 0.0 if the BONUS_RATE output doesn't exist or wasn't set.	**RATES['BONUS'] = GET_OUTPUT ('BONUS_RATE' ,0.0)**	**RATES['BONUS'] <'BONUS_RATE' DEFAULT 0.0**

4 Formula Functions

Formula Functions: Explained

Functions manipulate data in different ways and always return a value. They're restricted to simple data types of date, number, and text. A function is specified by its name, return data type, data types, and usage behavior.

The general form of a function is:

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

Operands can be optional or mandatory. They may be repeated any number of times, such as with the **GREATEST** function. The formula compiler resolves functions by matching function calls against function specifications. You can use multiple functions with the same name within a formula provided that they have different return or data types.

Some functions return values that are useful in specific formula types, such as absence management, benefits, or compensation. The more generic functions fall into the categories shown in the following table.

Category	Purpose of Functions
Text	Manipulate text data
Number	Manipulate numeric data
Date	Manipulate dates
Data Conversion	Convert data to a different data type
Message Handling	Return messages

Text Formula Functions

The following formula functions manipulate text data.

CHR(n)

Returns the character having the binary equivalent to a number operand n in the ASCII character set.

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

Compares the values of all the text string operands. It returns the value of the last string in alphabetic order.

INITCAP(expr)

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

Searches **expr1** beginning with its **n**th character for the **m**th occurrence of **expr2** and returns the character position in **expr1** for the first character of this occurrence. If **n** is negative, **INSTR** 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 **INSTR** 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 (**expr1** does not appear **m** times after the **n**th character of **expr1**) the return value is 0.

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 character set, **INSTRB** is equivalent to **INSTR**.

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

Compares the values of all the text string operands. Returns the first string in alphabetic order from among its operands.

LENGTH(expr)

Returns the number of characters in the text string operand **expr**.

LENGTHB(expr)

Returns the length of **expr** in units of bytes.

LOWER(expr)

Converts a text string to lower case.

LPAD(expr, n [,pad])

Returns the text string operand **expr** left-padded to length **n** with the sequence of characters in **pad**. The default value for **pad** is a blank. If **expr** is longer than **n**, then **LPAD** returns 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])

Returns the text string operand **expr** with all the left-most characters that appear in **set** removed. The default for **set** is a blank. If none of the left-most characters of **expr** appear in **set**, then **LTRIM** returns **expr**.

Examples:

```
/* A is set to 'def' */
A = LTRIM ('abcdef','abc')
/* A is set to 'abcdef' */
A = LTRIM ('abcdef','bc')
```

REPLACE(expr, search [,replacement])

Returns the text string operand **expr** with every occurrence of **search** replaced with **replacement**. If you omit **replacement**, it removes all occurrences of **search**. Use **REPLACE** to substitute one string for another or to remove character strings.

Example:

```
/* Set A to 'BLACK and BLUE'. */
A = REPLACE('JACK and JUE', 'J', BL')
```

RPAD(expr, n [,pad])

Returns the text string operand **expr** right-padded to length **n** with the sequence of characters in **pad**. The default value for **pad** is a blank. If **expr** is longer than **n**, then **RPAD** returns 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])

Returns the text string operand **expr** with all the right-most characters that appear in **set** removed. The default value for **set** is a blank. If none of the right-most 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')
```

SUBSTR(expr, m [,n]) or SUBSTRING(expr, m [,n])

SUBSTRING returns a substring of the text string operand **expr** of length **n** characters beginning at the **m**th character. If **n** is negative, **SUBSTR** counts backwards from the end of **expr**. If you omit the **n**, the substring starts from **m** and finishes at the end of **expr**.

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 ...
```

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)

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.

TRIM(expr)

Trims leading and trailing spaces from a character string.

UPPER(expr)

Converts a text string to upper case.

Numeric Formula Functions

The following formula functions manipulate numeric data.

ABS(n)

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)

It returns 17.

FLOOR(n)

Returns the integer part of a numeric operand **n**. If the value of the operand contains information after the decimal point, **FLOOR** discards that information and returns a whole number.

Example:

FLOOR(35.455)

It returns 35.

GREATEST(n, n [, n] ...) or GREATEST_OF(n, n [, n] ...)

Compares all the operands and returns the largest value.

LEAST(n, n [, n] ...) or LEAST_OF(n, n [, n] ...)

Compares all the operands and returns the smallest value.

MOD(m, n)

Returns the remainder from dividing **m** by **n**.

POWER(m, n)

Returns **m** raised to the **n**th power.

ROUND(m [,n])

Rounds **m** to **n** decimal places. The default number of decimal places is 0.

Examples:

ROUND(2.3401, 2)

It returns 2.34.

ROUND (2.3461, 2)

It returns 2.35.

ROUNDUP(m [,n]) or ROUND_UP(m [,n])

Rounds **m** up to **n** decimal places. The default number of places is 0.

Examples:

ROUND_UP(2.3401, 2)

It returns 2.35.

ROUND_UP(2.3400, 2)

It returns 2.34.

TRUNC(n [,m]) or TRUNCATE(n [,m])

Truncates **m** down to **n** decimal places. The default number of places is 0.

Examples:

TRUNC(2.3401, 2)

It returns 2.34.

Date Formula Functions

The following formula functions manipulate date data.

ADD_DAYS(date, n)

Adds **n** whole days to **date**.

Example:

ADD_DAYS ('30-DEC-1990' (date), 6)

It returns 5 JAN 1991.

ADD_MONTHS(date, n)

Adds **n** whole months to **date**.

ADD_YEARS(date, n)

Adds **n** whole years to **date**.

DAYS_BETWEEN(date1, date2)

Returns the number of days between **date1** and **date2**. If **date1** is later than **date2** then the result is a positive number. If **date1** is earlier than **date2** then the result is a negative number.

Returns the number of days between date1 and date2. If date1 is later than date2 then the result is a positive number. If date1 is earlier than date2 then the result is a negative number. Example: DAYS_BETWEEN('1995/06/27 00:00:00' (date), '1995/07/03 00:00:00' (date)) It returns - 5.

Example:

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

It returns - 5.

GREATEST(date, date [, date] ...)

Compares its operands and returns the latest date.

LAST_DAY(date)

Returns the last day of the month containing **date**.

LEAST(date, date [, date] ...)

Compares the operands and returns the earliest **date**.

MONTHS_BETWEEN(date1, date2)

Returns the number of months between **date1** and **date2**. If **date1** is later than **date2**, the result is a positive number. If **date1** is earlier than **date2**, the result is a negative number. The return value has a numeric data type that can contain a fraction if the dates do not differ by a whole number of months.

NEW_TIME(date, zone1, zone2)

Returns the date and time in zone **zone2** when the date and time in zone **zone1** are date.

The arguments **zone1** and **zone2** can be any one of the standard text strings such as:

Time Zone	Description
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 Daylight Time
GMT	Greenwich Mean Time
HST or HDT	Alaska-Hawaii Standard Time or Daylight Time
MST or MDT	Mountain Standard or Daylight Time
NST	Newfoundland Standard Time

Time Zone	Description
PST or PDT	Pacific Standard or Daylight Time
YST or YDT	Yukon Standard or Daylight Time

NEXT_DAY(d, expr)

Returns the first date following **d** of the weekday named by **expr**.

ROUND(date [,format])

Returns the result of rounding **date** according to **format**. The default format is **DDD**, which represents the nearest day.

TRUNC(date [,format])

Returns the result of truncating **date** according to **format**. The default format is **DDD**, which represents a whole day.

Data Conversion Formula Functions

The following formula functions perform data conversions.

DATE_TO_TEXT(date [,format]), TO_CHAR(date [,format]), and TO_TEXT(date [,format])

Converts **date** to a character string with format specified by **format**. The default format is the application canonical format.

NUM_TO_CHAR(n, format)

Converts the number **n** to a character string in the specified format. This function is equivalent to the SQL TO_CHAR function.

TO_CHAR(n) and TO_TEXT(n)

Converts the number **n** to a character string in canonical number format.

TO_DATE (expr [, format])

Converts the character string **expr** in the specified format to a date. If no format is specified then **expr** must be in canonical format.

TO_NUMBER(expr) and TO_NUM(expr)

Converts the character string **expr** to a number. The character string must be in canonical number format. A period is used for the decimal point, such as 1.234. Negative numbers are preceded with a minus, such as -1.234.

Miscellaneous Formula Functions

The following formula functions manipulate messaging data or retrieve values from user-defined tables.

GET_MESG, GET_FND_MESG

GET_MESG(appname, msgname [, token1, value1] [, token2, value2] [, token3, value3] [, token4, value4] [, token5, value5])

GET_FND_MESG(appname, msgname [, token1, value1] [, token2, value2] [, token3, value3] [, token4, value4] [, token5, value5])

Returns an expanded version of the application message specified using **appname**, **msgname**, and up to five pairs of message tokens and their corresponding values.

GET_TABLE_VALUE

GET_TABLE_VALUE(table_name, column_name, row_value [,default_value])

GET_TABLE_VALUE(table_name, column_name, row_value, effective date)

Returns the value of a cell in a user-defined table on the effective date of the session or process. The first three text operands identify the cell. An optional fourth parameter does one of the following, depending on its data type:

- Text: Returns a text default value if no data is found.
- Date: Returns the value of the cell on the specified date.

Example: **GET_TABLE_VALUE('WAGE RATES', 'Wage Rate', Rate_Code, 'DEFAULT')** would return the row_value for Wage Rate or DEFAULT if no row was found.

HR_TRACE(expr)

Outputs a trace message.

Note
It's more efficient to use an application-specific logging function than **HR_TRACE**.

5 Formulas for Payroll

Creating a Proration Formula: Procedure

The predefined proration formula GLB_EARN_PRORATION controls how the payroll calculation prorates an element entry when it encounters an event, such as a change to an element entry value. You can copy and edit a predefined proration formula to customize the calculation, and then select the custom formula as the proration formula for your element.

Creating a Formula

Follow these rules to write a formula:

1. Select the formula type Payroll Run Proration.
2. Add the formula inputs:
    - Element input values
    - Prorate_start (DATE)
    - Prorate_end (DATE)
3. Add the formula outputs for the element input values.

Some predefined legislations supply proration formulas that you can use as the basis for your customized version.

Customizing the Method for Deriving Fixed Rates

You can replace the method for deriving the fixed rate in a formula. For fixed rate calculations, the formula uses an annualized rate conversion. It annualizes the source amount and periodicity, and then converts it to the required periodic rate using the following constants:

- 206 - Annual working days
- 2080 - Annual working hours
- Predefined period, such as a calendar month or week

To replace the method in the formula used to process the element, edit the call for the RATE_CONVERTER as shown in the following sample:

```
CALL_FORMULA( 'RATE_CONVERTER',
l_rate > 'SOURCE_AMOUNT',
l_source_periodicity > 'SOURCE_PERIODICITY',
l_target_periodicity > 'TARGET_PERIODICITY',
'ANNUALIZED RATE CONVERSION' > 'method',
l_actual_amount < 'TARGET_AMOUNT' DEFAULT 0)
```

For example, if your enterprise is based on a 32-hour work week, with an annual number of working hours of 1664, you might create a customized version of the formula. You would edit RATE_CONVERTER call, overriding the ANNUALIZED RATE CONVERSION method with the customized formula.

Related Topics

- Setting Up Element Proration: Procedure

* Prorated Earnings and Deductions: How They're Calculated
* Periodicity Conversion: Explained

Creating a Rate Conversion Formula for a Periodicity: Procedure

Create your own rate conversion formula if you use different values for periodicity than the predefined formulas. The formulas are used when converting rates for proration, hours multiplied by rates calculation of an element run result, and rates based on rate definitions. For example, you might create a formula to specify a different number of working hours to use when converting annual values into hourly rates.

Creating a Formula

Complete the following steps in the Payroll Calculation work area to create a rate conversion formula:

1. Use the Manage Formulas task. Search for formulas with the Rate Conversion formula type.
2. Search for and display the rate conversion formula you want to copy, such as Annualized Rate Conversion or Daily Rate Conversion.
3. Create a new formula with the formula type Rate Conversion.
4. Copy the formula text into your new formula.
5. Edit the periodicity values.

   For example, if you create a rate conversion formula that uses 7.5 hours instead of 8 hours for the number of work hours in a day, you would edit the periodicity as follows:

   ```
   else if (out_periodicity = 'WORKHOUR') then
    l_amt = (l_year_amt /260)/ 7.5
   ```

6. Save, submit, and compile the formula.
7. Use the Manage Elements task to search for the element. Identify the formula used to calculate the element.
8. Use the Manage Formulas task to edit the formula. Edit the rate converter call, replacing the rate conversion formula name with the name of your new formula.

   ```
   CALL_FORMULA( 'RATE_CONVERTER',
   l_rate > 'SOURCE_AMOUNT',
   l_source_periodicity > 'SOURCE_PERIODICITY',
   l_target_periodicity > 'TARGET_PERIODICITY',
   'ANNUALIZED RATE CONVERSION' > 'method', /* replace with the name of new formula*/
   l_actual_amount < 'TARGET_AMOUNT' DEFAULT 0)
   ```

9. Save, submit, and recompile the formula.

Related Topics

* Periodicity Conversion: Explained
* Using Formulas: Explained

Element Input Validation Formula Type

You can use an element input validation formula to validate one or more element entry values. You can also use this formula type to provide a default value for an element entry value, or to calculate entry values based on the user's entries in other entry values.

You select the formula on the Element Summary page in the following fields:

Page Section	Field	Purpose	When the Formula Runs
Element Details, or Element Eligibility	Validation Formula	To validate one or more entry values for the element based on entries in other entry values.	When you save the element entry.
Element Details, or Element Eligibility	Calculation Formula	To provide values for one or more entry values using a calculation that takes input from these or other entry values.	When you save the element entry.
Element Details, or Element Eligibility	Defaulting Formula	To provide default values for one or more entry values.	When you create the element entry.
Input Value	Validation Formula	To validate one entry value independently of others.	When you enter the value.

Important
In all cases, a formula at the element eligibility level overrides an equivalent formula at the element level.

Contexts

The following contexts are available to all formulas of this type:

- LEGISLATIVE_DATA_GROUP_ID
- DATE_EARNED
- EFFECTIVE_DATE

The following contexts are available to formulas at element or element eligibility level only, not to validation formulas at the input value level:

- PERSON_ID

- PAYROLL_RELATIONSHIP_ID
- PAYROLL_TERM_ID
- PAYROLL_ASSIGNMENT_ID
- HR_RELATIONSHIP_ID
- HR_TERM_ID
- HR_ASSIGNMENT_ID

Input Variables

The following input variables are available to formulas of this type.

Formula Usage	Input Variables	Comments
Validation formula at input value level	entry_value	Passes the value to be validated. You must declare the input variable as the appropriate type for the element input value.
Validation formula at element or element eligibility level	Any element input value name that corresponds to an entry value.	Replace spaces in the input value name with underscores in the input variable name. It doesn't matter whether you use uppercase or lowercase for the name.
Defaulting formula	None	Use database items or other logic instead.
Calculation formula	Any element input value name of an entry value.	Replace spaces with underscores. You don't need to provide all of the available entry values.

Return Values

The following return values are available to formulas of this type.

Formula Usage	Return Values	Comments
Validation formula at any level.	formula_status	Must be either 'S' (success) or 'E' (error). Required.

Formula Usage	Return Values	Comments
Validation formula at any level.	formula_ message	Text of message passed to user if the validation fails. Optional.
Defaulting formula	Any element input value name of an entry value.	A return value overrides any default value provided on the input value in the element or element eligibility record.
Calculation formula	Any element input value name of an entry value.	You don't need to return all of the available entry values. You can return the entry values that were passed in as input variables, or other entry values.

Sample Formula

This section contains the following sample formulas:

- Validation formula at input value level
- Validation formula at element or element eligibility level
- Calculation formula at element or element eligibility level
- Defaulting formula at element or element eligibility level

Validation formula at input value level:

```
inputs are entry_value(date)
if(entry_value = '01-APR-2008' (date)) then
(
 formula_message = 'Valid date'
 formula_status = 'S'
)
else
(
 formula_message = 'Invalid date'
 formula_status = 'E'
)
return formula_message, formula_status
```

Validation formula at element or element eligibility level:

```
inputs are hours_worked, rate, earning_date(date), comment(text)
if(hours_worked > 80) then
(
 formula_message = 'You are within the working limit.'
 formula_status = 'S'
)
else
(
 formula_message = 'You have worked too many hours.'
 formula_status = 'E'
)
```

```
return formula_message, formula_status
```

Calculation formula at element or element eligibility level:

```
inputs are hours_worked, rate, comment(text)
if(hours_worked > 80) then
(
 rate = rate * 1.2
 comment = 'Your rate has been increased'
)
return rate, comment
```

Defaulting formula at element or element eligibility level:

```
if(CATEGORY = 'S') then
(
 rate = 20
)
else
(
 rate = 30
)
rate_code = 'B'
return rate, rate_code
```

User Table Validation Formula Type

The User Table Validation formula type validates entries in user-defined tables. Select the formula in the **Formula** field for user-defined columns when you create or edit user-defined tables.

For example, you can use this formula type to ensure that entries are:

- Between a specified range
- Not a negative amount

Contexts

The EFFECTIVE_DATE (text) context is used for formulas of this type.

Input Variables

There must be one input variable and it must be called ENTRY_VALUE. The data type is text.

Return Values

The following return values are available to formulas of this type:

Return Value	Data Type	Required	Description
FORMULA_ MESSAGE	Text	N	Returns a text message for either or both statuses. The message is displayed on the Create User-Defined Table: User-Defined Table Values page.

Return Value	Data Type	Required	Description
FORMULA_STATUS	Text	Y	Returns the value S (success) or E (error).

Sample Formula

This formula 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
```

Flow Schedule Formula Type

The Flow Schedule formula controls when the application submits the current flow and how often it submits future instances of the flow. You create scheduling formulas on the Manage Fast Formulas page when the predefined formulas don't cover your requirements.

For example, you might create a formula that loads time card batches daily, and increases to four times a day at the end of a payroll period when workers typically submit their time cards. You might create a formula that schedules the frequency with which an extract process checks for new starter details.

Follow these tips when creating or updating a scheduling formula:

- Specify a meaningful name to assist the person selecting the formula.
- Review the formula to ensure it doesn't contain negative numbers that might produce an error condition, such as running a process continually.
- After updating the formula, cancel any scheduled flows that use the formula. Resubmit the flow to apply the updated definition.

Contexts

The SCHEDULED_DATE (scheduled date) context is available to formula of this type.

Database Items

The following database items are available to formulas of this type:

Database Item	Data Type	Description
FF_ADD_DAYS	Date	Function to add days to a date.
FF_ADD_MONTHS	Date	Function to add months to a date.
NEXT_ SCHEDULED_ DATE	Date	Calculated value for the date to schedule the next flow.
SCHEDULED_DATE	Date	Date used to schedule the flow.

Input Variables

The following input variables are available to formulas of this type:

Input	Data Type	Required	Description
SCHEDULED_ DATE(DATE)	Date	Y	Date on which to schedule the flow. The date is passed to the formula when it calculates the next date to schedule the flow.

Return Values

Use predefined names for return variables. The following return value is available to formulas of this type:

Return Values	Data Type	Required	Description
NEXT_ SCHEDULED_ DATE	Date	Y	The date calculated by the formula to schedule the next flow.

Sample Formula

This predefined formula schedules a flow so that it's submitted weekly from the date the flow owner initially submitted it.

```
/*****************************************************************************
FORMULA NAME: Weekly
FORMULA TYPE: Flow Schedule
```

```
DESCRIPTION: Formula to return a date time.
 Returns NEXT_SCHEDULED_DATE;
Formula Results :
 NEXT_SCHEDULED_DATE This is a date time value with yyyy-MM-dd HH:mm:ss format.
***************************************************************************/

/* Inputs */
INPUTS ARE SUBMISSION_DATE(DATE), SCHEDULED_DATE(DATE)

/* Calculations */
NEXT_SCHEDULED_DATE = ADD_DAYS(SCHEDULED_DATE,7)

/* Returns */
RETURN NEXT_SCHEDULED_DATE

/* End Formula Text */
```

You can calculate units smaller than a day by replacing the calculation portion of the formula text using a decimal or a fraction. This table shows examples of submitting a flow several times a day.

Flow Submission	Formula Text for Calculation
Twice a day	`NEXT_SCHEDULED_DATE =ADD_DAYS(SCHEDULED_DATE,0.5)`
Hourly	`NEXT_SCHEDULED_DATE =ADD_DAYS(SCHEDULED_DATE,1/24)`

Related Topics

- Writing a Fast Formula Using Formula Text: Worked Example
- Date Formula Functions
- Using Formula Components: Explained
- Creating a Daily Schedule for a Flow that Skips Weekends: Worked Example

Batch Loader Formula Type

Overview

The Batch Loader formula type controls how the application retrieves information when loading a batch from a file. You create batch loader formulas on the Manage Fast Formulas page when the file you want to upload requires transformation.

Examples of why you might create a batch loader formula include:

- To assign the position of attributes in the file because they are not in the expected sequence.
- To convert an attribute in the file to another attribute that you derive using value sets.
- To use a single formula for separate batch loader tasks.

Contexts

The following context is available to all formulas of the Batch Loader type:

- EFFECTIVE_DATE (text)

Database Items

The Batch Loader formula type does not support database items. You can use database items in your batch loader formulas by calling a formula within your formula.

To retrieve information when a database item isn't available, use the GET_VALUE_SET function.

Restriction

When using the GET_VALUE_SET function, ensure that the Value Attributes Table Alias field for the value set has no value. The function will fail when an alias is provided.

Input Variables

The following input variables are available for all formulas of Batch Loader formula type. Additional variables may be available depending on the selected task and task action.

Input	Data Type	Required	Description
OPERATION	Text	Yes	Available operations are: • FILETYPE • DELIMITER • MAP • READ
LINESEQUENCE	Number	No	
LINEREPEATNO	Number	No	
LINENO	Number	Yes	
POSITION1	Text	Yes	
POSITION2	Text	Yes	
POSITION3-POSITION30	Text	No	

Input	Data Type	Required	Description

Return Values

The return values for batch loader formulas vary based on the business object and task action. Return values are the same as the parameter names with space characters replaced by underscore characters.

Sample Formula for Batch Loader File Transformation

The transformation formula in this example specifies the position of attributes because the attributes in the flat file are in a different sequence than the sequence expected in payroll batch loader workbook. The formula also converts Person Number in the flat file into Assignment Number.

The formula's return values are the same as the parameter name with space characters replaced by underscore characters.

The following snippet in the formula assigns the attributes to the required position and performs the conversion using values sets to derive the necessary values:

```
/*Task Action Related Outputs */

 PAYROLL=POSITION2
 ASSIGNMENT_NUMBER=GET_VALUE_SET('SAMPLE_GET_ASG_NUM','|=PERSON_NUMBER='''||POSITION3||'''')
 Effective_As_Of_Date=POSITION1
```

Tip

To debug value sets, create a BI report with the following query as a data model to return the required data: **SELECT pay_ff_functions.gvs('SAMPLE_GET_ASG_NUM','|=PERSON_ID=100000012092216') value FROM dual;**

The complete formula for this example is as follows:

```
/******************************************************************************
FORMULA NAME: Transform Attribute - Payroll Relationship
FORMULA TYPE: Batch Loader
Sample Input File Format:
Effective As Of Date|Payroll|Person Number
*******************************************************************************
*/

/* Inputs */
INPUTS ARE OPERATION (text), LINESEQUENCE (number), LINEREPEATNO (number),
 POSITION1 (text), POSITION2 (text), POSITION3 (text)

DEFAULT FOR LINENO IS 1
DEFAULT FOR LINEREPEATNO IS 1
DEFAULT FOR LINESEQUENCE IS 1
DEFAULT FOR POSITION1 IS 'NO DATA'
DEFAULT FOR POSITION2 IS 'NO DATA'
DEFAULT FOR POSITION3 IS 'NO DATA'


/* Calculations */
IF OPERATION='FILETYPE' THEN
 OUTPUTVALUE='DELIMITED'
ELSE IF OPERATION='DELIMITER' THEN
 OUTPUTVALUE='|'
ELSE IF OPERATION='READ' THEN
 OUTPUTVALUE='NONE'
```

```
ELSE IF OPERATION='MAP' THEN
 (
 /*Batch Related Outputs*/
 TASK='Payroll Relationship'
 TASKACTION='Add Payroll'

 /*Task Action Related Outputs */
 PAYROLL=POSITION2
 ASSIGNMENT_NUMBER=GET_VALUE_SET('SAMPLE_GET_ASG_NUM','|=PERSON_NUMBER='''||POSITION3||'''')
 Effective_As_Of_Date=POSITION1

 )
ELSE
 OUTPUTVALUE='NONE'

/* Returns */
IF OPERATION='MAP' THEN
 RETURN TASK,TASKACTION,LINESEQUENCE,PAYROLL,ASSIGNMENT_NUMBER,Effective_As_Of_Date
ELSE
 RETURN OUTPUTVALUE

/* End Formula Text */
```

Related Topics

- File Format and Data Transformation for Payroll Batch Uploads: Critical Choices

Sample Batch Loader Formula for Multiple Tasks

The formula in this example uses two separate tasks to create bank and also add payroll to a payroll relationship. The formula also converts Person Number in the flat file into Assignment Number.

The formula for this example is as follows:

```
/*************************************************************************************
FORMULA NAME: Create Bank and Payroll Relationship
FORMULA TYPE: Batch Loader
Sample Input File Format:
Effective As Of Date|Payroll|Assignment Number
*************************************************************************************
*/

/* Inputs */
INPUTS ARE OPERATION (text), LINEREPEATNO (number), POSITION1 (text),
 POSITION2 (text), POSITION3 (text), POSITION4 (text),
 POSITION5 (text),

DEFAULT FOR LINENO IS 1
DEFAULT FOR LINEREPEATNO IS 1
DEFAULT FOR POSITION1 IS 'NO DATA'
DEFAULT FOR POSITION2 IS 'NO DATA'
DEFAULT FOR POSITION3 IS 'NO DATA'
DEFAULT FOR POSITION4 IS 'NO DATA'
DEFAULT FOR POSITION5 IS 'NO DATA'

/* Calculations */
IF OPERATION='FILETYPE' THEN
 OUTPUTVALUE='DELIMITED'
ELSE IF OPERATION='DELIMITER' THEN
 OUTPUTVALUE='|'
ELSE IF OPERATION='READ' THEN
```

```
 OUTPUTVALUE='NONE'
ELSE IF OPERATION='MAP' THEN
 (
 /*Batch Related Outputs*/
 IF POSITION1 = 'Payroll Relationship' AND POSITION2 = 'Add Payroll' THEN(
 TASK='Payroll Relationship'
 TASKACTION='Add Payroll'
 LINESEQUENCE=LINENO

 /*Task Action Related Outputs */
 PAYROLL=POSITION4
 ASSIGNMENT_NUMBER=POSITION5
 Effective_As_Of_Date=POSITION3
 )
 IF POSITION1 = 'Bank and Branch' AND POSITION2 = 'Create Bank' THEN(
 TASK='Bank and Branch'
 TASKACTION='Create Bank'
 LINESEQUENCE=LINENO

 /*Task Action Related Outputs */
 Bank_Name=POSITION3
 )

 )
ELSE
 OUTPUTVALUE='NONE'
/* Returns */
IF OPERATION='MAP' THEN

 IF POSITION1 = 'Payroll Relationship' AND POSITION2 = 'Add Payroll' THEN (
 RETURN TASK,TASKACTION,LINESEQUENCE,PAYROLL,ASSIGNMENT_NUMBER,Effective_As_Of_Date
 )

 IF POSITION1 = 'Bank and Branch' AND POSITION2 = 'Create Bank' THEN (
 RETURN TASK,TASKACTION,LINESEQUENCE,Bank_Name
 )
ELSE
 RETURN OUTPUTVALUE

/* End Formula Text */
```

6 Formulas for Absence Management

Qualification Absence Plan Rules: Points to Consider

Configure the following rules when you create an absence qualification plan in accordance with the leave policy of your enterprise:

- Plan term
- Plan eligibility
- Enrollment and termination
- Waiting period
- Payments

Plan Term

A qualification plan term is an assessment period for which the Evaluate Absence process calculates entitlements for the total absent time recorded in that period.

When you create an absence qualification plan, you must select the type of plan term.

Example: You can limit the duration of the plan term to the duration of the absence.

Plan Eligibility

Associate an eligibility profile with the qualification plan to determine the set of workers who are eligible to record an absence that belongs to that plan.

Follow these steps to associate an eligibility profile with an absence plan:

1. Create the eligibility profile using the Manage Eligibility Profiles task in the Setup and Maintenance work area.
2. Associate the eligibility profile with the absence plan using the Manage Absence Plans task.

If you want all employees to be eligible for the absence plan, then do not add an eligibility profile. If you associate multiple absence plans with an absence type, the worker must be eligible for at least one absence plan to record an absence of that type.

Enrollment and Termination

Decide when to enroll workers in the qualification plan.

- Typically, you enroll workers in the plan when a worker or an administrator schedules an absence using an absence type associated with a qualification plan.
- Use a formula if you must consider other aspects or rules that determine when to enroll workers in the plan.

Decide whether ongoing payments under this plan must continue if a worker is:

- Terminated
- Not terminated, but loses eligibility for the plan

Payments

Use an entitlement band matrix to determine the payment percentages that apply for specific time periods during an absence.

Decide how you want to calculate the payment rate of a single unit of absence. You can use a rate definition to include the calculation rules, or use a formula.

Example: You want workers who have completed a particular tenure to receive specific percentage of pay for a specific absence period.

The following table shows a sample scenario:

Length of Service	Payment Rule
5 to 10 years	75 percent up to 10 absent days.
10 to 20 years	75 percent up to 20 absent days.

Decide how you want to calculate the payment rate of a single unit of absence. You can use a rate definition to include the calculation rules, or use a formula.

Related Topics

- Creating a Maternity Absence Qualification Plan: Worked Example

Accrual Absence Plan Rules: Points to Consider

Configure the following rules when you create an absence accrual plan in accordance with the leave policy of your enterprise:

- Accrual term and frequency
- Plan eligibility
- Enrollment and termination
- Waiting period and vesting period
- Plan limits
- Payments
- Adjustments

Accrual Term and Frequency

An accrual term is a period of time during which workers accrue time. You must specify the type of accrual term to use for the plan.

Example: You can define one of these term types:

- An accrual term of one calendar year that restarts on January 1
- An accrual term that starts on the worker's annual hire date and restarts on every anniversary

Use one of these methods to determine how workers accrue time in an accrual term:

- Award time in increments, also known as accrual periods, throughout an accrual term. Use the worker's pay periods or define your own repeating periods to determine the number of accrual periods in a term.

  Example: Workers who are paid monthly have 12 accrual periods in a year. The accrual amounts for each accrual period are automatically calculated based on the defined accrual rate.

- Award time at the beginning of each accrual term.

Plan Eligibility

Associate an eligibility profile with the accrual plan to determine the set of workers who can enroll in that plan.

Follow these steps to associate an eligibility profile with an accrual plan:

1. Create the eligibility profile using the Manage Eligibility Profiles task in the Absence Administration work area.
2. Associate the eligibility profile with the absence plan using the Manage Absence Plans task.

If you want all employees to be eligible for the absence plan, then do not add an eligibility profile. If you associate multiple absence plans with an absence type, the worker must be eligible for at least one absence plan to record an absence of that type.

Enrollment and Termination

Decide when to enroll workers in the accrual plan.

- You can enroll workers in the plan when the new-hire event or transfer event occurs.
- Use a formula if you want to consider other aspects or rules to determine when to enroll workers.

Choose when to disenroll a terminated worker from the plan.

Choose how you want to deal with negative and positive balances in situations where:

- Only plan enrollment ends
- Both plan enrollment and employment ends

Waiting Period and Vesting Period

Define a waiting period if you want newly enrolled workers to accrue time only after a specific amount of time elapses after the enrollment date.

Define a vesting period if you want newly enrolled workers to accrue time, but not use it until after a specific amount of time.

Plan Limits

Configure the following plan limits:

Plan Limit	Description
Accrual rate	Determines how much time a worker can accrue in an accrual term.
Carryover limit	Determines the maximum time that workers can carry over to the next term.
Ceiling	Determines the maximum leave time that workers can accrue.

Use an accrual band matrix to build criteria using various factors, such as length of service, to determine workers who qualify for specific plan limits. Alternatively, you can use a formula to determine each plan limit.

Payments

Decide how you want to calculate payment of accrual balances for the following scenarios:

- When workers must be paid a different rate during the absence period
- When a part of the accrual balance must be disbursed to workers as cash
- When the cost of accrual balance must be calculated to determine employer liability
- When the accrual balance must be paid to workers when their plan participation ends

Adjustments

You can enable the following types of adjustments that HR specialists can make during maintenance of absence records and entitlements:

- Discretionary disbursements of accrual balance
- Accrual balance transfers across plans
- Other adjustments

Related Topics

- Creating a Vacation Absence Accrual Plan: Worked Example

Formulas for Accrual Plan Rules

Use the Manage Absence Plan page to apply delivered accrual plan rules in the plan. However, if you want to define other special rules to suit your requirement, you can write your own formulas.

Formulas for Accrual Plan Rules

The following table lists the aspects of an accrual plan for which you can write a formula and identifies the formula type for each.

Rule	Description	Formula Type to Use
Enrollment Start	Date when eligible workers enroll in the plan.	Global Absence Plan Enrollment Start
Enrollment End	Date when workers disenroll from the plan.	Global Absence Plan Enrollment End
Plan Duration Conversion	Method to calculate the absence duration differently. Example: You might have a requirement to consider only whole working days in a vacation absence to update the accrual balance. In such cases, you define logic in a formula to convert the absence duration to a value that excludes partial days.	Global Absence Plan Duration

Rule	Description	Formula Type to Use
Anniversary Event Rule	Method to determine the employment anniversary date on which you want the accrual plan to restart.	Global Absence Plan Period Anniversary Event Date
Accrual Vesting	A period during which workers accrue time, but cannot use it.	Global Absence Vesting
Accrual Proration	Method to calculate the time workers accrue if they enroll in the middle of an accrual period.	Global Absence Proration
Ceiling	The maximum time that a worker can accrue.	Global Absence Ceiling
Carryover	The maximum unused time that a worker can transfer to the next accrual term.	Global Absence Carryover
Accrual Band Formula	Range of eligibility criteria that identify how much paid time eligible workers accrue over the course of an accrual term. The criteria may be years of service, grades, hours worked, or some other factor that you can define.	Global Absence Accrual
Absence Payment Rate	Method to calculate payment during absence period.	Global Absence Plan Use Base Rate
Disbursement Rate	Method to calculate payment when paying out part of the accrual balance.	Global Absence Plan Use Cash Out Rate
Final Balance Payment Rate	Method to calculate payment of accruals when plan participation ends.	Global Absence Plan Use Payout Rate
Liability Booking Rate	Method to calculate cost of accrual balance to determine employer liability.	Global Absence Plan Use Leave Liability Rate

Formulas for Qualification Plan Rules

Use the Manage Absence Plan page to incorporate qualification plan rules. However, if you want to define other special rules to suit your requirement, you can write your own formulas.

Formulas for Qualification Plan Rules

The following table lists the aspects of a qualification plan for which you can write a formula and identifies the formula type for each.

Rule	Description	Formula Type to Use
Rolling Backward Start Rule	When the rolling backward plan term starts. A rolling backward term is a specific time period that precedes the absence start date.	Global Absence Roll Backward Start Date
Enrollment Start	Date when eligible workers enroll in the plan.	Global Absence Plan Enrollment Start
Enrollment End	Date when workers disenroll from the plan.	Global Absence Plan Enrollment End
Plan Duration Conversion	Method to calculate the absence duration differently. Example: You want to consider only whole working days in a sickness absence in the entitlement calculation. In such cases, you define logic in a formula to convert the absence duration to a value that excludes partial working days.	Global Absence Plan Duration Conversion
Entitlement Definition	Determines payment percentages to apply during the absence period.	Global Absence Entitlement
Qualification Band	A level that determines the payment that workers receive for a specific number of days during a long leave of absence based on their length of service.	Global Absence Entitlement

Formulas for Absence Type Rules

Use the Manage Absence Types pages to define absence type rules. However, if you want to define other special rules to suit your requirement, you can write your own formulas.

Formulas for Absence Types

The following table lists the aspects of an absence type for which you can write a formula and identifies the formula type for each.

Rule	Description	Formula Type to Use
Duration Conversion	Method to convert the absence duration to other units of measure. For example, your workers' work schedules are in work hours, but you want to display the duration in work days.	Global Absence Type Duration You can use the formula to convert absence duration values that are in work days or work hours only.
Validation	Rules in addition to the ones that you can define on the Manage Absence Types pages to check the validity of the absence.	Global Absence Entry Validation

7 Formulas for Compensation Plans

Compensation Currency Selection Formula Type

The Compensation Currency Selection formula determines the currency associated with a workforce compensation component.

You select the formula on the Configure Compensation Components page.

Contexts

The following contexts are available to formulas of this type:

- DATE_EARNED
- EFFECTIVE_DATE
- END_DATE
- START_DATE
- HR_ASSIGNMENT_ID
- HR_TERM_ID
- JOB_ID
- LEGISLATIVE_DATA_GROUP_ID
- COMPENSATION_RECORD_TYPE
- ORGANIZATION_ID
- PAYROLL_ASSIGNMENT_ID
- PAYROLL_RELATIONSHIP_ID
- PAYROLL_TERM_ID
- PERSON_ID

Database Items

Database items related to Person, Assignment, Salary, Element Entries, Compensation Record, and From and End Dates are available to formulas of this type.

Input Variables

The following input variables are available to formulas of this type.

Input	Data Type	Required	Description
CMP_IV_PLAN_ID	Number	Y	Plan ID
CMP_ IV_ ASSIGNMENT_ID	Number	Y	Assignment ID
CMP_ IV_PERIOD_ID	Number	Y	Period ID

Input	Data Type	Required	Description
CMP_ IV_ COMPONENT_ID	Number	Y	Component ID
CMP_ IV_ PLAN_ START_DATE	Date	Y	Plan Start Date
CMP_ IV_ PLAN_END_DATE	Date	Y	Plan End Date
CMP_ IV_ PLAN_ EXTRACTION_ DATE	Date	Y	Plan Extraction Date
CMP_ IV_ PLAN_ ELIG_DATE	Date	Y	Plan Eligibility Date
CMP_ IV_ PERFORMANCE_ EFF_DATE	Date	Y	Performance Effective Date
CMP_ IV_ PROMOTION_ EFF_DATE	Date	Y	Promotion Effective Date
CMP_ IV_ XCHG_ RATE_DATE	Date	Y	Currency Conversion Date
CMP_ IV_ ASSIGNMENT_ID	Number	Y	Assignment ID
CMP_ IV_PERSON_ID	Number	Y	Worker ID

Return Values

The following return variables are available to formulas of this type.

Return Value	Data Type	Required	Description
L_CURR_CODE	Char	N	Currency code from the formula

Sample Formula

This sample formula determines the currency for a plan based on the component ID.

```
/*******************************************************************
FORMULA NAME : Compensation Currency Selection Formula
FORMULA TYPE : Compensation Currency Selection
DESCRIPTION : It returns the currency code based on
 component_id.
*******************************************************************/

/*=========== INPUT VALUES DEFAULTS BEGIN =====================*/
INPUTS ARE CMP_IV_ASSIGNMENT_ID (number), CMP_IV_PLAN_ID (number), CMP_IV_PERIOD_ID (number),
 CMP_IV_COMPONENT_ID (number)
/*=========== INPUT VALUES DEFAULTS ENDS======================*/

/*================ FORMULA SECTION BEGIN =======================*/

 DEFAULT FOR CMP_IV_COMPONENT_ID IS 0

 l_curr_code = 'XXX'

 IF (CMP_IV_COMPONENT_ID = 489) THEN
 (
 l_curr_code = 'USD'
 )
 ELSE IF (CMP_IV_COMPONENT_ID = 490) THEN
 (
 l_curr_code = 'GBP'
 )

 RETURN l_curr_code

/*================ FORMULA SECTION END =======================*/
```

Compensation Default and Override Formula Type

The Compensation Default and Override formula determines the default values populated in a column for a workforce compensation plan. When you configure the worksheet display for a column in the Configure Column Properties dialog box, Default Values tab, you can select this formula.

The following predefined formulas are available for the eligible salary column for this formula type.

Note
Use these formulas as samples for testing purposes only. Copy and create your own version of a formula for use in your own compensation plans. Modifying the sample formula might provide unexpected results upon upgrade.

Formula	Description
CMP_ ELIGIBLE_ SALARY_ PRORATION_ DAILY_AVERAGE	Eligible salary calculated by averaging daily salary. Accounts for number of days that a salary is in effect during the workforce compensation cycle evaluation period.

Formula	Description
CMP_ ELIGIBLE_ SALARY_ PRORATION_ MONTH_ END_AVERAGE	Eligible salary calculated by averaging salary on the last day of each month in the workforce compensation cycle evaluation period. Uses salary on the last day of the evaluation period for midmonth evaluation end dates.
CMP_ ELIGIBLE_ SALARY_ PRORATION_ DAILY_ AVERAGE_ NINETY_ DAY_MIN	Eligible salary calculated by averaging daily salary. Accounts for number of days that a salary is in effect during the workforce compensation cycle evaluation period. Returns zero for workers who worked fewer than 90 days.
CMP_ ELIGIBLE_ SALARY_ PRORATION_ DAILY_ AVERAGE_ USING_FTE	Eligible salary calculated by averaging daily salary adjusted for part-time workers. Accounts for number days that a salary is in effect and FTE during the workforce compensation cycle evaluation period.
CMP_ ELIGIBLE_ SALARY_ PRORATION_ DAILY_ AVERAGE_ FOR_JOBS	Eligible salary calculated by averaging salary for the number of days a worker holds a specific job code on the assignment. Accounts for the number of days that a salary is in effect during the workforce compensation cycle evaluation period.

Contexts

The following contexts are available to formulas of this type:

- DATE_EARNED
- EFFECTIVE_DATE
- END_DATE
- START_DATE
- HR_ASSIGNMENT_ID
- HR_TERM_ID
- JOB_ID
- LEGISLATIVE_DATA_GROUP_ID
- COMPENSATION_RECORD_TYPE
- ORGANIZATION_ID
- PAYROLL_ASSIGNMENT_ID
- PAYROLL_RELATIONSHIP_ID
- PAYROLL_TERM_ID

- PERSON_ID

Database Items

Database items related to Person, Assignment, Salary, Element Entries, Compensation Record, and From and End Dates are available to formulas of this type.

Input Variables

The following input variables are available to formulas of this type.

Input	Data Type	Required	Description
CMP_IV_PLAN_ID	Number	Y	Unique numeric identifier for the workforce compensation plan
CMP_ IV_PERIOD_ID	Number	Y	Unique numeric identifier for the fiscal calendar period
CMP_ IV_ COMPONENT_ID	Number	Y	Unique numeric identifier for the workforce compensation plan component
CMP_ IV_ITEM_NAME	Char	Y	Name for the workforce compensation plan item
CMP_ IV_PERSON_ID	Number	Y	Unique numeric identifier for the worker associated with the workforce compensation plan
CMP_ IV_ PLAN_ START_DATE	Date	Y	Date on which the workforce compensation plan becomes active
CMP_ IV_ PLAN_END_DATE	Date	Y	Date on which the workforce compensation plan becomes inactive
CMP_ IV_ PLAN_ ELIG_DATE	Date	Y	Date on which the workforce compensation plan becomes eligible

Input	Data Type	Required	Description
CMP_ IV_ PERFORMANCE_ EFF_DATE	Date	Y	Date to use for compensation performance ratings
CMP_ IV_ PROMOTION_ EFF_DATE	Date	Y	Date on which job, grade, and position changes take effect
CMP_ IV_ XCHG_ RATE_DATE	Date	Y	Date on which the application obtains conversion rates from the GL daily rates table
CMP_ IV_ ASSIGNMENT_ID	Number	Y	Date to use for assignments

Return Values

The following return variables are available to formulas of this type.

Return Value	Data Type	Required	Description
L_ DEFAULT_VALUE	Number/ Char/Date	Y	Default value from the formula. The date should be in yyyy/mm/dd format
L_DATA_TYPE	Char	Y	Data type of the column

Sample Formula

This sample formula determines the value of a column based on its item name.

```
/***********************************************************************
FORMULA NAME : Compensation Default and Override Formula
FORMULA TYPE : Compensation Default and Override
DESCRIPTION : Defaults the value of a column based on its
 item_name.
***********************************************************************/

/*=========== INPUT VALUES DEFAULTS BEGIN =====================*/
INPUTS ARE CMP_IV_PLAN_ID (number), CMP_IV_PERIOD_ID (number), CMP_IV_COMPONENT_ID (number),
 CMP_IV_ITEM_NAME (text)
/*=========== INPUT VALUES DEFAULTS ENDS======================*/

/*================ FORMULA SECTION BEGIN =======================*/

 DEFAULT FOR CMP_IV_ITEM_NAME IS 'YYYYYYY'
 L_DEFAULT_VALUE = to_char(0)

IF (CMP_IV_ITEM_NAME = 'AmountComp1') THEN
 (
 L_DEFAULT_VALUE = to_char(3333)
 )
 ELSE IF (CMP_IV_ITEM_NAME = 'AmountComp2') THEN
 (
 L_DEFAULT_VALUE = to_char(7777)
 )
 ELSE
 (
 L_DEFAULT_VALUE = to_char(-999)
 )

 RETURN L_DEFAULT_VALUE

/*================ FORMULA SECTION END =======================*/
```

Compensation Hierarchy Determination Formula Type

The Compensation Hierarchy Determination formula determines the hierarchy for an associated workforce compensation plan.

You select the formula on the Configure Plan Details page.

Contexts

The following contexts are available to formulas of this type:

- DATE_EARNED
- EFFECTIVE_DATE
- HR_ASSIGNMENT_ID
- END_DATE
- START_DATE
- HR_TERM_ID
- JOB_ID
- LEGISLATIVE_DATA_GROUP_ID
- COMPENSATION_RECORD_TYPE
- ORGANIZATION_ID
- PAYROLL_ASSIGNMENT_ID
- PAYROLL_RELATIONSHIP_ID
- PAYROLL_TERM_ID
- PERSON_ID

Database Items

Database items related to Person, Assignment, Salary, Element Entries, Compensation Record, and From and End Dates are available to formulas of this type.

Input Variables

The following input variables are available to formulas of this type.

Input	Data Type	Required	Description
CMP_ IV_ ASSIGNMENT_ID	Number	Y	Assignment ID
CMP_IV_PLAN_ID	Number	Y	Plan ID
CMP_ IV_PERIOD_ID	Number	Y	Period ID
CMP_ IV_ COMPONENT_ID	Number	Y	Component ID
CMP_ IV_PERSON_ID	Number	Y	Worker ID
CMP_ IV_ PLAN_ START_DATE	Date	Y	Plan Start Date
CMP_ IV_ PLAN_END_DATE	Date	Y	Plan End Date
CMP_ IV_ PLAN_ EXTRACTION_ DATE	Date	Y	Plan Extraction Date
CMP_ IV_ PLAN_ ELIG_DATE	Date	Y	Plan Eligibility Date
CMP_ IV_ PERFORMANCE_ EFF_DATE	Date	Y	Performance Effective Date
CMP_ IV_ PROMOTION_ EFF_DATE	Date	Y	Promotion Effective Date
CMP_ IV_ XCHG_ RATE_DATE	Date	Y	Currency Conversion Date

Input	Data Type	Required	Description

Return Values

The following return variables are available to formulas of this type.

Return Value	Data Type	Required	Description
L_PERSON_ID	Number	Y	Person ID of manager
L_ ASSIGNMENT_ID	Number	Y	Assignment ID of manager

Or

Return Value	Data Type	Required	Description
L_ PERSON_NUMBER	Number	Y	Person number of manager

You receive the following error if the formula returns an invalid PERSON_NUMBER and the application can't obtain the ASSIGNMENT_ID:

Formula passed in an invalid person number <15465857>. Assignment ID could not be obtained.

Sample Formula

This sample formula determines the manager of a person when the assignment_id is passed.

```
/**************************************************************************
FORMULA NAME : Compensation Hierarchy Determination Formula
FORMULA TYPE : Compensation Hierarchy Determination
DESCRIPTION : Hierarchy determination fast formula which is based on assignment_id
*************************************************************************/

/*=========== INPUT VALUES DEFAULTS BEGIN =====================*/
INPUTS ARE CMP_IV_ASSIGNMENT_ID (number), CMP_IV_PLAN_ID (number), CMP_IV_PERIOD_ID (number)
/*=========== INPUT VALUES DEFAULTS ENDS======================*/

/*================ FORMULA SECTION BEGIN =======================*/

 DEFAULT FOR CMP_IV_ASSIGNMENT_ID IS 0

 L_PERSON_ID = '0'
 L_ASSIGNMENT_ID = '0'

 if ( CMP_IV_ASSIGNMENT_ID = 100000008154060 ) THEN
 (
 L_PERSON_ID = to_char(-999) //-999 indicates top level
//Manager.
```

```
 L_ASSIGNMENT_ID = to_char(-999)
 )
 ELSE
 (
 L_PERSON_ID = to_char(100000008153756)
 L_ASSIGNMENT_ID = to_char(100000008154060)
 )

 RETURN L_PERSON_ID , L_ASSIGNMENT_ID

/*================ FORMULA SECTION END =======================*/
```

Compensation Person Selection Formula Type

The Compensation Person Selection formula determines the person selected for an associated workforce compensation plan.

You select the formula when you run the Start Workforce Compensation Cycle process.

Contexts

The following contexts are available to formulas of this type:

- DATE_EARNED
- EFFECTIVE_DATE
- END_DATE
- START_DATE
- HR_ASSIGNMENT_ID
- HR_TERM_ID
- JOB_ID
- LEGISLATIVE_DATA_GROUP_ID
- COMPENSATION_RECORD_TYPE
- ORGANIZATION_ID
- PAYROLL_ASSIGNMENT_ID
- PAYROLL_RELATIONSHIP_ID
- PAYROLL_TERM_ID
- PERSON_ID

Database Items

Database items related to Person, Assignment, Salary, Element Entries, Compensation Record, and From and End Dates are available to formulas of this type.

Input Variables

The following input variables are available to formulas of this type.

Input	Data Type	Required	Description
CMP_IV_PLAN_ID	Number	Y	Plan ID

Input	Data Type	Required	Description
CMP_ IV_PERIOD_ID	Number	Y	Period ID
CMP_ IV_ PLAN_ START_DATE	Date	Y	Plan Start Date
CMP_ IV_ PLAN_END_DATE	Date	Y	Plan End Date
CMP_ IV_ PLAN_ ELIG_DATE	Date	Y	Plan Eligibility Date
CMP_ IV_ PERFORMANCE_ EFF_DATE	Date	Y	Performance Effective Date
CMP_ IV_ PROMOTION_ EFF_DATE	Date	Y	Promotion Effective Date
CMP_ IV_ XCHG_ RATE_DATE	Date	Y	Currency Conversion Date
CMP_ IV_ ASSIGNMENT_ID	Number	Y	Assignment ID
CMP_ IV_PERSON_ID	Number	Y	Worker ID

Return Values

The following return variables are available to formulas of this type.

Return Value	Data Type	Required	Description
L_SELECTED	Char	N	Y or N

Sample Formula

This sample formula determines if a person is selected for a workforce compensation plan based on their assignment_id.

```
/*************************************************************************
FORMULA NAME : Compensation Selection Formula
FORMULA TYPE : Compensation Person Selection
DESCRIPTION : Assignment_id based selection fast formula
**********************************************************************/

/*=========== INPUT VALUES DEFAULTS BEGIN =====================*/
INPUTS ARE CMP_IV_ASSIGNMENT_ID (number), CMP_IV_PLAN_ID (number)
/*=========== INPUT VALUES DEFAULTS ENDS======================*/

/*================ FORMULA SECTION BEGIN =======================*/

 DEFAULT FOR CMP_IV_ASSIGNMENT_ID IS 0

 l_selected = 'Y'

 /* 100000008154095 - Ariel.Aimar@oracle.com - GBI data*/

 if (CMP_IV_ASSIGNMENT_ID = 100000008154095) THEN
 (
 l_selected = 'N'
 )
 else
 (
 l_selected = 'Y'
 )

 RETURN l_selected
/*================ FORMULA SECTION END =======================*/
```

Total Compensation Item Formula Type

The Total Compensation Item formula determines compensation information that isn't stored in the other predefined item source types.

You select the formula when you manage compensation items on the Create or Edit Compensation Items page.

Contexts

The following contexts are available to formulas of this type:

- DATE_EARNED
- EFFECTIVE_DATE
- END_DATE
- START_DATE
- HR_ASSIGNMENT_ID
- HR_TERM_ID
- JOB_ID
- LEGISLATIVE_DATA_GROUP_ID
- COMPENSATION_RECORD_TYPE
- ORGANIZATION_ID
- PAYROLL_ASSIGNMENT_ID
- PAYROLL_RELATIONSHIP_ID
- PAYROLL_TERM_ID

- PERSON_ID

Database Items

Database items related to Person, Assignment, Salary, Element Entries, Compensation Record, and From and End Dates are available to formulas of this type.

Input Variables

The following input variables are available to formula of this type.

Input Value	Data Type	Required	Description
CMP_ IV_PERIOD_ID	Char	Y	Period ID
CMP_ IV_ PERIOD_ START_DATE	Date	Y	Statement Period Start Date
CMP_ IV_ PERIOD_ END_DATE	Date	Y	Statement Period End Date

Return Values

The following return variables are available to formula of this type.

Return Value	Data Type	Required	Description
COMPENSATION_ DATES	Date	Y	One to 15 transaction dates delimited by semicolon, maximum 250 characters.
VALUES	Char	Y	One to 15 transaction values delimited by semicolon, maximum 250 characters. Must be the same number of values as dates.
ASSIGNMENTS	Char	N	One to 15 transaction assignments delimited by semicolon, maximum 250 characters. Must be the same number of assignments as dates. Can return an empty space with a delimiter (; ;).

Return Value	Data Type	Required	Description
LEGALEMPLOYERS	Char	N	One to 15 legal employer IDs delimited by semicolon, maximum 250 characters. Must be the same number of assignments as dates. Can return an empty space with a delimiter (; ;).
COMPENSATION_ DATES1	Date	Y	Second variable for transaction dates from 16 to 30 if limit of 250 characters is exceeded.
VALUES1	Char	Y	Second variable for transaction values from 16 to 30 if limit of 250 characters is exceeded.
ASSIGNMENTS1	Char	N	Second variable for transaction assignments from 16 to 30 if limit of 250 characters is exceeded.
LEGALEMPLOYERS1	Char	N	Second variable for legal employer IDs from 16 to 30 if limit of 250 characters is exceeded.
COMPENSATION_ DATES2	Date	Y	Transaction dates from 31 to 45.
VALUES2	Char	Y	Transaction values from 31 to 45.
ASSIGNMENTS2	Char	N	Transaction assignments from 31 to 45.
LEGALEMPLOYERS2	Char	N	Legal employers from 31 to 45.

Return Value	Data Type	Required	Description
COMPENSATION_ DATES3	Dates	Y	Transaction dates from 46 to 60.
VALUES3	Char	Y	Transaction values from 46 to 60.
ASSIGNMENTS3	Char	N	Transaction assignments from 46 to 60.
LEGALEMPLOYERS3	Char	N	Legal employers from 46 to 60.

Sample Formula

This sample formula returns one date and one value based on the worker ID.

```
/**************************************************************************
FORMULA NAME : Total Compensation Simple Item Formula
FORMULA TYPE : Total Compensation Item
DESCRIPTION : Returns one date and one value.
*************************************************************************/

/*=========== INPUT VALUES DEFAULTS BEGIN =====================*/
INPUTS ARE CMP_IV_PERSON_ID (text), CMP_IV_PERIOD_START_DATE (date), CMP_IV_PERIOD_END_DATE (date)
DEFAULT FOR CMP_IV_PERSON_ID IS '-1'
DEFAULT FOR CMP_IV_PERIOD_START_DATE IS '4712/12/31' (date)
DEFAULT FOR CMP_IV_PERIOD_END_DATE IS '4712/12/31' (date)

/*============ INPUT VALUES DEFAULTS ENDS =====================*/

/*=================== FORMULA SECTION BEGIN =================*/

 COMPENSATION_DATES = '2005/01/01'
 VALUES = '500.00'

RETURN COMPENSATION_DATES, VALUES

/*================ FORMULA SECTION END ======================*/
```

This sample formula returns multiple variables.

```
/**************************************************************************
FORMULA NAME : Total Compensation Multi Item Formula
FORMULA TYPE : Total Compensation Item
DESCRIPTION : Returns multiple variables.
*************************************************************************/

/*=========== INPUT VALUES DEFAULTS BEGIN =====================*/
INPUTS ARE CMP_IV_PERSON_ID (text), CMP_IV_PERIOD_START_DATE (date) , CMP_IV_PERIOD_END_DATE (date)

/*=========== INPUT VALUES DEFAULTS ENDS =====================*/

/*================= FORMULA SECTION BEGIN =================*/
```

```
 COMPENSATION_DATES = '2009/01/01;2009/02/01;2009/03/01'
 COMPENSATION_DATES1 = '2009/07/01;2009/08/01;2009/09/01'
 COMPENSATION_DATES2 = '2009/10/01;2009/11/01;2009/12/01'
 COMPENSATION_DATES3 = '2009/10/01;2009/11/01;2009/12/01'
 VALUES = '200.00;200.00;300.00'
 VALUES1 = '300.00;500.00;500.00'
 VALUES2 = '500.00;500.00;600.00'
 VALUES3 = '600.00;600.00;700.00'

 /* Returns only first two assignment */
 ASSIGNMENTS = ';1234567890;1234567890'
 ASSIGNMENTS1 = '1234567890;1234567890;1234567890'
 /* Returns last two assignments */
 ASSIGNMENTS2 = ';1234567890;1234567890'
 /* Returns first and last assignments */
 ASSIGNMENTS3 = '1234567890;;1234567890'


 LEGALEMPLOYERS = '0123456789;;0123456789'
 LEGALEMPLOYERS1 = '0123456789;0123456789;0123456789'
 LEGALEMPLOYERS2 = '0123456789;0123456789;0123456789'
 LEGALEMPLOYERS3 = '0123456789;0123456789'

RETURN
 COMPENSATION_DATES,VALUES,COMPENSATION_DATES1,VALUES1,COMPENSATION_DATES2,VALUES2,COMPENSATION_D

/*================ FORMULA SECTION END =======================*/
```

8 Formulas for Benefits

Benefits Formulas: Overview

Use formulas to configure your plan design to the requirements of your enterprise. They provide a flexible alternative to delivered business rules for such purposes as:

- Date calculations, such as:
  - Enrollment start and end dates
  - Rate or coverage start and end dates
  - Waiting periods and enrollment periods
  - Action item due dates
- Calculations of rate and coverage amount, minimum and maximum, or upper and lower limits
- Certification requirements
- Partial month and proration calculations
- Eligibility and participation evaluation

For example, you can write a formula to calculate benefits eligibility for those cases where the provided eligibility criteria don't accommodate your particular requirements.

Benefits Fast Formula Reference Guide

The Benefits Fast Formula Reference guide explains some of the most frequently used benefits formula types. All formula types explained in the guide include sample code, contexts, database items, input variables, and return variables.

For more information, see Benefits Fast Formula Reference Guide (1456985.1) on My Oracle Support at https://support.oracle.com.

Related Topics

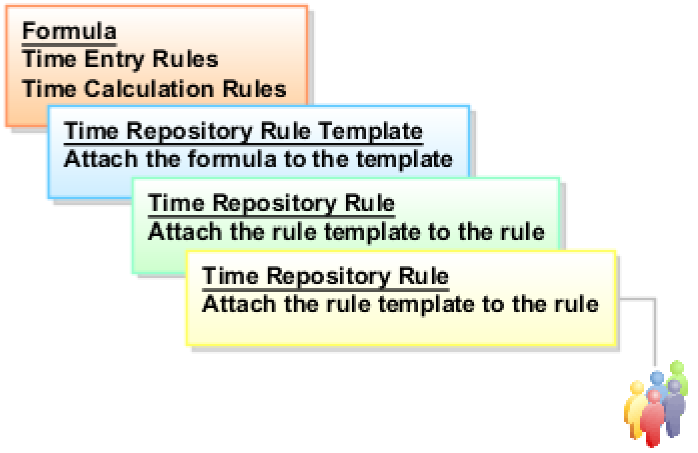
- Benefits Formula Evaluation: Points to Consider

9 Formulas for Time and Labor

Formulas for Time and Labor: Explained

Oracle Fusion Time and Labor uses time repository rule templates and rules to simplify the customization of formulas created with Oracle Fusion Fast Formula.

The following diagram shows that you use rule templates to create rules from formulas, and combine rules into rule sets to assign to worker profiles.


Formula
Time Entry Rules
Time Calculation Rules
Time Repository Rule Template
Attach the formula to the template
Time Repository Rule
Attach the rule template to the rule
Time Repository Rule
Attach the rule template to the rule

The formulas contain delivered combinations of rule parameters and output results. You can use a single formula in multiple templates. You can define many rules using the same template, varying the input or output parameters.

Rule Templates

Rule templates simplify rule configuration. Administrators determine and document exactly which parameters the formula requires and the value that the formula returns. When administrators create a rule, they select the template to use rather than the formula. The template automatically populates the description of all outputs and helps administrators enter the parameters.

The rule template ensures that:

- The parameters are of the correct data type
- The output uses only specific time attributes
- The correct number of outputs are associated with the formula results

Formula Types

The following table lists and describes formula types you can use with templates to create time repository rules for Time and Labor:

Formula and Rule Type	Description	Example
Time Entry	Validate time card entries and can generate a message with a defined severity	Workers can't enter more than a specified number of hours per week.

Formula and Rule Type	Description	Example
Time Calculation	Update or create time card entries and manipulate the data to create processed results based on formula logic.	The worker enters 10 hours. The calculation rule updates the entries to 8 hours of regular time and 2 hours of overtime.

Time and Labor Fast Formula Reference Guide

The Time and Labor Fast Formula Reference guide explains how to use Fast Formula with Time and Labor. This guide provides the contexts, database items, and input variables for the formula types used in Time and Labor. It also provides sample formulas and the fixed parameters, valid functions, and return variables for each.

For more information, see Time and Labor Fast Formula Reference Guide (1990057.1) on My Oracle Support at https://support.oracle.com.

Glossary

absence type

A grouping of absences, such as illness or personal business that is used for reporting, accrual, and compensation calculations.

accrual absence plan

A benefit that entitles workers to accrue time for the purpose of taking leave.

assignment statement

A statement that formulas use to set a value for a local variable.

balance

Positive or negative accumulations of values over periods of time, typically generated by payroll runs. A balance can sum pay values, time periods, or numbers.

database item

An item of information that has special programming attached, which formulas and HCM extracts use to locate and retrieve the data.

element

Component in the calculation of a person's pay. An element may represent a compensation or benefit type, such as salary, wages, stock purchase plans, pension contributions, and medical insurance.

fast formula

A simple way to write formulas using English words and basic mathematical functions. Formulas are generic expressions of calculations or comparisons that repeat with different input values.

globals

Store values that are constant over a period of time. You can reference them in several formulas. Examples include the name of a rate, a specific date, or a company term.

local variable

A variable that you use in only one formula. You can change the value of a local variable by assigning a value in an assignment statement.

object group

User-defined set of elements or people that restrict the items you want to include in various processes and reports.

qualification absence plan

A benefit that entitles workers to paid leave time as a result of an event, such as childbirth, illness, or injury.

return statement

A statement that formulas use to return values in local fast formula variables.