# Oracle Fusion Cloud Human Resources

## How do I configure business rules for Redwood salary pages?

Oracle Fusion Cloud Human Resources
How do I configure business rules for Redwood salary pages?

G42323-02



Author: Srinivas Vellikad

# Contents

# Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

## Get Help in the Applications

Some application pages have help icons to give you access to contextual help. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons. If the page has contextual help, help icons will appear.

## Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

## Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

## Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

# 1 How do I configure business rules for Redwood salary pages?

## Salary Business Rules Introduction

You can extend the Redwood salary pages by configuring fields and regions, changing the page properties using business rules, and creating field-level validations and tailored messages to meet your business requirements.

Business rules let you to control the display of regions and fields on Redwood pages. You can configure business rules in Oracle Visual Builder Studio Express mode to personalize Redwood pages, including defaulting and validating field values.

You can either use delivered rules built for best practices or create your own rules.

## Getting Started

You need to set up Visual Builder Studio before you can start configuring business rules. For details, see *Set Up VB Studio to Extend Oracle Cloud Applications*.

## Access Requirement

You need the Human Capital Management Application Administrator role to work in an Oracle Fusion apps sandbox. For details, see the *Securing HCM* guide.

## Implementation Considerations for Salary Processes

Here are the considerations for the Change Salary and Salary History processes.

### Change Salary

You can typically use field value defaulting and validation in these cases.

- Default the proposed salary basis when employee legal employer is Vision when line managers propose salary change.
- Default the effective date to current date and next salary review date to 1 year after the effective date, when line managers propose salary change.
- Default the salary basis based on assignment flexfields when line managers propose salary change.
- Default salary basis for a full time employee belonging to US and a certain grade.

- Validate that the when date isn't in the past.
- Validate that the salary amount isn't below a minimum value.
- Validate that the when start date is 1st of next month.
- Validate that the user hasn't changed the rate component value.
- Advanced expressions working on Global HR and Recruiting pages need to be updated somewhat to work on the Change Salary page.

**Note:** Business rule defaults happen only on the first visit to the Salary section. They don't happen again, even if changes were made to other sections.

This table lists the supported attributes, exceptions, and the implementation recommendations for the **My Team** > **Change Salary** and **My Client Groups** > **Change Salary** pages.

| In the Conditions | To Default Field Values | To Validate Field Values | Implementation Guidelines |
|---|---|---|---|
| • User Roles<br>• Legal Employer<br>• Business Unit<br>• Country<br>• Start Date<br>• Action<br>• Reason<br>• Assignment attributes including<br>  ○ Assignment Category<br>  ○ Assignment Status<br>  ○ Bargaining Unit<br>  ○ Department<br>  ○ FTE<br>  ○ Full time or part time<br>  ○ Grade<br>  ○ Grade Code<br>  ○ Grade Ladder<br>  ○ Grade Ladder Step<br>  ○ Salary Range Default Value (Grade Rate Value)<br>  ○ Hourly or Salaried<br>  ○ Hourly Paid or Salaried<br>  ○ Job<br>  ○ Job Code<br>  ○ Job Family | • Start Date<br>• Action<br>• Reason<br>• Proposed Salary Basis<br>• Salary Amount<br>• Next Salary Review Date | • User Roles<br>• Legal Employer<br>• Business Unit<br>• Country<br>• Start Date<br>• Action<br>• Reason<br>• Assignment attributes including<br>  ○ Assignment Category<br>  ○ Assignment Status<br>  ○ Bargaining Unit<br>  ○ Department<br>  ○ FTE<br>  ○ Full time or part time<br>  ○ Grade<br>  ○ Grade Code<br>  ○ Grade Ladder<br>  ○ Grade Ladder Step<br>  ○ Hourly or Salaried<br>  ○ Hourly Paid or Salaried<br>  ○ Job<br>  ○ Job Code<br>  ○ Job Family<br>  ○ Job Function<br>  ○ Legal Employer | **Defaulting**<br>• Salary amount can be defaulted when using the user determined type of salary basis only.<br>• Fields in the salary section can be defaulted based on the fields listed "in the conditions" column of this table.<br>• Initial Field Values and Target Fields on salary fields are now supported.<br>• When there's more than one assignment on the same day, details from the highest sequenced assignment will be considered.<br>• Warnings are triggered on clicking Continue in a CGP flow and hence can't be seen by the user. |

| In the Conditions | To Default Field Values | To Validate Field Values | Implementation Guidelines |
|---|---|---|---|
| ○ Job Function<br>○ Legal Employer<br>○ Legislation Code<br>○ Location<br>○ Next Payroll Start Date<br>○ Payroll Start Date<br>○ Person Assignment Union<br>○ Person Assignment Union Member<br>○ Person Type<br>○ Position<br>○ Position Budget Value<br>○ Position Code<br>○ Position Standard Working Hours<br>○ Primary Work Relationship<br>○ Termination Date<br>○ US Job Info Overtime Status<br>○ Worker Category<br>○ Working Hours<br>○ Working Hours Frequency<br><br>Additional Assignment Info segments<br>• Assignment Flex<br>• Departments Flex<br>• Grade Ladders Flex<br>• Grades Flex<br>• Jobs Flex<br>• Positions Flex | | ○ Legislation Code<br>○ Location<br>○ Next Payroll Start Date<br>○ Payroll Start Date<br>○ Person Assignment Union<br>○ Person Assignment Union Member<br>○ Person Type<br>○ Position<br>○ Position Budget Value<br>○ Position Code<br>○ Position Standard Working Hours<br>○ Primary Work Relationship<br>○ Salary Range Minimum<br>○ Salary Range Midpoint<br>○ Salary Range Maximum<br>○ Salary Range Default Value (Grade Rate Value)<br>○ Termination Date<br>○ US Job Info Overtime Status<br>○ Worker Category<br>○ Working Hours<br>○ Working Hours Frequency<br>○ Simple Components<br>○ Rate Components<br>○ Additional Assignment Info segments<br>-<br>- Assignment Flex<br>- Departments Flex<br>- Grade Ladders Flex<br>- Grades Flex<br>- Jobs Flex<br>Positions Flex | |

## Salary History

You can typically use field value defaulting and validation in these cases.

- Default the proposed salary basis based on the employee's legal employer when compensation managers propose a salary change.

- Default the effective date to first day of next month and next salary review date to 1 year after the effective date, when line managers propose a salary change.
- Default the salary basis based on assignment flexfields when proposing salary change.
- Validate that users don't enter retro-active salary changes.
- Validate that the change of salary occurs on the first of the month.
- Validate that the salary amount is above a minimum value.
- Validate that users don't change a simple component value.
- Validate that users don't enter a zero value for a simple component.
- Validate the sum of a certain simple component is based on a formula (Adjustment needs to be sum of all components minus cost of living).
- Advanced expressions working on Global HR and Recruiting pages need to be updated somewhat to work on the Salary History page.

**Note:** Business rule defaults happen only on the first visit to the Salary section. They don't happen again, even if changes were made to other sections.

This table lists the supported attributes, exceptions, and the implementation recommendations for Salary History. It applies to the **My Client Groups** > **Admin Salary History** page.

| In the Conditions to Default Values | To Default Field Values | In the Condition to Validate Values | To Validate Field Values | Implementation Guidelines |
|---|---|---|---|---|
| • User Roles<br>• Legal Employer<br>• Business Unit<br>• Country<br>• Start Date<br>• Action<br>• Reason | • Start Date<br>• Action<br>• Reason<br>• Proposed Salary Basis<br>• Salary Amount<br>• Next Salary Review Date | • User Roles<br>• Legal Employer<br>• Business Unit<br>• Country<br>• Start Date<br>• Action<br>• Reason | Not Applicable | • Salary amount can be defaulted when using user determined type of salary basis only.<br>• Defaulting is supported when salary is being created and not on correction of existing salary.<br>• When there's more than one assignment on the same day, |

| In the Conditions to Default Values | To Default Field Values | In the Condition to Validate Values | To Validate Field Values | Implementation Guidelines |
|---|---|---|---|---|
| • Assignment attributes including<br>o Assignment Category<br>o Assignment Status<br>o Bargaining Unit<br>o Department<br>o FTE<br>o Full time or part time<br>o Grade<br>o Grade Code<br>o Grade Ladder<br>o Grade Ladder Step<br>o Salary Range Default Value (Grade Rate Value)<br>o Hourly or Salaried<br>o Hourly Paid or Salaried<br>o Job<br>o Job Code<br>o Job Family<br>o Job Function<br>o Legal Employer<br>o Legislation Code<br>o Location<br>o Next Payroll Start Date<br>o Payroll Start Date<br>o Person Assignment Union<br>o Person Assignment Union Member<br>o Person Type<br>o Position<br>o Position Budget Value<br>o Position Code<br>o Position Standard Working Hours<br>o Primary Work Relationship<br>o Termination Date | | • Assignment attributes including<br>o Assignment Category<br>o Assignment Status<br>o Bargaining Unit<br>o Department<br>o FTE<br>o Full time or part time<br>o Grade<br>o Grade Code<br>o Grade Ladder<br>o Grade Ladder Step<br>o Hourly or Salaried<br>o Hourly Paid or Salaried<br>o Job<br>o Job Code<br>o Job Family<br>o Job Function<br>o Legal Employer<br>o Legislation Code<br>o Location<br>o Next Payroll Start Date<br>o Payroll Start Date<br>o Person Assignment Union<br>o Person Assignment Union Member<br>o Person Type<br>o Position<br>o Position Budget Value<br>o Position Code<br>o Position Standard Working Hours<br>o Primary Work Relationship<br>o Salary Range Minimum<br>o Salary Range Midpoint<br>o Salary Range Maximum | | details from the highest sequenced assignment will be considered.<br>• Warnings are triggered on clicking Continue in a CGP flow and hence can't be seen by the user.<br>• Validations can't be performed across salaries.<br>• Initial Field Values and Target Fields on salary fields are supported now.<br>• Validations will be triggered on clicking Save or OK button and not on clicking Submit button. |

| In the Conditions to Default Values | To Default Field Values | In the Condition to Validate Values | To Validate Field Values | Implementation Guidelines |
|---|---|---|---|---|
| ○ US Job Info Overtime Status<br>○ Worker Category<br>○ Working Hours<br>○ Working Hours Frequency<br><br>Additional Assignment Info segments<br>• Assignment Flex<br>• Departments Flex<br>• Grade Ladders Flex<br>• Grades Flex<br>• Jobs Flex<br>• Positions Flex | | ○ Salary Range Default Value (Grade Rate Value)<br>○ Termination Date<br>○ US Job Info Overtime Status<br>○ Worker Category<br>○ Working Hours<br>○ Working Hours Frequency<br>○ Simple Components<br>○ Rate Components<br><br>Additional Assignment Info segments<br>• Assignment Flex<br>• Departments Flex<br>• Grade Ladders Flex<br>• Grades Flex<br>• Jobs Flex<br>• Positions Flex | | |

# Configure Business Rules for Salary Processes

Here are examples of a defaulting rule and multiple validating rules.

## Validate that the user hasn't changed the rate component value

Advanced Expression

```
/* eslint-disable dot-notation */
define([], () => {
 'use strict';



 /**
 *
 * @param {object} context
 * @return {boolean}
 */
 function runCondition(context) {
 const { $componentContext, $fields, $modules, $user } = context;

 let salBasisType = $fields.compensationSalaries.SalaryBasisType.$value();
```

```
if(salBasisType == 'R'){
let newSalaryBasisId = $fields.compensationSalaries.SalaryBasisId.$value();
let curSalaryBasisId = $fields.compensationSalaries.SalaryBasisId.$initialValue();

if(newSalaryBasisId != curSalaryBasisId){
return false;
}

let curValue = 0;
let newValue = 0;

let curComponentsArray = $fields.compensationSalaries.salaryPayRateComponents.$initialValue().items;

for(let i in curComponentsArray){
let cmpt=curComponentsArray[i];
if(cmpt.Name == 'ZCMP GSP RATE BASE'){
curValue = Number(cmpt.RateAmount);
}
}


let componentsArray = $fields.compensationSalaries.salaryPayRateComponents.$value().items;
for(let j in componentsArray){
let cmpt=componentsArray[j];
if(cmpt.Name == 'ZCMP GSP RATE BASE'){
newValue = Number(cmpt.RateAmount);
}

}

if(curValue != newValue){
return true;
}
}

return false;
}

return { runCondition };
});
```

# Validate that a value is provided for the rate component

## Advanced Expression

```
/* eslint-disable dot-notation */
define([], () => {
 'use strict';



 /**
 *
 * @param {object} context
 * @return {boolean}
 */
 function runCondition(context) {
 const { $objectContext, $fields, $modules, $user, $value } = context;

 let salary = $fields['compensationSalaries'].$value();
 let payRateComponents = salary['salaryPayRateComponents'];
 if (payRateComponents) {
 let componentsArray = payRateComponents.items;
 if (componentsArray) {
 for (let i = 0; i < componentsArray.length; i++) {
```

```
 let element = componentsArray[i];
 if (element.Name === 'ZCMP RTS USVS ARS Base Salary' && element.RateAmount == 0) {
 return true;
 }
 }
 }
 }
 return false;
 }

 return { runCondition };
});
```

## **Validate that the sum of a certain simple component needs to be based on a formula** (Adjustment needs to be sum of all components minus cost of living)

Advanced Expression

```
/* eslint-disable dot-notation */
define([], () => {
 'use strict';



 /**
 *
 * @param {object} context
 * @return {boolean}
 */
 function runCondition(context) {
 const { $componentContext, $fields, $modules, $user } = context;

 let salBasisType = $fields.compensationSalaries.SalaryBasisType.$value();
 let sum = 0;
 let result = 0;
 let difference = 0;

 if(salBasisType == 'C'){
 let componentsArray = $fields.compensationSalaries.salaryComponents.$value().items;
 for(let i in componentsArray){
 let cmpt=componentsArray[i];
 if(cmpt.ComponentReasonCode == 'ADJUSTMENT'){
 result = Number(cmpt.AdjustmentAmount);
 }
 else if(cmpt.ComponentReasonCode == 'COST_OF_LIVING'){
 difference += Number(cmpt.AdjustmentAmount);
 }
 else{
 sum += Number(cmpt.AdjustmentAmount);
 }
 }
 if(result != sum - difference){
 return true;
 }
 }

 return false;
 }

 return { runCondition };
});
```

# Default the effective date to current date and next salary review date to 1 year after the effective date, when line managers propose salary change

Advanced Expression

```
/* eslint-disable dot-notation */
define([], () => {
 'use strict';



 /**
 * Default value expression for whenAndWhy.StartDate
 * @param {object} context
 * @return {date}
 */
 function getWhenAndWhyStartDate(context) {
 const { $objectContext, $fields, $modules, $user } = context;

 let date = new Date();
 let firstDay = new Date(date.getFullYear(), date.getMonth()+1, 1);

 let year = firstDay.toLocaleString("default", { year: "numeric" });
 let month = firstDay.toLocaleString("default", { month: "2-digit" });
 let day = firstDay.toLocaleString("default", { day: "2-digit" });

 return (year + "-" + month + "-" + day);
 }

 return { getWhenAndWhyStartDate };
});
```

# Validate that the user hasn't changed the simple component value

Advanced Expression

```
/* eslint-disable dot-notation */
define([], () => {
 'use strict';



 /**
 *
 * @param {object} context
 * @return {boolean}
 */
 function runCondition(context) {
 const { $componentContext, $fields, $modules, $user } = context;

 let salBasisType = $fields.compensationSalaries.SalaryBasisType.$value();

 if(salBasisType == 'ORA_SIMPLE_COMPONENTS'){
 let newSalaryBasisId = $fields.compensationSalaries.SalaryBasisId.$value();
 let curSalaryBasisId = $fields.compensationSalaries.SalaryBasisId.$initialValue();

 if(newSalaryBasisId != curSalaryBasisId){
 return false;
 }

 let curValue = 0;
 let newValue = 0;
```

```
let curComponentsArray = $fields.compensationSalaries.salarySimpleComponents.$initialValue().items;

for(let i in curComponentsArray){
let cmpt=curComponentsArray[i];
if(cmpt.ComponentCode == 'ORA_WAGE_PROGRESSION_RATE'){
curValue = Number(cmpt.Amount);
}
}


let componentsArray = $fields.compensationSalaries.salarySimpleComponents.$value().items;
for(let j in componentsArray){
let cmpt=componentsArray[j];
if(cmpt.ComponentCode == 'ORA_WAGE_PROGRESSION_RATE'){
newValue = Number(cmpt.Amount);
}

}

if(curValue != newValue){
return true;
}
}

return false;
}

return { runCondition };
});
```

# Validate that user hasn't entered a 0 value for a simple component

Advanced Expression

```
/* eslint-disable dot-notation */
define([], () => {
 'use strict';



 /**
 *
 * @param {object} context
 * @return {boolean}
 */
 function runCondition(context) {
 const { $componentContext, $fields, $modules, $user } = context;

 let salBasisType = $fields.compensationSalaries.SalaryBasisType.$value();

 if(salBasisType == 'ORA_SIMPLE_COMPONENTS'){
 let componentsArray = $fields.compensationSalaries.salarySimpleComponents.$value().items;
 for(let i in componentsArray){
 let cmpt=componentsArray[i];
 //To Check if specific component has 0 amount value
 // if(cmpt.ComponentCode == 'ORA_WAGE_PROGRESSION_RATE' && cmpt.Amount==0){
 // return true;
 // }
 //To Check if any component has 0 amount value
 if(cmpt.Amount==0){
 return true;
 }

 }
 }
```

```
 return false;
 }

 return { runCondition };
});
```