Business Rule

Business rules run when a ServiceNow form is displayed, or when the update, save, or delete operations occur. They are "event-driven". When they do execute, Business Rules can set field values, add a message, or run a script.

A business rule is a server-side script that runs when a record is displayed, inserted, updated, or deleted, or when a table is queried. Use business rules to accomplish tasks like automatically changing values in form fields when certain conditions are met, or to create events for email notifications and script actions.

Even though there are now other scripting methods besides using Business rules, there are likely 1500+ business rules existing in your ServiceNow instance.

WHY USE BUSINESS RULES?

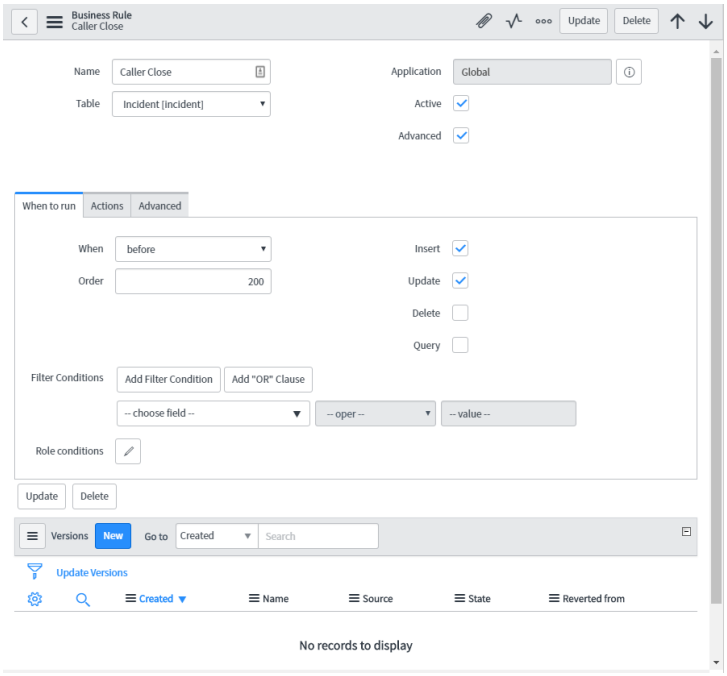
Business rules are fast. They are server-side and run much faster than other types of scripting in ServiceNow. It is advised to use Business Rules and Workflow as much as possible. Avoid using client scripts, except for Catalog Client Scripts (which are unavoidable). This is due to performance reasons and browser issues that client scripts can introduce.

ADVANTAGES OF BUSINESS RULES

Performance. When running code on the server, it often has a faster load time and processing time than a client script. Not affected by type of browser
Can perform complex database lookups
Can dot-walk many levels, however three levels is often a recommend maximum

DISADVANTAGES OF BUSINESS RULES

Not as interactive as client scripts, needs an event like save, delete, or display to run



WHEN FIELD

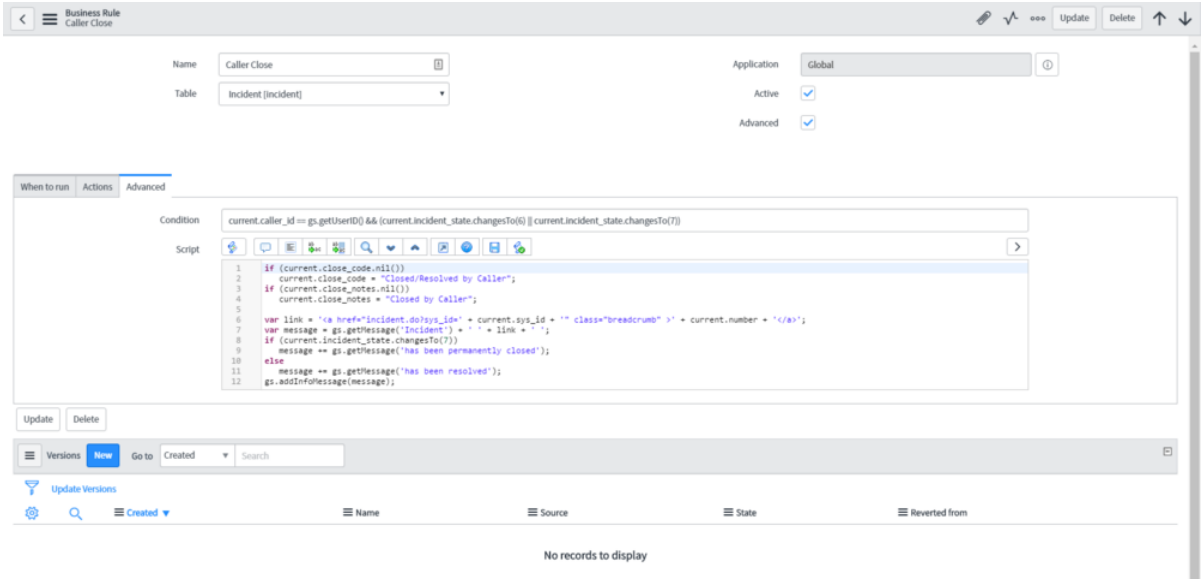
*The **when** field on a business rule is very important. Most business rules run "before", which means before save. However, occasionally you will use an "after" business rule to update a related table, which is purely for performance reasons.*

- ***display** - Use to provide client scripts access to server-side objects.*
- ***before** - Use to update information on the current object. For example, a business rule containing current.state=3; would set the **State** field on the current record to the state with a **Value** of **3**.*
- ***after** - Use to update information on related objects that need to be displayed immediately, such as GlideRecord queries.*
- ***async** - Use to update information on related objects that do not need to be displayed immediately, such as calculating metrics and SLAs.*

ADVANCED

You don't have to use scripting in Business Rules anymore. You can just use Filter Conditions, Role Conditions, and Actions to accomplish what you need instead.

*I am from a older time when that functionality was not included in business rules. To get to the "old way" or "advanced", click that **Advanced checkbox**. When you click Advanced, you get the advanced tab and can begin scripting.*



CONDITION

The condition field indicates when the business rule should run. ServiceNow evaluates the condition separately from the script. So, if you use a limiting condition, you can improve performance by not having the script section run.

It is easier to debug business rules when you can see which one meet a particular condition and which do not.

SCRIPTS AND SCRIPTING

For scripts, I thought I would include some important concepts:

#1 PREVENT RECURSIVE BUSINESS RULES

Avoid using current. Update() in a business rule script. The update () method triggers business rules to run on the same table for insert and update operations, leading to a business rule calling itself over and over.

Business Rules run before, after, and display of a record. When you use current.update() in a business rule, that will cause a "double update" of a record or worse.

Here are some examples of the chaos this can cause:

- ***before Business rule and current.update():*** *Business rule runs, current.update() saves the record, remaining business rules run and record will saved again. This results in duplicate operations such as duplicate notifications and updates.*
- ***after Business rule and current.update():*** *Record saves. after Business rule runs, current.update() saves the record again. This results in duplicate operations such as duplicate notifications and updates.*

- ***async Business rule and current.update()****: Record saves. async Business rule runs later on, current.update() saves the record again. This results in duplicate operations such as duplicate notifications and updates with a gap of time in-between.*
- ***display Business rule and current.update()****: display Business rule runs every time the form is displayed and the form attempts to save due to current.update(). User might not have filled out the form all the way and it is an annoying experience to the user.*

Don't use current.update() in a business rule! There are certain situations when it is ok, but very rarely. Same goes with using g_form.save() in a client script.

#2 ENCLOSE CODE IN FUNCTIONS

You should always enclose your scripts in a function. When code is not enclosed in a function, variables and other objects are available to all other server-side scripts. This availability can lead to unexpected consequences that are difficult to troubleshoot.

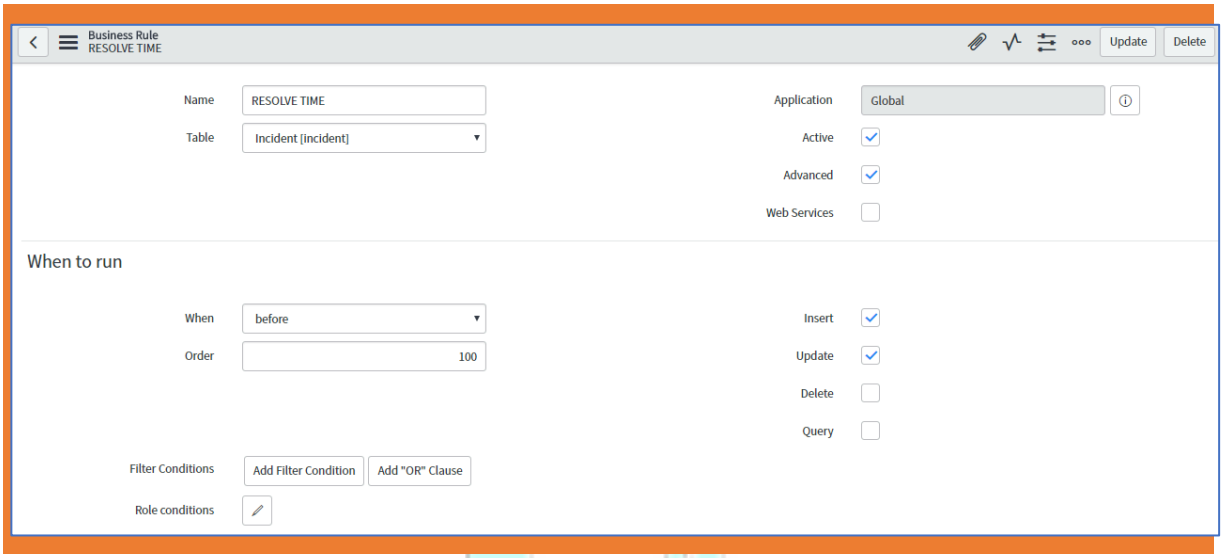
#3 USE SMALL AND SPECIFIC BUSINESS RULES

By using small and specific business rules, they are easier to debug and maintain than a large and complex business rule.

It is tempting to make big business rule, but that will often hinder performance or make it difficult to evaluate.

Business Rule Samples

1. Calculating time difference between open time & resolved time



CONDITION : *current.incident_state == IncidentState.RESOLVED || current.incident_state == IncidentState.CLOSED*

Note: Use Condition Builder instead of code

SCRIPT :

```
setResolutionFields();

function setResolutionFields() {
if (current.resolved_by.nil())
current.resolved_by = gs.getUserID();
if (current.resolved_at.nil())
current.resolved_at = gs.nowDateTime();

var dataChange = current.opened_at.changes() || current.resolved_at.changes();
var opened = current.opened_at.getDisplayValue();
var resolved = current.resolved_at.getDisplayValue();

if (dataChange || current.business_duration.nil())
current.business_duration = gs.calDateDiff(opened, resolved, false);

if (dataChange || current.business_stc.nil())
current.business_stc = gs.calDateDiff(opened, resolved, true);

if (dataChange || current.calendar_duration.nil())
current.calendar_duration = gs.dateDiff(opened, resolved, false);

if (dataChange || current.calendar_stc.nil())
current.calendar_stc = gs.dateDiff(opened, resolved, true);
}
```


K-now Cloud
know your cloud