summary:

the idea is based on our experiences in current project and previous projects during Data migration activities where we had to manually go into each object and turn off the rules taking lot of effort. Also an instance where the integration user was facing issue pushing data and some of the records were failing to get processed whereas the business need was to store the data even if it’s incomplete.

Requirement is to turn them off for individual users or set of users

For example say, when a Data migration user (if selected above for bypass) tries to load data into the org in an object. If the rules/ process builders /triggers are turned off for this user then they won’t run during data load at all. But at the same time if a normal user accessing the application from UI tries to create the record for same object then the rules/ process builders /triggers will execute as normal.

So here the same rules/process builders /triggers is turned off for one user and continues to run for other user at the same time.

Business Value:

• Helpful in multiple use cases (Data migration, inbound Integration, batch processing, Super Admins, etc).

• Single screen to manage which rules & process builders needs to be deactivated or activated saving time of deployment team.

• No downtime or deployment needed when activating or deactivating the rules / Process builders/triggers for a user.

• For example of Data migration user, the rules/ process builders /triggers can stay deactivated even after the data load activity is completed saving time of deployment team as there is no need to turn these ON again for other users.

• No impact on other users when bypass rule created for a particular user.

• The overall effort (deployment team & Data migration) is reduced hence saving time and money.

Solution explanation:

Bypass Rule custom metadata is used to store validation rules, triggers and process builders’ information in Salesforce that are enabled or disabled from UI.

Fields available:

• Disable Rule  true or false based on weather rule is enable or disable and used in scripts

• Object text field 255 char limit and stores object API name

• Type picklist field with values Process Builders, Triggers and Validation

• Rules

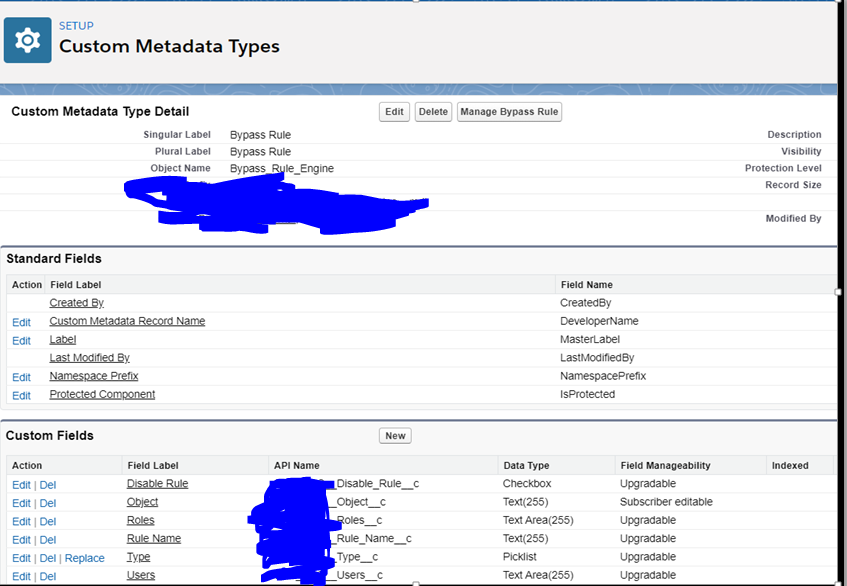
• Roles Text area (255) this is used to role Ids and used in scripts

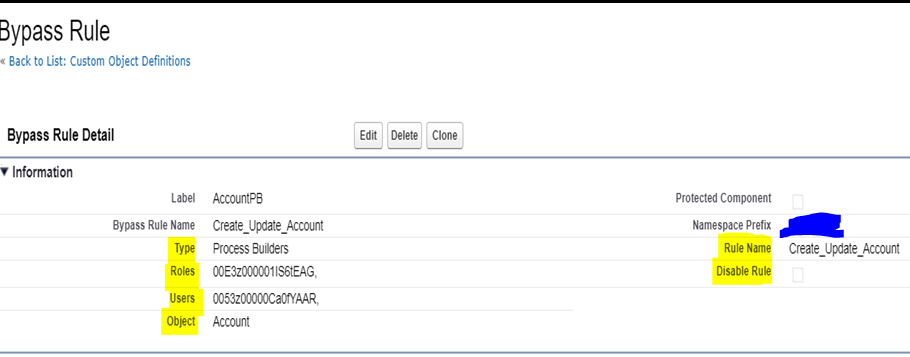
• Users Text area (255) this is used to user Ids and used in scripts

• Rule Name This field stores the API name of triggers, process builders and validation rules and used in scripts

• Label standard master label field and we are setting value in format of object name + short alias for each i.e. PB for Process Builders, VR for validation rules and TR for Triggers

• Bypass Rule Name this is standard name field and we are storing API name similar to Rule name.





Replace Yellow color value to respective metadata record **rule name or api name of process builder** and update/add these condition in Process builder nodes.

**If (**

**AND (**

**$CustomMetadata.Bypass\_Rule\_Engine\_\_mdt.Create\_Update\_Account.Disable\_Rule\_\_c,**

**OR (**

**CONTAINS($CustomMetadata.Bypass\_Rule\_Engine\_\_mdt.Create\_Update\_Account.Users\_\_c, $User.Id),**

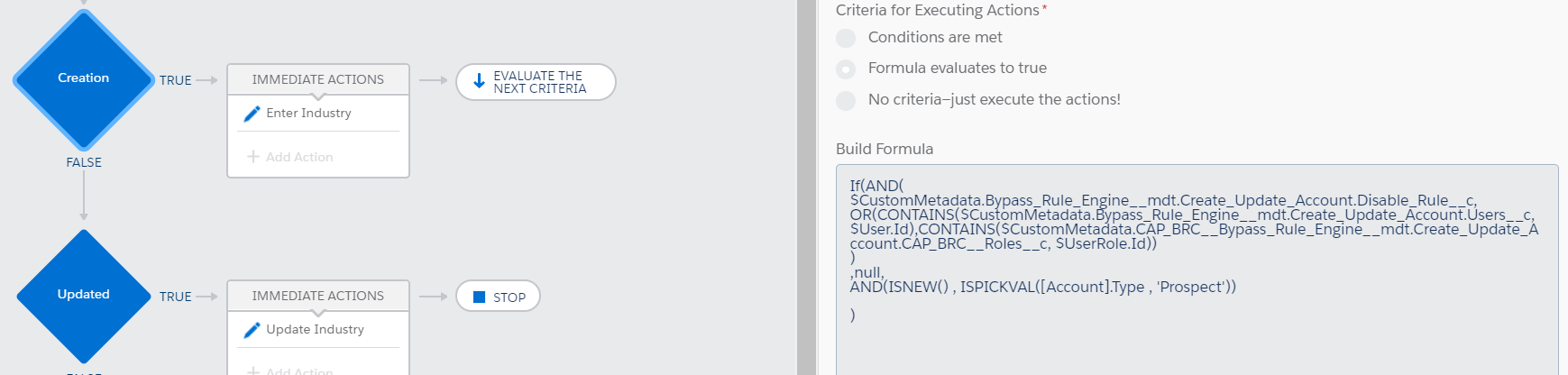
**CONTAINS($CustomMetadata.CAP\_BRC\_\_Bypass\_Rule\_Engine\_\_mdt.Create\_Update\_Account.CAP\_BRC\_\_Roles\_\_c, $UserRole.Id)**

**)**

**)**

**, null,**

**“Execution condition here”))**



### Validation Rules

* In case of **New Validation Rule** 🡪 please do not include script directly, first activate the validation rule and
* Then run the **Bypass Rules Configurator** tool so that rule entry will be created automatically in custom metadata.
* Once this is completed then edit the validation rule and add below script into each node
* In case of **Existing Validation rule🡪** add the script in each node

Replace Yellow color value to respective metadata record **rule name** or **api name of validation rule** and update/add these condition in validation Rule

IF(  
  
AND(  
$CustomMetadata.CAP\_CC\_Bypass\_Rule\_Engine\_\_mdt.<rulename in metadata/API name of validation in metadata>.CAP\_CC\_Disable\_Rule\_\_c,  
**OR(  
CONTAINS($CustomMetadata.CAP\_BRC\_\_Bypass\_Rule\_Engine\_\_mdt.** .<rulename in metadata/API name of validation in metadata>**.CAP\_BRC\_\_Users\_\_c, $User.Id),  
CONTAINS($CustomMetadata.CAP\_BRC\_\_Bypass\_Rule\_Engine\_\_mdt.** .<rulename in metadata/API name of validation in metadata>**.CAP\_BRC\_\_Roles\_\_c, $UserRole.Id)  
)),**

null, **“Execution condition here”)**

### Triggers

Please include below line at start of each apex trigger either while creating **new** trigger or if its **existing** trigger then update it and add the below script.

**if(ByPassRuleUtility.** fetchMetadataAccess **(userinfo.getuserid(),'TriggerName') || ByPassRuleUtility.** fetchMetadataAccess **(userinfo.getUserRoleId(),'TriggerName')){**

**// trigger logic**

**}**

**'TriggerName' 🡪** this is API name of trigger

 /\*\* @description fetchMetadataAccess this must be used in apex triggers. if logged users Id or role Id matches to the bypass metadata apex trigger records data then return true else false.  
     \*  @param loggedInUser  
     \*  @param ruleName  
     \*  @return boolean  
     \*/  
    public static boolean fetchMetadataAccess(String loggedInUser, String ruleName){  
        system.debug('new learn ' + ruleName);  
        Boolean flag = false;  
        List<Bypass\_Rule\_Engine\_\_mdt> lstAccess = new List<Bypass\_Rule\_Engine\_\_mdt> ();  
        lstAccess = queryBypassRuleMetadata(lstAccess, ruleName);

 if(loggedInUser.length() >= 18){   
            loggedInUser = loggedInUser.substring(0, loggedInUser.length()-3);  
        }  
        if(lstAccess != Null && !lstAccess.isEmpty()){             
            flag = returnFlag(lstAccess,ruleName,loggedInUser );  
        }//if  
        return flag;       
    }

 /\*\* @description returnFlag called from fetchMetadataAccess and return true if logged in usersId/ or roles Id matches with metadata records user or role Id.  
     \*  @param lstAccess  
     \*  @param ruleName  
     \*  @param loggedInUser  
     \*  @return Boolean  
     \*/   
    public static boolean returnFlag(List<Bypass\_Rule\_Engine\_\_mdt> lstAccess, String ruleName, String loggedInUser){  
        for(Bypass\_Rule\_Engine\_\_mdt objAccess : lstAccess ){  
            if(ruleName == objAccess.Rule\_Name\_\_c &&   
            objAccess.Disable\_Rule\_\_c &&  
            ((objAccess.Users\_\_c != null &&  
            (string.valueOf(objAccess.Users\_\_c).contains(loggedInUser))) ||  
            (objAccess.Roles\_\_c != null &&  
            (string.valueOf(objAccess.Roles\_\_c).contains(loggedInUser)))  
            )){  
                return true;  
            }//if                
        }//for  
        return false;  
    }