

Microsoft Dynamics Qualification Exam

SILVJA HOXHAJ

Crm url : <https://org97372cbd.crm4.dynamics.com/>

BUSINESS SCENARIO 1

In the initial phase of business scenario 1, I commenced by establishing the Skill entity, where I incorporated a proficiency level field, offering options ranging from beginner to expert to accurately gauge skill levels. Furthermore, recognizing the intrinsic relationship between skills and users, I implemented a many-to-many relationship between Skill and User entities. To facilitate efficient management and association, I augmented this relationship by adding a subgrid named "Associated Employees" within the Skill entity. This subgrid empowers users to conveniently select and designate which employees possess specific skills, enhancing the organization's capability to manage skill sets effectively across its workforce.

Skill Name

C#

Proficiency Level

Expert

Associated Employee

Add Existing User

Refresh

<input type="radio"/> Full Name ↑ ▾	Site ▾	Business Unit ▾	Title ▾	Position ▾	Main Phone ▾
<input type="radio"/> silvia hoxhaj		org97372cbd			0692554229

1 - 1 of 1

Page 1

Proficiency Level

Expert

--Select--

Beginner

Intermediate

Proficient

Advanced

Expert

Associated Employee

☐ Full Name ↑ ▾

- LFT
- Users

Skills

Agents

Work Orders

Agreements

In addition, to ensure accessibility and visibility of the Skill entity within the system, I incorporated it into the site map

SH silvia hoxhaj - Saved
User - User

Summary Details Administration Related

Skills

AI	API Integration Advanced	
C#	C# Expert	
Ja	JavaScript Proficient	

1 - 3 of 3 |< < Page 1 > >

I've incorporated a subgrid beneath the user details section on the main form. This subgrid displays the skills possessed by the user.

Regarding the aspect of identifying employees possessing specific skills to add them to the sales team within opportunities, I've introduced a new tab on the main form of opportunities titled "Sales Team." Within this tab, I've integrated a subgrid displaying a comprehensive view of employee skills alongside their respective details. Through this subgrid, users have the capability to filter, select and add desired employees to the sales team.

5 Café Duo Espresso Machines for Fabrikam - Saved
Opportunity - Sales Insights

Fabrikam, Inc. Account Est. close date Est. revenue SH silvia hoxhaj Owner

Lead to Opportunity Sal... Active for 8 days

Qualify Contracting Develop (37 Hrs) Propose Close

Summary Sales Team Relationship analytics Quotes Products Related



Full Name (Associated Employee)	Skill Name	Proficiency Level
silvia hoxhaj	API Integration	Advanced
silvia hoxhaj	C#	Expert
# AppDeploymentOrchestration	Database Management	Advanced
# AriaMdlExporter	JavaScript	Proficient

1 - 4 of 5 (1 Selected) |< < Page 1 > >

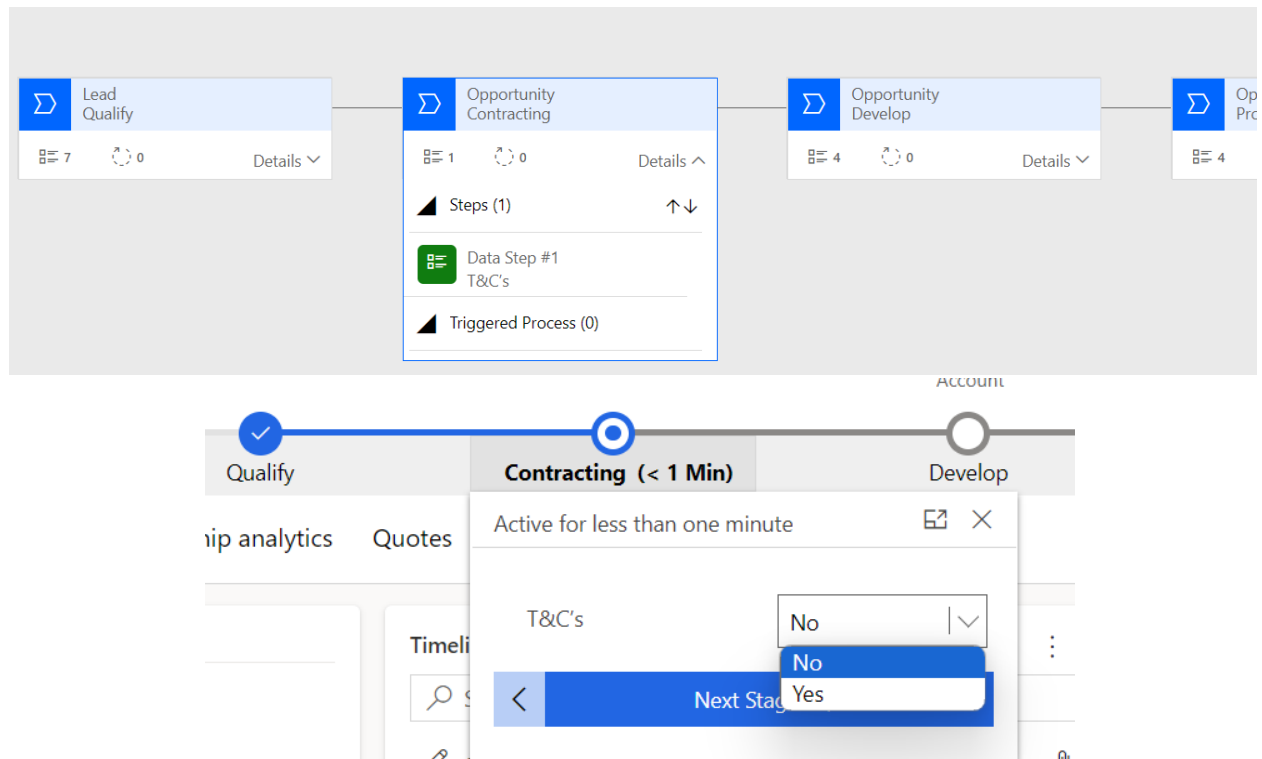
The next step involves the creation of a button that dynamically appears when a user is selected to be added to the sales team. This button, labeled "Add to the Sales Team," will facilitate the process of adding the selected user to the team upon clicking. However, due to limitations with the trial version of CRM, the Button Workbench tool cannot be utilized.

BUSINESS SCENARIO 2

In business scenario 2, the initial step involved the creation of the agreement entity along with its associated fields.

Agreement Name	*	---
Agreement Start Date	---	
Agreement End Date	---	
Account	---	
Agreement Type	---	

During the progression of the business process flow from lead to opportunity, a pivotal addition was made to the opportunity stage, specifically introducing a new stage named "Contracting." This stage was designed to accommodate the inclusion of Terms and Conditions (T&Cs) as a field within the opportunity entity. The T&Cs field was configured as an option set, allowing users to indicate their choice with a binary selection of "Yes" or "No".



Within the Agreement entity, a significant enhancement was implemented through a plugin function titled "UpdateOpportunities." This function was specifically designed to automate the updating of the T&Cs field to "Yes" under certain conditions. Notably, this function was invoked during both the creation and update processes of agreements falling under the category of "onboarding." By seamlessly integrating this functionality, the system ensures that the T&Cs field is consistently updated to reflect affirmative status.

2 references

```
private void UpdateOpportunities(IOrganizationService service, EntityReference account)
{
    QueryExpression query = new QueryExpression("opportunity");
    query.ColumnSet = new ColumnSet("parentaccountid");

    FilterExpression filter = new FilterExpression(LogicalOperator.And);
    ConditionExpression accountCondition = new ConditionExpression("parentaccountid", ConditionOperator.Equal, account.Id);
    filter.AddCondition(accountCondition);

    query.Criteria = filter;

    EntityCollection opportunities = service.RetrieveMultiple(query);

    foreach (var opportunity in opportunities.Entities)
    {
        Entity updatedOpportunity = new Entity(opportunity.LogicalName, opportunity.Id);

        updatedOpportunity["lft_tcs"] = true;
        service.Update(updatedOpportunity);
    }
}
```

Create:

```
if (agreementType == 1)
{
    if (agreements.Entities.Count > 0)
    {
        throw new InvalidPluginExecutionException($"An agreement of type {(agreementType == 1 ? "Onboarding" : "NDA")} already exists for this account.");
    }
    UpdateOpportunities(service, targetEntity.GetAttributeValue<EntityReference>("lft_account"));
}
```

Update :

```
if (agreementType == 1)
{
    if (agreements.Entities.Count > 1)
    {
        throw new InvalidPluginExecutionException($"An agreement of type {(agreementType == 1 ? "Onboarding" : "NDA")} already exists for this account.");
    }

    if (accountReference != null)
    {
        UpdateOpportunities(service, accountReference);
    }
}
```

Within the Agreement plugin, an additional logic layer was integrated to enforce a business rule preventing the creation of agreements labeled as "Onboarding" or "NDA" if an existing agreement of the same type already exists for the associated account. This enforcement spans across both the creation and update processes of agreements.

New Agreement - Unsaved

General

Agreement Name * test

Agreement Start Date 4/8/2024

Agreement End Date 4/9/2024

Account Alpine Ski House

Agreement Type Onboarding

Business Process Error

An agreement of type Onboarding already exists for this account.

Show Details

OK


BUSINESS SCENARIO 3

Initially, I established an agent entity encompassing all requisite fields, followed by the creation of a work order along with its associated fields.

New Agent

New Tab General

Agent Name* ---

Birthday --- 

Phone Number ---

Agent Availability

Is Scheduled Monday

No

Is Scheduled Tuesday

No

Is Scheduled Wednesday

No

Is Scheduled Thursday

No

Is Scheduled Friday

No

No

Yes

New Work Order


General


Work Order Number ---

Assigned Agent ---

Duration ---

Scheduled On ---

Start time --- 

End time --- 

Description ---

By integrating functionality within the WorkOrder.cs plugin, I enabled the system to verify the availability of the assigned agent on the specified day outlined in the work order during both creation and update processes.


```
private void Create(IOrganizationService service, Entity target)
{
    if (target.Attributes.Contains("lft_assignedagent"))
    {
        Entity getAssignedAgent = service.Retrieve(target.GetAttributeValue<EntityReference>("lft_assignedagent").LogicalName, target.GetAttributeValue<EntityReference>("lft_assignedagent").Id, new ColumnSet(true));
        if (getAssignedAgent != null)
        {
            string AgentName = getAssignedAgent.GetAttributeValue<string>("lft_agentname");
            bool WorksOnMonday = getAssignedAgent.GetAttributeValue<bool>("lft_isscheduledmonday");
            bool WorksOnTuesday = getAssignedAgent.GetAttributeValue<bool>("lft_isscheduledtuesday");
            bool WorksOnWednesday = getAssignedAgent.GetAttributeValue<bool>("lft_isscheduledwednesday");
            bool WorksOnThursday = getAssignedAgent.GetAttributeValue<bool>("lft_isscheduledthursday");
            bool WorksOnFriday = getAssignedAgent.GetAttributeValue<bool>("lft_isscheduledfriday");
            int scheduleDay = target.GetAttributeValue<OptionSetValue>("lft_scheduledon").Value;

            bool ItMatches = true;
            switch (scheduleDay)
            {
                case 1:
                    ItMatches = WorksOnMonday;
                    break;
                case 2:
                    ItMatches = WorksOnTuesday;
                    break;
                case 3:
                    ItMatches = WorksOnWednesday;
                    break;
                case 4:
                    ItMatches = WorksOnThursday;
                    break;
                case 5:
                    ItMatches = WorksOnFriday;
                    break;
            }

            if (!ItMatches)
            {
                throw new InvalidPluginExecutionException("The Agent " + AgentName + " isn't available on that day");
            }
        }
    }
}
```

New Work Order - Unsaved

General


Work Order Number	1
Assigned Agent	 Silvia Hoxhaj
Duration	1dite
Scheduled On	Friday
Start time	4/8/2024
End time	4/8/2024
Description	test test

Business Process Error

The Agent isn't available on that day

[Show Details](#)

OK

 5:00 PM

BUSINESS SCENARIO 4

In business scenario 4, I initially addressed system requirements by making email a business required field, thereby ensuring that the creation of a lead without an email became impossible.

Email * ---

Subsequently, I refined the phone number format by incorporating prefixes and specifying the exact length required for a valid entry.

The screenshot shows the 'Field Properties' dialog box with the 'Controls' tab selected. The 'Input mask (retired)' control is selected, and its properties are displayed below.

Control	Web	Phone	Tablet
Text Box (default)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Input mask (retired)	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>

[Add Control...](#)

Input mask (retired)

Property	Value
Field *	telephone1
Mask *	+355000000000 (SingleLine.Text)
Mask Description	

Mask (required)
Compatible types: SingleLine.Text
Enter the mask to use for validating data as the user types

☐ Hide Default Control

OK Cancel

Later, I implemented a duplicate detection rule to identify leads sharing identical business phone numbers, thereby enhancing data integrity and streamlining lead management processes.

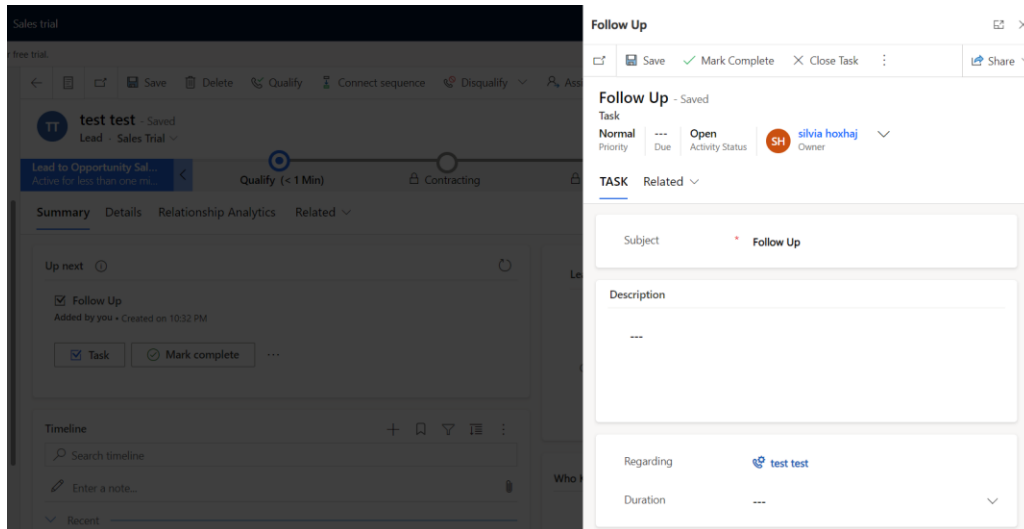
The screenshot shows the 'Duplicate Detection Rule: New Duplicate Detection Rule' configuration page in Microsoft Dynamics 365. The page is titled 'Duplicate Detection Rule: New Duplicate Detection Rule' and is located at 'org97372cbd.crm4.dynamics.com/tools/duplicatedetection/duplicatedetectionrules/edit.aspx'. The 'General' tab is active, showing the rule's name, description, and criteria. The rule is named 'Check leads with the same business phone' and has a status of 'Unpublished'. The description field is empty. The 'Duplicate Detection Rule Criteria' section shows 'Base Record Type' set to 'Lead' and 'Matching Record Type' set to 'Lead'. The 'Case-sensitive' checkbox is unchecked, and the 'Exclude inactive matching records' checkbox is checked. A table below lists the criteria: 'Business Phone' with 'Exact Match' and 'No. of Characters' set to 1. The 'Status' is 'New'.

Field	Criteria	No. of Characters	Ignore Blanks
Business Phone	Exact Match	1	<input type="checkbox"/>

Following the establishment of a lead, I proposed two methods to seamlessly generate a follow-up task. The first involves employing a plugin, which would trigger the creation of a follow-up task immediately upon lead creation, ensuring prompt action. Alternatively, I considered implementing a workflow to automate the task creation process, providing another avenue to streamline operations and ensure that follow-up actions are promptly initiated upon lead creation. Both approaches offer efficient solutions to enhance task management within the system.

The screenshot shows the 'Process: Follow up Task' configuration page in Microsoft Dynamics 365. The page is titled 'Process: Follow up Task' and is located at 'org97372cbd.crm4.dynamics.com/tools/workflowdesigner/workflowdesigner/edit.aspx'. The 'General' tab is active, showing the process name, activate as, and available to run options. The process name is 'Follow up Task' and it is activated as a 'Process'. The 'Available to Run' section shows three options: 'Run this workflow in the background (recommended)', 'As an on-demand process', and 'As a child process'. The 'Options for Automatic Processes' section shows 'Scope' set to 'Organization' and 'Start when' set to 'After'. The 'Execute as' section shows 'The owner of the workflow' selected. The 'Workflow Log Retention' section shows 'Keep logs for workflow jobs that encountered errors' checked. The 'Entity' is 'Lead' and the 'Category' is 'Workflow'.

Field	Value
Process Name	Follow up Task
Activate As	Process
Available to Run	<input type="checkbox"/> Run this workflow in the background (recommended) <input type="checkbox"/> As an on-demand process <input type="checkbox"/> As a child process
Options for Automatic Processes	Scope: Organization Start when: After Execute as: The owner of the workflow
Workflow Log Retention	<input checked="" type="checkbox"/> Keep logs for workflow jobs that encountered errors



In conclusion, to further optimize lead management processes, I ensured that upon lead creation, the topic field is automatically populated with the date of creation. This enhancement was achieved through the implementation of a plugin

```
public void Execute(IServiceProvider serviceProvider)
{
    IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));
    IOrganizationServiceFactory serviceFactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));
    IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

    try
    {
        Entity target = null;
        target = (Entity)context.InputParameters["Target"];

        string topic = target.GetAttributeValue<string>("subject");
        topic += " " + DateTime.UtcNow.ToString("dd/MM/yyyy");

        if (context.Stage == 40)
        {
            Entity task = new Entity("task");
            task["subject"] = "Follow Up";
            task["regardingobjectid"] = new EntityReference(target.LogicalName, target.Id);
            service.Create(task);

            target["subject"] = topic;
            service.Update(target);
        }
    }
    catch (Exception ex)
    {
    }
}
```

Topic * test 08/04/2024

Integration, ribbon buttons, action, business rules, and Power Automate process automation are powerful customization and automation capabilities available in Microsoft Dynamics 365.

1. **Integration:** Dynamics 365 offers robust integration capabilities to connect with external systems, enabling seamless data exchange and workflow automation. Integrations can be achieved using various methods such as RESTful APIs, Azure Integration Services, and custom integration code, ensuring interoperability and data consistency across different platforms.
2. **Ribbon Buttons:** Ribbon buttons in Dynamics 365 enable users to trigger specific actions or workflows manually. By customizing the ribbon interface, organizations can provide users with easy access to frequently used functionalities, enhancing user productivity and efficiency within the system.
3. **Action:** Actions in Dynamics 365 allow for the creation of custom operations that can be invoked from various components within the system, including ribbon buttons, workflows, and plugins. Actions can accept input parameters and return output values, providing flexibility in implementing specific business logic or executing complex operations tailored to organizational needs.
4. **Business Rules:** Business rules offer a simple yet powerful way to implement logic without writing code. Organizations can use business rules to enforce data validation, automate calculations, and control the behavior of forms based on predefined conditions. Business rules are user-friendly and can be configured by power users or administrators directly within the system.
5. **Power Automate Process Automation:** Power Automate (formerly Microsoft Flow) is a cloud-based workflow automation platform that seamlessly integrates with Dynamics 365 and other Microsoft services. With Power Automate, organizations can automate repetitive tasks, streamline business processes, and trigger actions based on predefined triggers or events, enhancing operational efficiency and productivity.