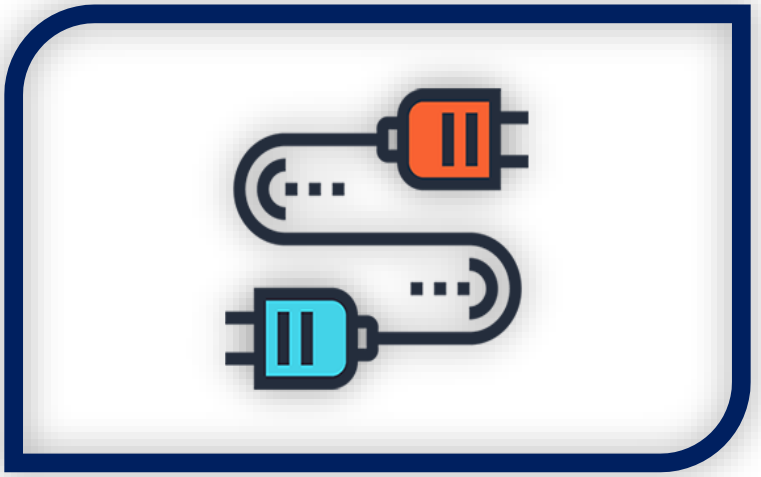
Dynamics 365 – Custom Plugin Development - Custom Development Practice



Cognitive Convergence
http://www.cognitiveconvergence.com
+1 4242530744
shahzad@cognitiveconvergence.com

Cognitive Convergence is Subject Matter Expert in Office 365, Dynamics 365, SharePoint, Project Server, Power Platform: Power Apps-Power BI-Power Automate-Power Virtual Agents. Our Microsoft Dynamics 365 Consulting, Development, Customization, Integration services and solutions, can help companies maximize business performance, overcoming market challenges, achieving profitability and providing best customer service.

Contents

OBJECTIVES

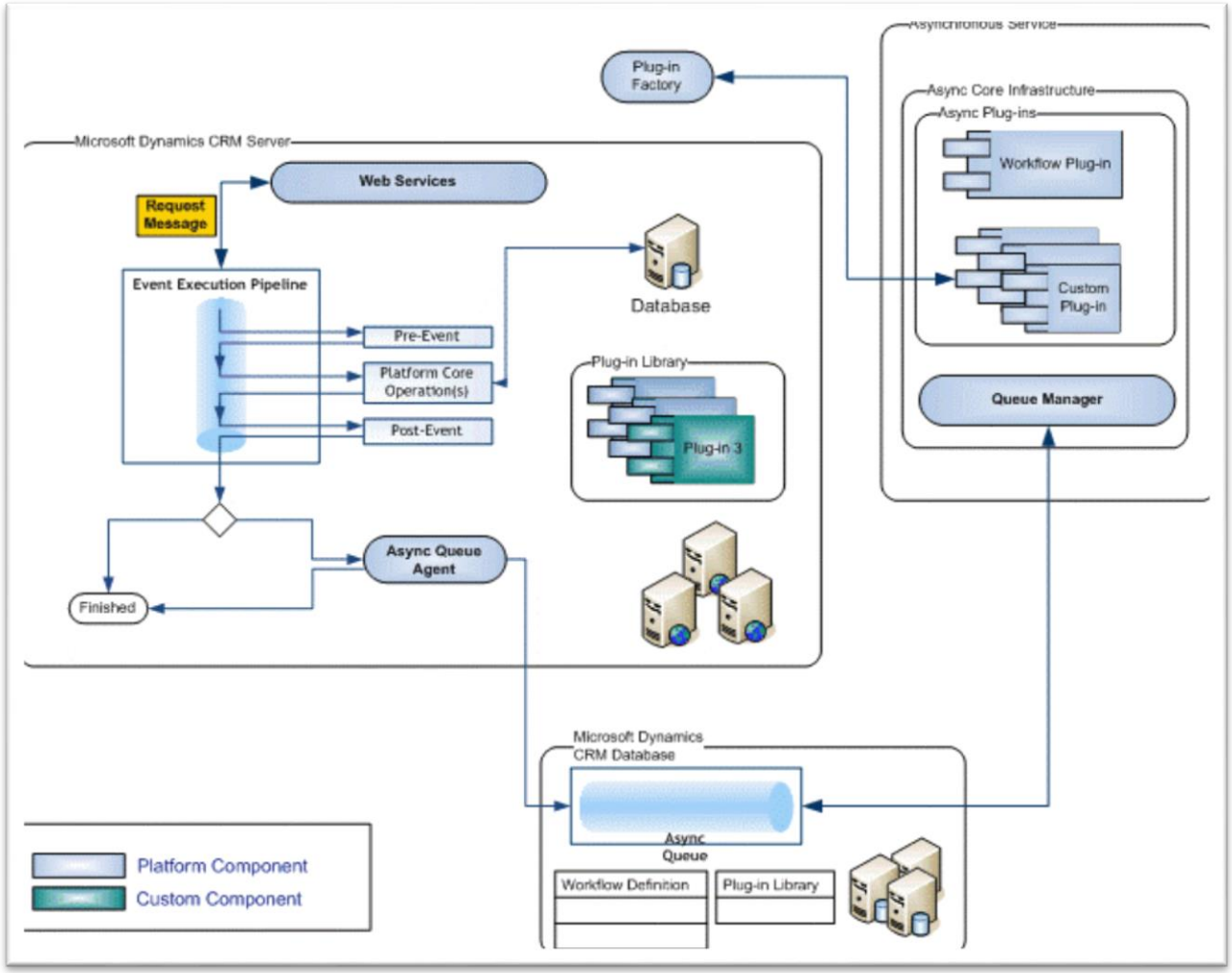
Dynamics 365 plugin development is an important development trend, followed for custom business logic, to modify or augment the standard behavior of Dynamics 365 platform. This case study covers a custom plugin developed for generating a task in activity against the created account. Dynamics 365 custom plugin for on-premises is also discussed.

DYNAMICS PLUGIN

A plug-in is a custom business logic (code) that integrates with Microsoft Dynamics CRM to modify or extend or argument the standard behavior of the platform. Another way to think about plug-ins is that they are handlers for events fired by Dynamics 365 Customer Engagement. Plug-ins act as event handlers and are registered to execute on a particular event in CRM. Plugins are written in either C# or VB and can run either in synchronous or asynchronous mode.

POSSIBLE SCENARIOS FOR WRITING PLUGINS

- Some scenarios to write a plugin, are to execute some business logic such as updating certain fields of a record or updating related records, etc. when the purpose is to create or update a CRM record.
- To call an external web service on certain events such as saving or updating a record.
- To dynamically calculate the field values when any record is opened.
- To automate processes such as sending e-mails to the customers on certain events in CRM.



ON-PREMISES PLUG-IN DEVELOPMENT

A plug-in is custom business logic (code) that you can integrate with Dynamics 365 Customer Engagement (on-premises) to modify or augment the standard behavior of the platform. Another way to think about plug-ins is that they are handlers for events fired by Dynamics 365 Customer Engagement (on-premises). You can subscribe, or register, a plug-in to a known set of events to have your code run when the event occurs.

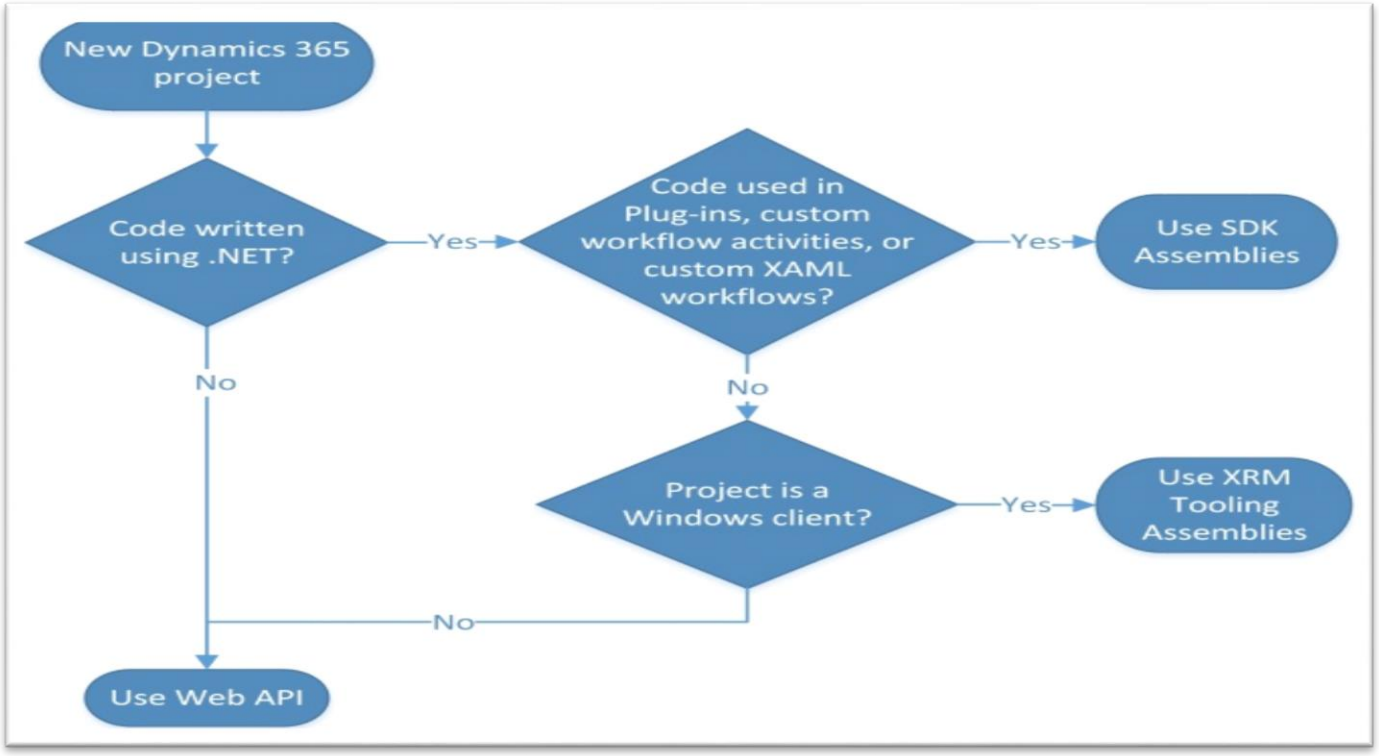
Reference variable of On-premises plug-in

In an on-premises environment where a full trust plug-ins are executed within the same app domain, don't expect that a variable that refers to data included in the plug-in context will maintain a reference to the object.

When data is passed into the event pipeline, the data is serialized and de-serialized to create a new object instances. The object instances do not refer to the same memory address. Any changes to the object in the plug-in execution pipeline will not be reflected in an object instance that was passed into an operation in the pipeline.

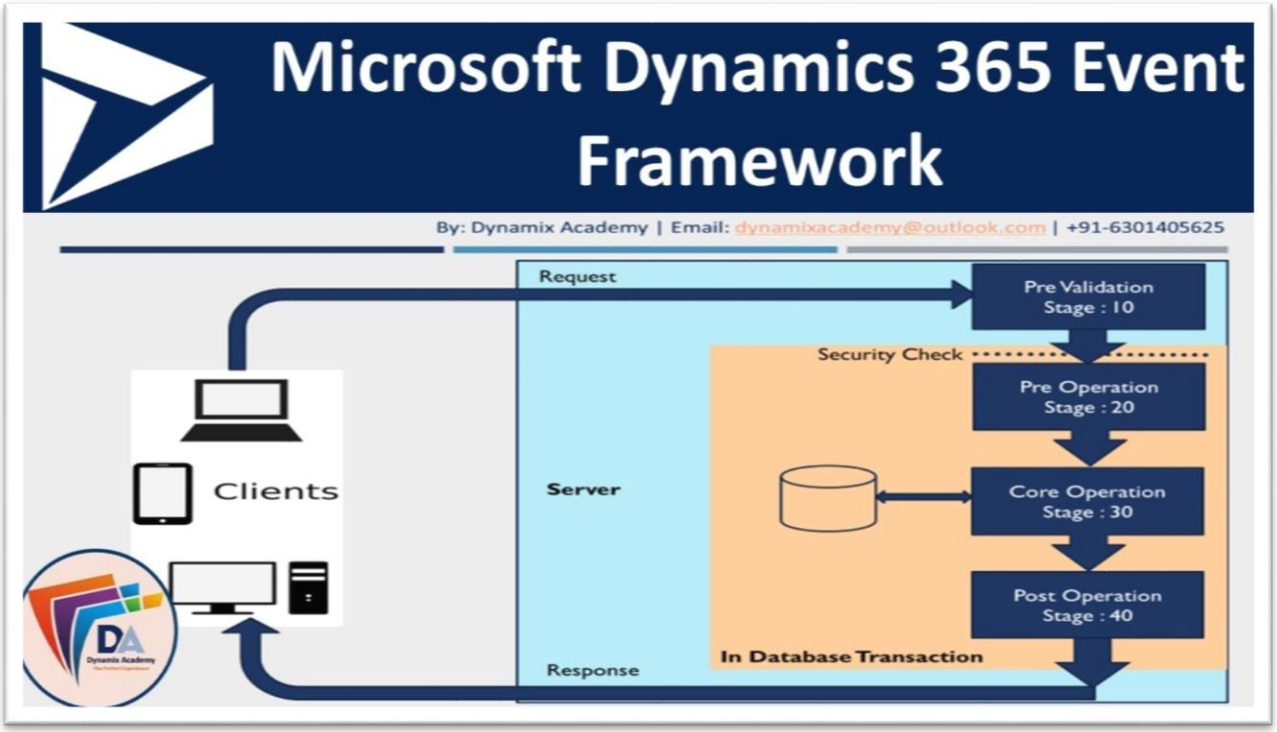
Example

If any code within a plug-in modifies a QueryExpression that is included in a RetrieveMultipleRequest, that change will not be reflected in the original QueryExpression instance variable that was given with the RetrieveMultiple request. The QueryExpression object properties may be modified during the data retrieval process within the pipeline. The QueryExpression.PageInfo property, for example, will be changed as part of the query execution. Examining the original QueryExpression variable that was used with the RetrieveMultipleRequest will not reveal these changes



EVENT FRAMEWORK

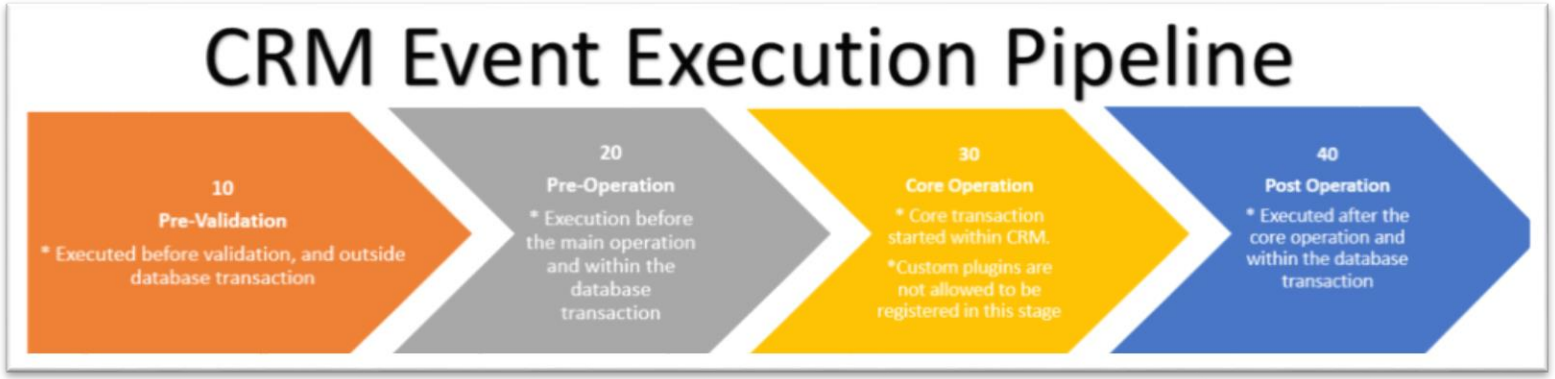
The Event Processing Framework in CRM processes the synchronous and asynchronous plugin requests by passing it to the event execution pipeline. Whenever an event triggers a plugin logic, a message is sent to the CRM Organization Web Service where it can be read or modified by other plugins or any core operations of the platform.



PLUGIN PIPELINE STAGES

The entire plugin pipeline is divided into multiple stages on which user can register custom business logic. The pipeline stage specified indicates at which stage of the plugin execution cycle, plugin code runs. Out of all the specified pipeline stages in the following table, registration of custom plugins only on be done on Pre- and Post-events. The process of Register plugins on Platform Core Main Operations is not available yet.

To understand the pipeline more in detail, just look at the table given below.

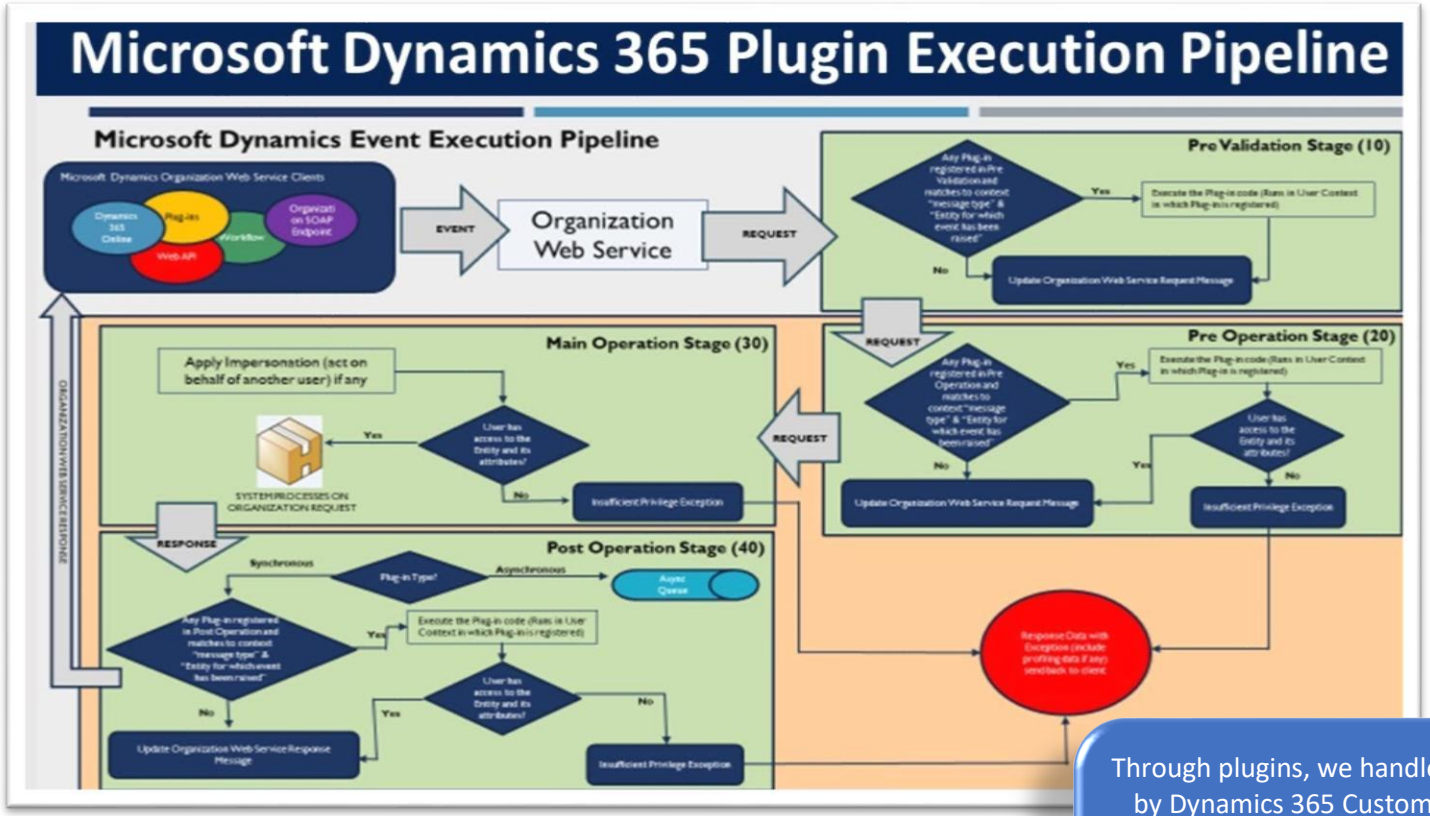


Event	Stage Name	Stage number	Description
Pre-Event	Pre-validation	10	Stage in the pipeline for plug-ins that are to execute before the main system operation. Plug-ins registered in this stage may execute outside the database transaction.
Pre-Event	Pre-operation	20	Stage in the pipeline for plug-ins that are to execute before the main system operation. Plugins registered in this stage are executed within the database transaction.
Platform Core Operation	MainOperation	30	In the transaction, the main operation of the system, such as create, update, delete, and so on. No custom plug-ins can be registered in this stage. For internal use only.
Post-Event	Post-operation	40	Stage in the pipeline for plug-ins which are to execute after the main operation. Plug-ins registered in this stage are executed within the database transaction.

Whenever the CRM application invokes an event (like saving or updating a record), the following sequence of actions takes place −

- The event triggers a Web service call and the execution is passed through the event pipeline stages (pre-event, platform core operations, post-event).
- The information is internally packaged as an OrganizationRequest message and finally sent to the internal CRM Web service methods and platform core operations.

- The OrganizationRequest message is first received by pre-event plugins, which can modify the information before passing it to platform core operations. After the platform core operations, the message is packaged as OrganizationResponse and passed to the post-operation plugins. The post-operations plugins can optionally modify this information before passing it to the async plugin.
- The plugins receive this information in the form of context object that is passed to the Execute method after which further processing happens.
- After all the plugin processing completes, the execution is passed back to the application which triggered the event.



Through plugins, we handle the events fired by Dynamics 365 Customer Engagement

Cognitive Convergence
http://www.cognitiveconvergence.com
+1 4242530744
shahzad@cognitiveconvergence.com

PLUGIN MESSAGES

Messages are the events on which the plugin (or business logic) is registered. For example, registering a plugin on Create Message of Contact entity. This would fire the business logic whenever a new Contact record is created. For custom entities, the following are the supported messages based on whether the entity is user-owned or organization-owned.

Message Name	Ownership Type	Message Availability	Entity Supported Deployment
Assign	User-owned entities only	Server	Server
Create	User-owned and organization-owned entities	Both	Server
Delete	User-owned and organization-owned entities	Both	Server
GrantAccess	User-owned entities only	Server	Server
ModifyAccess	User-owned entities only	Server	Server
Retrieve	User-owned and organization-owned entities	Both	Server
RetrieveMultiple	User-owned and organization-owned entities	Both	Server
RetrievePrincipalAccess	User-owned entities only	Both	Server
RetrieveSharedPrincipalsAndAccess	User-owned entities only	Both	Server
RevokeAccess	User-owned entities only	Server	Server
SetState	User-owned and organization-owned entities	Both	Server
Update	User-owned and organization-owned entities	Both	Server

EXCEPTION HANDLING IN PLUGIN

More often than not, your plugin logic will need to handle run-time exceptions. For synchronous plugins, you can return an InvalidPluginExecutionException exception, which will show an error dialog box to the user. The error dialog will contain the custom error message that you pass to the Message object of the exception object.

If you look at our code, we are throwing the InvalidPluginExecutionException exception in our catch block.

```
throw new InvalidPluginExecutionException(ex.Message);
```

ISERVICEPROVIDER FOR PLUGIN EXECUTION

.Net framework provides a bundle of classes that can be used to develop a plugin to fulfill our business requirements. One of the most important, that is used oftenly for plugins is IServiceProvider. It defines a mechanism for retrieving a service object; that is, an object that provides custom support to other objects. It executes the plug-in code in response to an event.

IServiceProvider is a container for service objects. Contains references to the plug-in execution context. These are discussed below:

ITracingService Interface

The ITracingService enables writing to the tracing log that provides a method of logging run-time trace information for plug-ins. The ITracingService interface provides a way to log plug-in run-time information. This method of logging information is especially useful for sandboxed plug-ins registered with Microsoft Dynamics 365 (online) that cannot otherwise be debugged using a debugger. The tracing information is displayed in a dialog of the Microsoft Dynamics 365 Web application only if an exception is passed from a plug-in back to the platform.

```
C#

public interface ITracingService
```

IPluginExecutionContext Interface

The IPluginExecutionContext provides access to the context for the event that executed the plugin. It defines the contextual information passed to a plug-in at run-time. Contains information that describes the run-time environment that the plug-in is executing in, information related to the execution pipeline, and entity business information.

```
C#

public Microsoft.Xrm.Sdk.ParameterCollection InputParameters { get; }
```

IServiceEndpointNotificationService

Posts the plug-in execution context to the Microsoft Azure Service Bus. Both the Microsoft Azure Service Bus and Microsoft Dynamics 365 must be properly configured before using this service.

```
C#                                                    Copy

public interface IServiceEndpointNotificationService
```

IOrganizationServiceFactory Interface

The **IOrganizationServiceFactory** interface provides access to a service variable that implements the **IOrganizationService** interface (Provides programmatic access to the metadata and data for an organization) which provides the methods that will be use to interact with the service to create the task.

```
C#

public interface IOrganizationServiceFactory
```

TASK CREATION PLUGIN

A use-case identified for creating of a task after a few seconds, against an account entity. When an account entity is created, automatically a task is generated against it, instead of manually adding the task.

Pre-Requisites

Following things are required before starting the actual development of custom plugin.

1. Visual Studio 2017 or above.
2. .Net Framework 4.6.2 or above except .Net Framework 4.7.2.
3. Microsoft Plugin Registration Tool.
4. Microsoft 365 tenant credentials.
5. A Model-Driven App having an account entity.

Dynamics 365 XRM SDK

XRM SDK is basically a kind of data layer access to CRM and hence it only provides functionality related to data access such as Create/Update/Delete/Read. We use this class to create a task in activity entity.

```
C#

using System.ServiceModel;
using Microsoft.Xrm.Sdk;
```

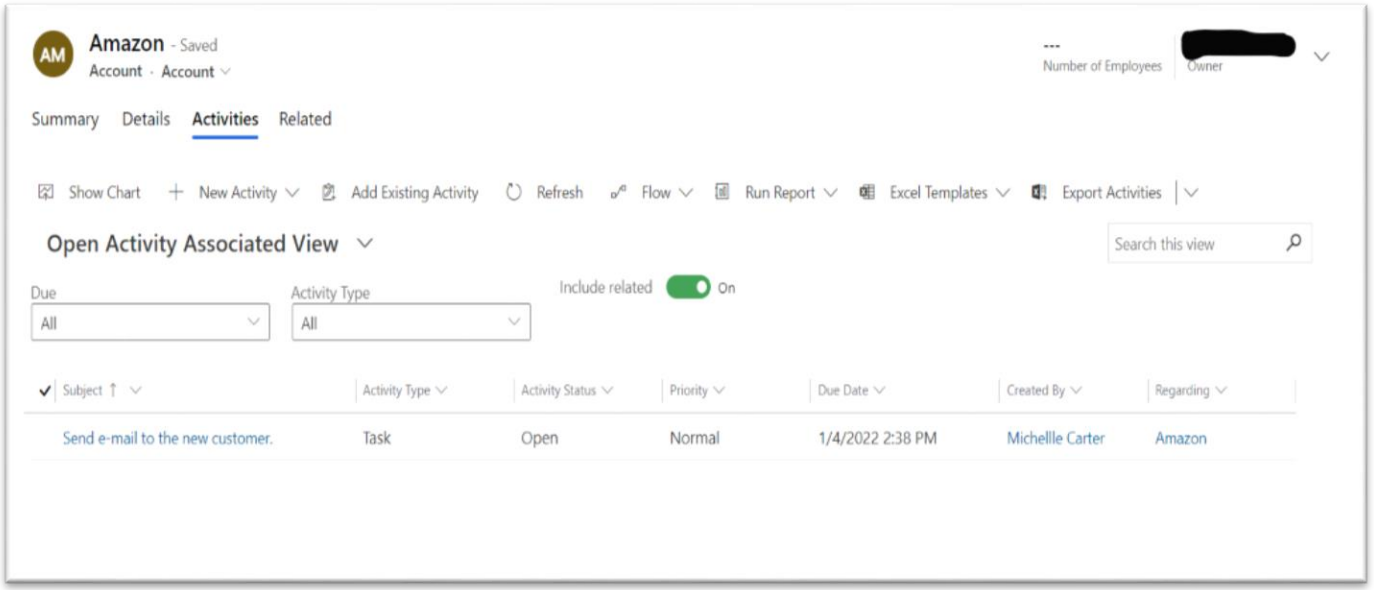
Using services

In this code ITracingSerivce is use to obtain tracing service. IPluginExecutionContext is user to get execution context from the service provider. The InputParameters collection contain all the data passed in the message request. Orgnizatoin service reference and web service calls through IOrgnizationServiceFactory. Then using Entity followup we create a task.

```
try
{
    // Plug-in business logic goes here.
    // Create a task activity to follow up with the account customer in 7 days.
    Entity followup = new Entity("task");
    followup["subject"] = "Send e-mail to the new customer.";
    followup["description"] = "Follow up with the customer. Check if there are any new issues that need resolution.";
    followup["scheduledstart"] = DateTime.Now.AddDays(7);
    followup["scheduledend"] = DateTime.Now.AddDays(7);
    followup["category"] = context.PrimaryEntityName;
    // Refer to the account in the task activity.
    if (context.OutputParameters.Contains("id"))
    {
        Guid regardingobjectid = new Guid(context.OutputParameters["id"].ToString());
        string regardingobjectidType = "account";
        followup["regardingobjectid"] = new EntityReference(regardingobjectidType, regardingobjectid);
    }
    // Create the task in Microsoft Dynamics CRM.
    tracingService.Trace("FollowupPlugin: Creating the task activity.");
    service.Create(followup);

}
```

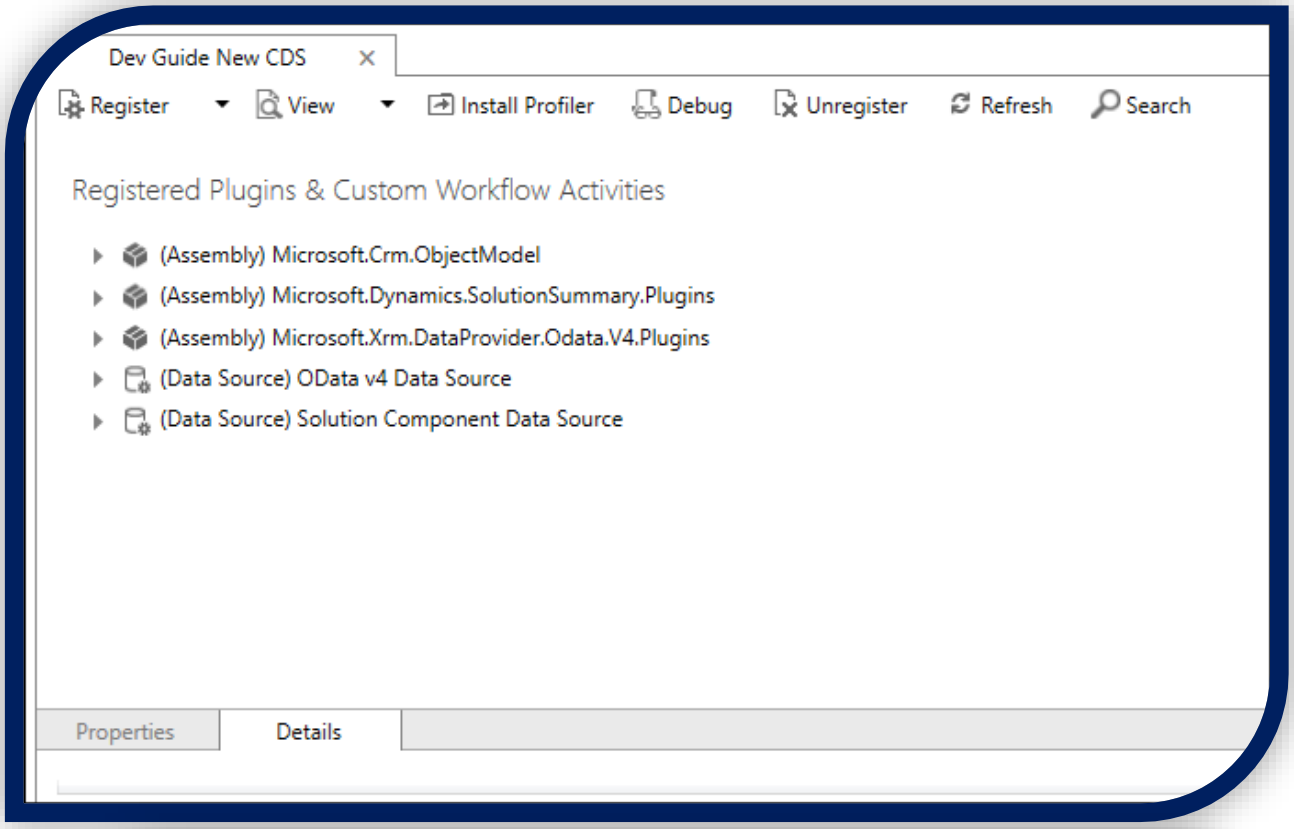
Task is created against account entity.

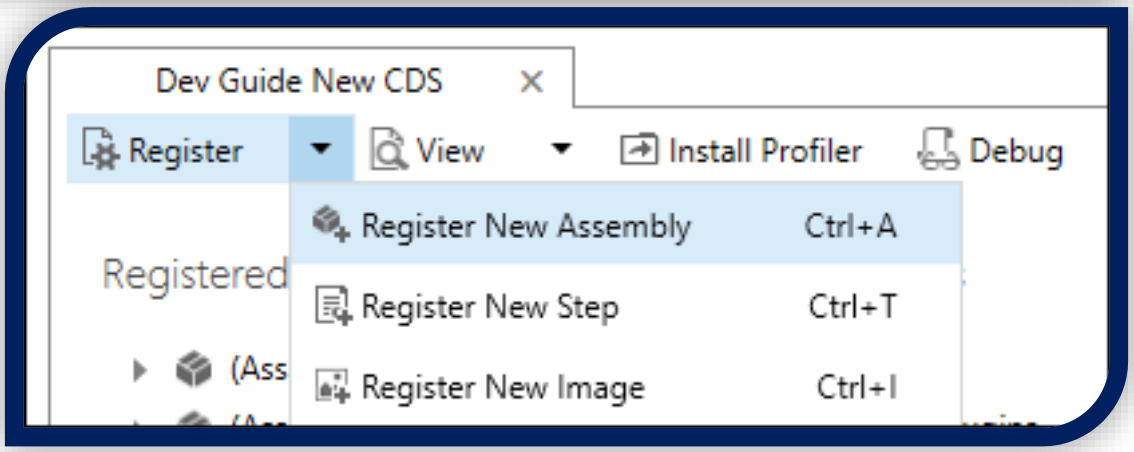


REGISTER PLUG-IN

Plug-in registration tool will be required for plug-in registration process. Tenant Credentials are also required.

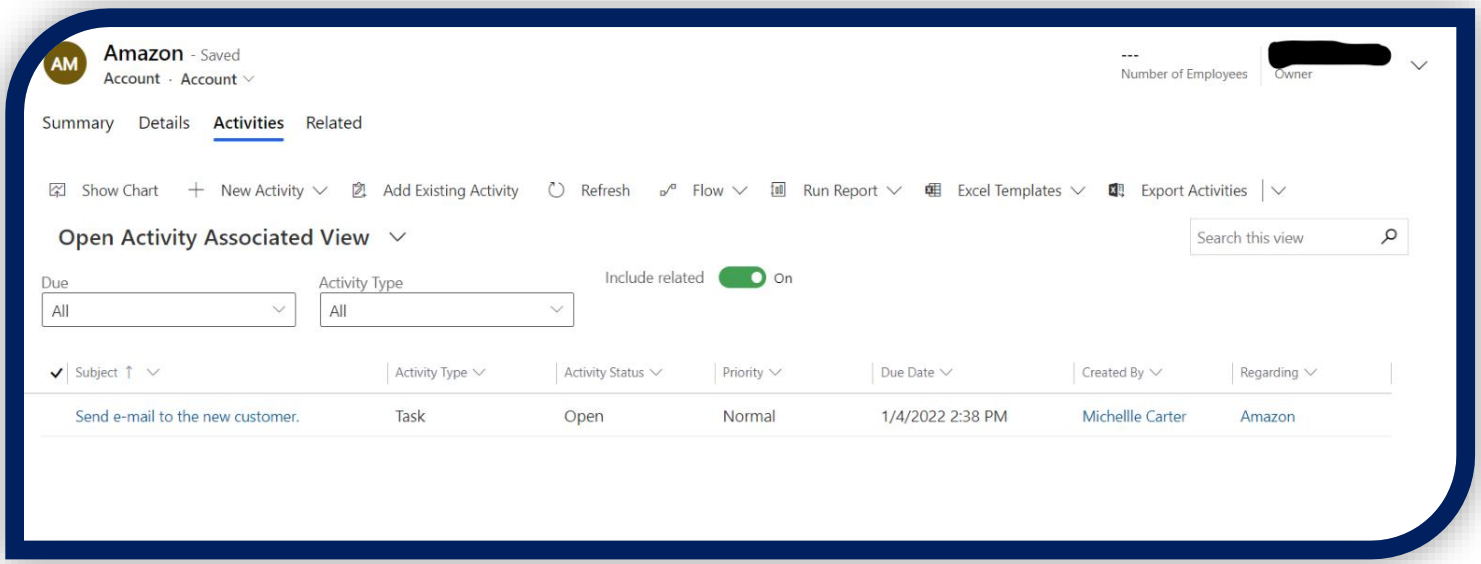
After login we create "**Register New Assembly**" and also "**Register New Assembly".**


Dev Guide New CDS
Register
View
Install Profiler
Debug
Unregister
Refresh
Search
Registered Plugins & Custom Workflow Activities
(Assembly) Microsoft.Crm.ObjectModel
(Assembly) Microsoft.Dynamics.SolutionSummary.Plugins
(Assembly) Microsoft.Xrm.DataProvider.Odata.V4.Plugins
(Data Source) OData v4 Data Source
(Data Source) Solution Component Data Source
Properties
Details


Dev Guide New CDS
Register
View
Install Profiler
Debug
Register New Assembly Ctrl+A
Register New Step Ctrl+T
Register New Image Ctrl+I
Registered

TEST PLUG-IN

1. Open a model-driven app and create an account entity.
2. Within a short time, open the account to verify the creation of the task.



CONCLUSION

In this case study, a brief introduction was discussed about the Dynamics 365 Plugin Development, Possible Scenarios that are used for writing the plugin, Plugin Pipeline process of Dynamic 365, plugin messages and Exception Handling of Dynamic 356.

Our Microsoft Dynamics 365 Consulting, Development, Customization, Integration services and solutions, can help companies maximize business performance, overcoming market challenges, achieving profitability, and providing the best customer care service.

Contact Us

Cognitive Convergence

http://www.cognitiveconvergence.com

+1 4242530744

shahzad@cognitiveconvergence.com