INNONE
innovation one

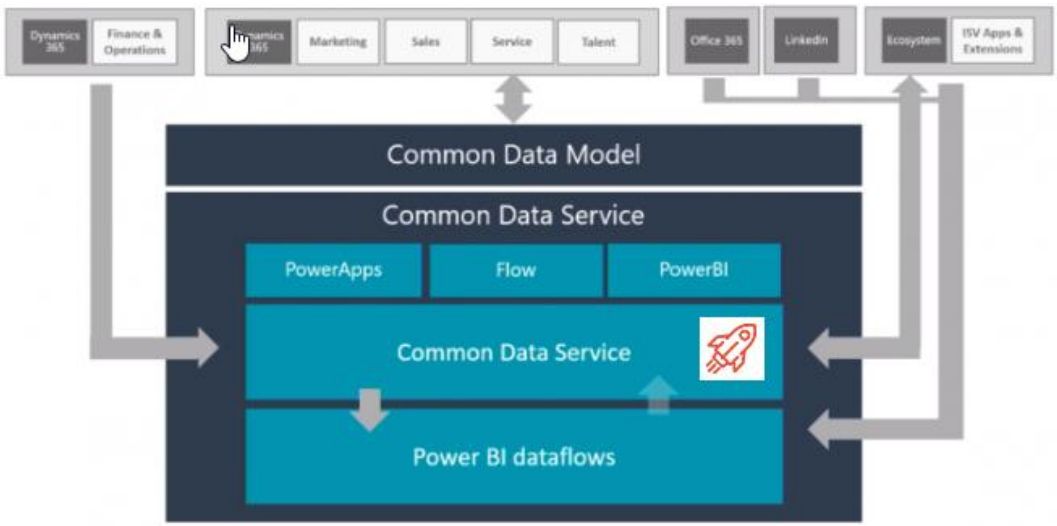
CDS Plugin Development Framework

Innovation One s.r.o

Version 1.3

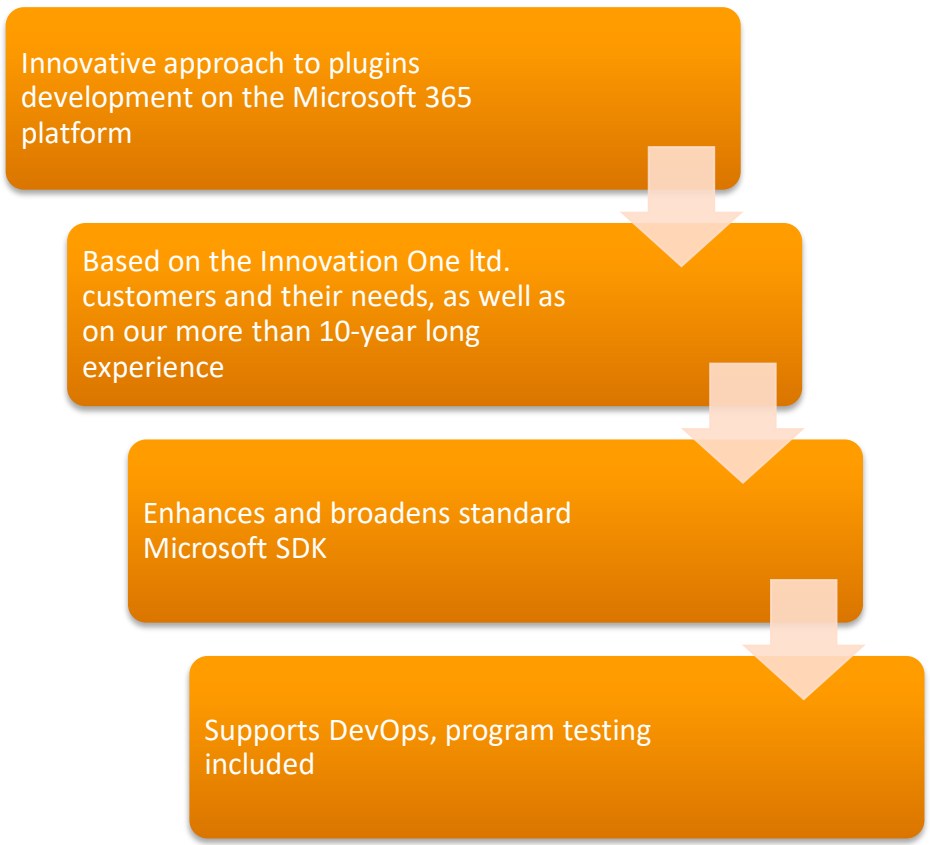
CDS Plugin Development Framework a part of Microsoft Common Data Service

- Structures the customer logic implemented in the form of plugins into independent functional units called Tasks
- Middleware between the implemented logic of custom plugins and standard Microsoft SDK
- Within the MS platform, detailed information is generated to diagnose errors and optimize performance issues
- Leads to clear and sustainable code structure supported with program testing



CDS Plugin Development Framework

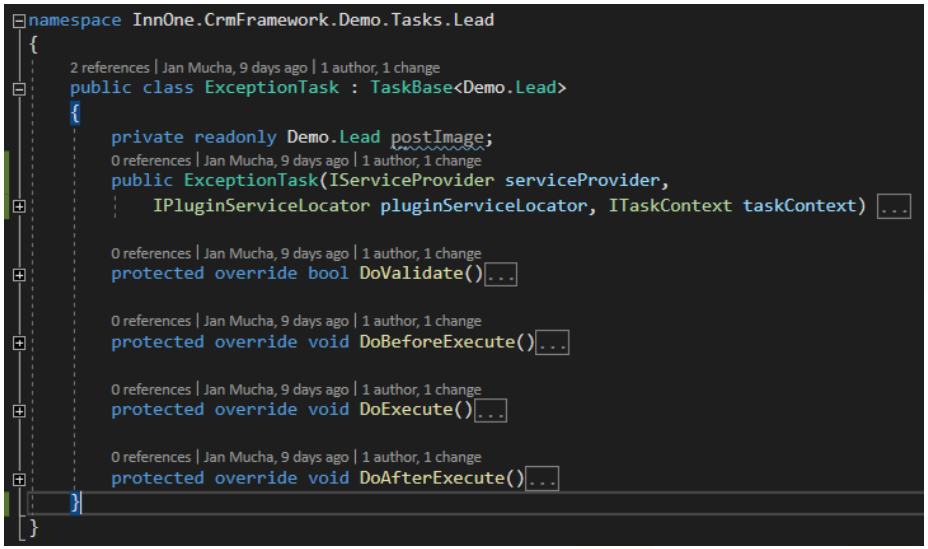
CDS Plugin Development Framework Benefits



- Information logs visualization
- Enables to coordinate analysis with the implemented code
- Tasks division into validation and execution
- In-built logs of launched Tasks
- Well-arranged and clearly defined code structure
- Results in development and management costs decrease

The Basis – Tasks

- Task is the basic logical unit
- Task Name is a unique identifier across analysis, implementation, and diagnostics
- The analyst defines Tasks at the point of functionality design
- Each Task consists of validation and execution parts
  - DoValidate
  - BeforeExecute
  - Execute
  - AfterExecute



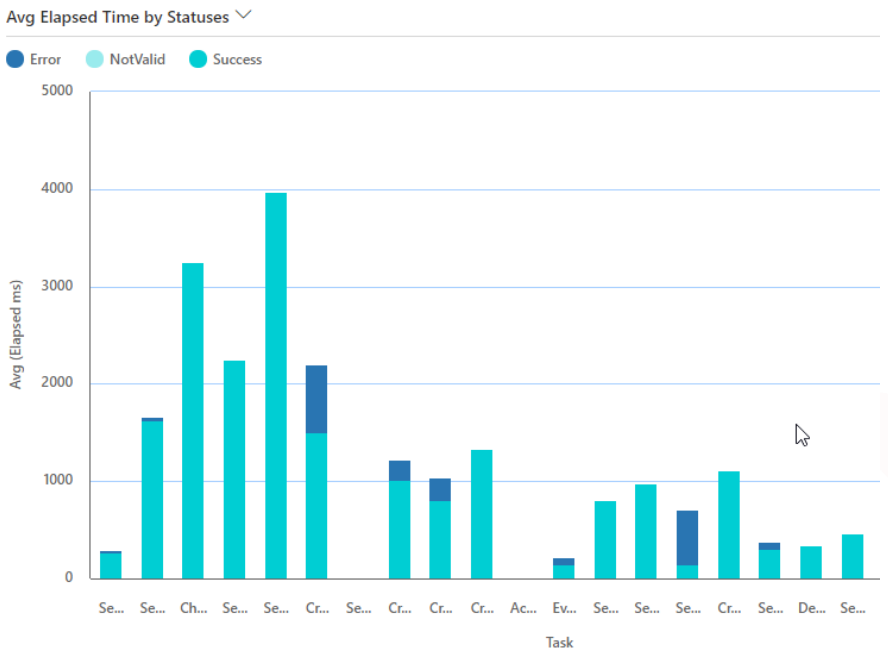
In CDS plugins only Tasks execution remains

```
public class GeneralConsentExecutor : PluginExecutor
{
    public GeneralConsentExecutor(string unsecureConfig, string secureConfig) : base(unsecureConfig, secureConfig)
    {
        RegisterEvents((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Preoperation,
        new[] { "Create", "Update" }, asc_generalConsent.EntityLogicalName, typeof(SetIdConsent));

        RegisterEvents((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Preoperation,
        new[] { "Create", "Update" }, asc_generalConsent.EntityLogicalName, typeof(GeneralConsentName));


        RegisterEvent((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Postoperation,
        "Create", asc_generalConsent.EntityLogicalName, typeof(GenralConsentValidation));

        RegisterEvents((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Postoperation,
        new[] { "Create", "Update" }, asc_generalConsent.EntityLogicalName, typeof(UpdateSubstitueLead));

        RegisterEvent((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Postoperation,
        "Create", asc_generalConsent.EntityLogicalName, typeof(UpdateConsentLead));

        RegisterEvents((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Postoperation,
        new[] { "Create", "Update" }, asc_generalConsent.EntityLogicalName, typeof(ResolveConsentDependence));

        RegisterEvents((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Postoperation,
        new[] { "Create", "Update" }, asc_generalConsent.EntityLogicalName, typeof(DeleteUselessConsent));

        RegisterEvents((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Postoperation,
        new[] { "Create", "Update" }, asc_generalConsent.EntityLogicalName, typeof(CountAllowAddressingAttributesTask));

        RegisterEvents((int)CrmFramework.Core.Enums.SdkMessageProcessingStepStage.Postoperation,
        new[] { "Create", "Update", "Delete" }, asc_generalConsent.EntityLogicalName, typeof(ConsentLog));
    }
```

- All plugins are of the same structure, without logic implementation
- Tasks are grouped into plugins based on either the entity or the functionality
- Tasks are launched in a defined order
- Plugin registration, registration steps as well as information on the versions are stored right in the code

Log System

- Every time a Task is launched, a log with detailed information is automatically generated
- Within the code, there is LogService available providing easily created logs of the required information
- Within an environment the required level of logs can be chosen
- Logs and the information they contain are available in clear charts on the dashboards
- Clicking on the chart a particular information log can be accessed for the purpose of detailed diagnostics



Dashboards – Logs Overview

All Logs Last 5 days

- Dashboard offers comparison of the execution count with the average execution time according to the task status
- Comparing this information helps the user identify any performance issues and design efficient optimization

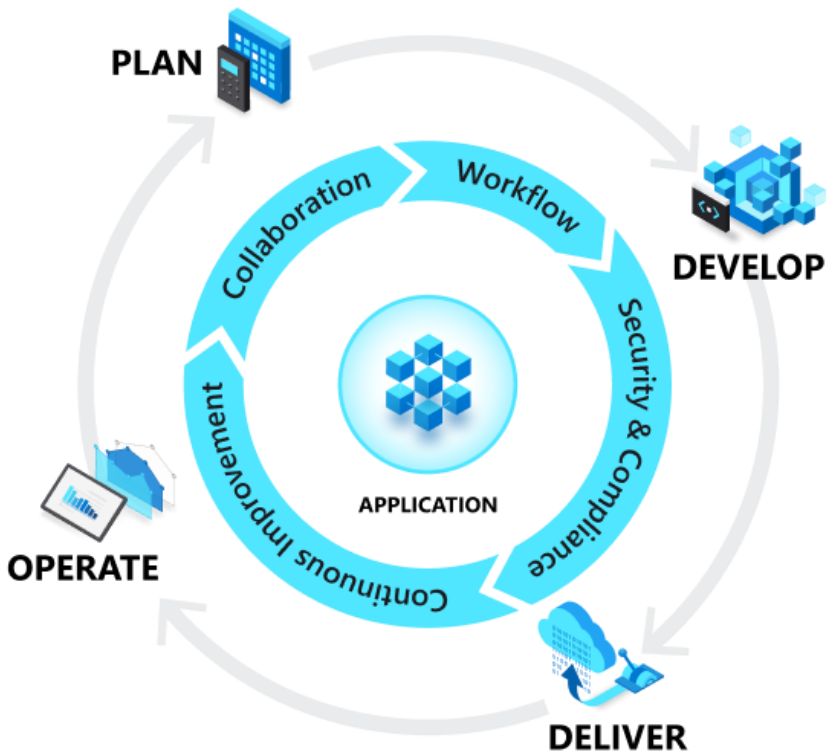
Errors and Fatals Last 5 Days

- Dashboard displays the application current status and informs about errors or performance issues
- These data provide the user with detailed information about errors and help them positively identify the cause and come up with efficient correction

CDS Plugin Development Framework Components

- CDS Plugin Development Framework Solution
  - InnOne Setting Entity
  - InnOne Log Entity
  - InnOne Framework Application
    - Error Log Dashboard Last 5 Days
    - All Log Dashboard Last 5 Days
- NuGet package with a set of DLL libraries
  - Support for plugins creation
  - Support for creation of applications communicating with CDS
  - Support for program tests creation
- Full support of cooperation with *SPKL Framework*

CDS Plugin Development Framework fully supports DevOps



- One-click deployment
- Automated versions deployment across all environments
- At the point of deployment program tests are launched automatically
- Due to automated processes the risk of human error is minimized

Implementation Process

Free preliminary consultation with an Innovation One ltd. specialist

Analysis of the issues at the customer

Key benefits identification of *CDS Plugin Development Framework*

Design of the implementation process based on the customer´s needs

Deployment of *CDS Plugin Development Framework* and personnel training

Clearer development, quality and sustainability enhancement of the solution due to Microsoft Power Platform

INNONE
innovation one

Partner for

digital transformation

Koněvova 2660/141, Žižkov,
130 00 Praha 3
Po-Pá: 8-17 hod
+420 736 213 603
obchod@innone.cz