



Microsoft Dynamics CRM 2013 Extending Training Material

Version 2.0

www.firebrandtraining.com

Course content

Module 0 – Course Content and Plan 5

 Objectives 5

 Course Plan 5

 Course Modules 6

 Resources 7

 Case Study 7

 SDK 7

 Exam 7

 Feedback 10

Module 1 – Extensibility Model and Framework 11

 Objectives 11

 Lesson 1-1 CRM Functionality 11

 Lesson 1-2 XRM Application Framework 12

 Lesson 1-3 Deployment Options 16

 Lesson 1-4 What’s New in CRM 2013 18

 Lesson 1-5 Resources 22

Module 2 –Platform Operations 23

 Objectives 23

 Lesson 2-1 Windows Communications Foundation 23

 Lesson 2-2 Microsoft Dynamics CRM Web Services 26

 Lesson 2-3 Early Bound and Late Bound Classes 29

 Lesson 2-4 Create, Retrieve, Update and Delete Methods 32

 Lesson 2-5 Execute Method 34

 Lesson 2-6 Exceptions 35

 Lesson 2-7 Metadata 36

Module 3 – Querying Data 38

Objectives 38

Lesson 3-1 Queries in Dynamics CRM..... 38

Lesson 3-2 Query Expression 38

Lesson 3-3 QuerybyAttribute..... 39

Lesson 3-4 LINQ 40

Lesson 3-5 FetchXML 41

Lesson 3-6 Filtered Views 42

Lesson 3-7 Reporting Options..... 43

Module 4 – Processes 46

 Objectives 46

 Lesson 4-1 Dynamics CRM Processes 46

 Lesson 4-2 Dynamics CRM Process Steps 49

 Lesson 4-3 Workflows..... 53

 Lesson 4-4 Dialogs..... 57

 Lesson 4-5 Business Process Flows 61

 Lesson 4-6 Actions 64

Module 5 – Server Side Code 69

 Objectives 69

 Lesson 5-1 Plugins..... 69

 Lesson 5-2 Event Framework..... 70

 Lesson 5-3 Plugin Isolation, Trusts and Statistics..... 72

 Lesson 5-4 Developing Plugins..... 73

 Lesson 5-5 Deploying Plugins and Impersonation 76

 Lesson 5-6 Debugging Plugins..... 79

 Lesson 5-7 Custom Workflow Activities..... 82

 Lesson 5-8 Declarative Workflows..... 86

 Lesson 5-9 Outlook and Plugins 87

Lesson 5-10 Azure 89

Lesson 5-11 Plugins vs Custom Workflow Activities..... 93

Lesson 5-12 Developer Toolkit..... 95

Module 6– Client Side Script..... 96

 Objectives 96

 Lesson 6-1 Use of JavaScript 96

 Lesson 6-2 Xrm.Page Model..... 97

 Lesson 6-3 Web Resources 105

 Lesson 6-4 Forms and Client-Side Events 111

 Lesson 6-5 URL Addressable Forms 114

 Lesson 6-6 Best Practice 117

 Lesson 6-7 Debugging JavaScript 120

Module 7– Client Side Code..... 123

 Objectives 123

 Lesson 7-1 Data access using JavaScript 123

 Lesson 7-2 Modern app SOAP endpoint..... 125

 Lesson 7-3 ODATA..... 126

 Lesson 7-4 JSON 129

 Lesson 7-5 JQUERY..... 130

Module 8 – User Interface 132

 Objectives 132

 Lesson 8-1 User Interface 132

 Lesson 8-2 SiteMap..... 132

 Lesson 8-3 Command Bar 135

Module 0 – Course Content and Plan

Objectives

The key objective of this course is to enable you to understand different ways of developing against Dynamics CRM with C# and JavaScript.

What this course covers

- Extensibility Model and Framework
- Platform Operations
- Querying Data
- Processes
- Server Side Code
- Client Side Scripts
- Client Side Code
- User Interface

What this course does not cover

- C# Programming
- SQL Reporting Services reports
- Service functionality

Course Plan

This course takes 2.5 days to complete and helps prepare for the Microsoft Dynamics CRM 2013 Extending certification exam.

Course Modules

1. Extensibility Model and Framework
2. Platform Operations
3. Querying Data
4. Processes
5. Server Side Code
6. Client Side Scripts
7. Client Side Code
8. User Interface

Resources

Virtual Machines

The course employs a Virtual Machines running on Hyper-V:

- Firebrand-Extending. This is a Windows Server 2013 computer with Dynamics CRM 2013 and Visual Studio already installed

You start virtual machines by starting Hyper-V Manager and right-clicking on the virtual machine and selecting Start.

Case Study

There are separate PDFs for the Case Studies. These case studies build on the labs from the customisation course. There is one case study for demonstrations (Event Management) and another for the labs (New Product Development: Prototypes, Ideas, Feedback).

SDK

The Microsoft Dynamics CRM Software Developer kit is available from microsoft.com/downloads and from the MSDN CRM Developer Centre.

Exam

The Extending Microsoft Dynamics CRM 2013 exam is a Microsoft certification exam and is taken online in one of the testing rooms off the Firebrand reception and refreshments area

Id

You will need two forms of id; one with a photo id e.g., a passport or driving license and the other with your signature e.g., a debit/credit card.

Exam Format

The exam has 48 multiple choice questions and you are allowed 140 minutes.

There is three hours in total to allow for you to complete i) a test exam on the planets of the solar system to get you used to the examination format and ii) two surveys at the end of the exam.

Exam Preparation

There are currently no practice tests available to prepare for the exam.

Skills Measured

This certification exam measures your ability to extend Microsoft Dynamics CRM 2013 including planning the deployment of Microsoft Dynamics CRM, performing common and advanced platform operations, implementing business processes, creating plug-ins, implementing application events, and implementing web resources.

Planning the Deployment of Microsoft Dynamics CRM (15-20 percent)

- Describe the CRM application framework
 - This topic may include: describe the Dynamics CRM extensibility platform; describe Portable Business Logic functionality; explain the Application, Platform, and Database layers; describe the differences between the CRM application framework on-premises and online
- Identify the considerations for deploying Dynamics CRM.
 - This topic may include: describe the deployment model; identify support for Microsoft Outlook clients; identify the extensibility points; identify types of reporting; identify the web resources
- Plan for user interface customization.
 - This topic may include: describe the Web Application Navigation and Nav bars; customize the Site Map and command bar; describe the entity command bar templates; describe how customizations impact Microsoft Outlook; edit the custom actions and command definitions; describe Nav bar enable and display rules; describe how to use localized labels with command bars; describe URL-addressable forms and views

Performing Common Platform Operations (15-20 percent)

- Explain the Discovery Service.
 - This topic may include: describe Discovery Service authentication and authorization; explain the differences between the Discovery Service Web Service on-premises and online
- Explain the Organization Service.
 - This topic may include: describe Organization Service authentication and authorization; describe entity information; explain the RetrieveAttributeRequest and RetrieveAttributeResponse methods
- Describe how to use data types, methods, and classes.
 - This topic may include: explain the different data types; describe how to use the Create, Retrieve, Update, Delete, and RetrieveMultiple methods; describe how to use early- and late-bound classes
- Describe Windows Communication Foundation (WCF) web services.
 - This topic may include: describe integration of Dynamics CRM and WCF; describe how to handle WCF faults; use and specify Open Data Protocol (OData) elements when querying Dynamics CRM data

Performing Advanced Platform Operations (10-15 percent)

- Identify different ways to query data.
 - This topic may include: explain QueryExpression, QueryByAttribute, LINQ queries, and filtered views; explain how to save queries; explain FetchXML and custom SQL Server Reporting Services (SSRS) reporting in relation to Dynamics CRM; describe the integration of Windows Azure with Dynamics CRM
- Identify how to use requests and responses.

- This topic may include: identify how to use the Execute() method; use entity-specific and non-entity specific requests; use simple generic request messages; pass optional parameters in messages
- Explain the Metadata web service.
 - This topic may include: describe the metadata layer; explain the Read and Write actions possible with metadata; use metadata from custom applications

Implementing Business Processes (10-15 percent)

- Describe workflows.
 - This topic may include: describe workflow process architecture; describe custom workflow activities and custom XAML workflows (declarative workflows); describe workflow rules and binding rules to events
- Describe dialogs.
 - This topic may include: explain input arguments in dialogs; describe dialog rules, events, pages, prompts, responses, actions, and conditions
- Create and manage custom workflows.
 - This topic may include: set up custom workflow activity assemblies; create, configure, and debug custom workflow activities; create and modify workflows in Windows Workflow Foundation (WF); describe business process flows

Creating Plug-ins (15-20 percent)

- Describe plug-ins.
 - This topic may include: explain plug-ins and when to use them; describe cascading events; explain the information available in plug-ins; describe plug-in isolation, trusts, and statistics; describe impersonation in plug-ins; describe how to register and deploy plug-ins; describe how to debug plug-ins
- Describe the event framework.
 - This topic may include: describe the key features of the event framework; describe the event execution pipeline; describe how to use Entity classes in plug-ins

Implementing Application Events (15-20 percent)

- Describe application events.
 - This topic may include: describe the use of JavaScript libraries to customize Dynamics CRM; describe form and field events including OnChange, OnLoad, OnSave, TabStateChange, and OnReadyStateComplete; implement IFRAMES in entity forms
- Implement client-side code.
 - This topic may include: describe using JavaScript for client-side events; debug client-side code; request external data; pass parameters; access Dynamics CRM 2013 web services
- Implement form types.

- This topic may include: describe the different form types; implement the Xrm.Page object; implement Xrm.Utility; explain how to test form types; describe the form event handler execution context
- Describe global variables and functions.
 - This topic may include: explain how to set dependencies and pass parameters; define the allowed query string parameters; add and handle form parameters; explain the getQueryStringParameters method

Implementing Web Resources (10-15 percent)

- Explain how to use the different types of web resources.
 - This topic may include: define webpage (HTML), style sheet (CSS), script (JavaScript), data (XML), images (PNG, JPG, GIF, ICO), Silverlight (XAP), and style sheet (XSL) web resources; create web resources; identify the limitations of each web resource; implement the passing of parameters between web resources; reference web resources; implement JavaScript libraries for code reuse across multiple applications
- Explain how to use REST, OData, and JSON in Dynamics CRM.
 - This topic may include: explain Representational State Transfer (REST); use REST in AJAX and JavaScript; explain JavaScript Object Notation (JSON); explain how to use the XMLHttpRequest object; use OData and JavaScript to create and update an account record
- Explain how to use JQuery in Dynamics CRM.
 - This topic may include: explain JQuery and how to use the JQuery object; use JQuery with a web resource; use JQuery to interact with the CRM form

Feedback

You will need to complete two sets of feedback at the end of the course. One is for Firebrand and is available on your PC; <http://www.firebrandtraining.co.uk/feedback>. The other is for Microsoft and your instructor will give you the link to the KnowledgeAdvisors MetricsThatMatter website that Microsoft uses for feedback.

Module 1 – Extensibility Model and Framework

Objectives

The key objective of this module is to describe the Dynamics CRM application framework and the different ways CRM can be extended.

In this module we will cover:

- CRM Functionality
- XRM Application Framework
- Deployment Options for Dynamics CRM
- What's New in Dynamics CRM 2013
- Resources available

Lesson 1-1 CRM Functionality

Microsoft Dynamics CRM Functionality

Microsoft Dynamics CRM is designed to support the sales, marketing and service functions of an organisation.

Out of the box, Dynamics CRM provides significant functionality for many organisations and also provides capability to customise and extend the functionality to meet specific business requirements.

Third parties also provide pre-built customisations via the Dynamics CRM Marketplace.

Sales

The sales functionality within Dynamics CRM covers the generation of leads for prospecting and qualifying, managing opportunities and keeping track of stages of deal closure, managing and tracking communications between salespeople and the customers, and maintaining a database of product information

- Leads
- Opportunities
- Communication tracking
- Products and Pricing
- Sales processes

Sales Process

The sales process starts with the generation of a lead; that then follows a qualification process to convert it to an opportunity. A quote can be generated for the customer, which then can become an

order and, from this order, invoices can be generated. This entire sales process is modelled within Dynamics CRM.

Marketing

The marketing functionality within Dynamics CRM allows you to do campaign planning, campaign budgeting and creating target marketing lists of contacts, accounts and leads that you want to market your services to. You can generate campaigns in order to provide you the ability to send out email blasts or mail to a particular marketing lists, and then tracking and reporting the efficacy of those campaigns through reports and charts.

- Market lists
- Campaigns
- Tracking responses
- Reporting

Service

The service functionality within Dynamics CRM allows you to record cases for issues or ticket tracking for customers, managing services that you provide or contracts that you have with a customer.

You can manage the services and resources that you have at hand through appointment scheduling allowing you to take your available resources, schedule them out optimally, and find when they are available to be deployed to render the services that you provide your customers.

A knowledge base of existing information and intelligence that you can use to help you more effectively solve cases and issues that may arise with a customer.

- Case recording and resolution
- Contract Management
- Service and Resource Management
- Service Scheduling
- Knowledge Base

Lesson 1-2 XRM Application Framework

Microsoft Dynamics CRM 2013 includes a declarative development of relational business applications that have flexible data models and dynamic services. ISVs building Extended CRM applications on Microsoft Dynamics CRM 2013 use the .NET Framework and other common Microsoft platform technologies such as

- Web Server (IIS)
- Windows Workflow Foundation (WF)
- Windows Communication Foundation (WCF)
- SQL Server

XRM Application Framework

The XRM application framework is the common foundation used by both the core CRM applications built by Microsoft and the Extended CRM applications built by ISVs and partners. For example, an insurance agency or finance firm could use Microsoft Dynamics CRM 2013 in a traditional sense but could also benefit greatly with an extended CRM application to manage policies, documents and interoperate with other industry standard applications. These applications all take advantage of the following high level features of the xRM application framework:

- Models that consist of multiple domains that automatically include data, presentation, workflow and security to name a few.
- Business application services that include extensible client experiences, multi-tenancy, robust web services that all adapt to the current application's published model.
- Enterprise scalability and a platform that provides a proven commitment to backward compatibility and early adoption of key Microsoft technologies (for example .NET).
- Running in the cloud with CRM Online and interoperability with other Microsoft Cloud products such as Windows Azure.

Extended CRM applications can be fast to build with point-and-click customizations and drag-and-drop user interface (UI) designs.

Sometimes more than point and click customizations are needed to resolve issues. In this instance developers must extend the solution to include custom code which is the focus of this course. Developers work mostly with tools such as Microsoft Visual Studio to interact with and extend services. End-users interact with the application by the use of a familiar browser based interface or through the CRM client for Microsoft Outlook. Generally, applications built that use Microsoft Dynamics CRM 2013 work the way that they are expected to work.

Extended CRM applications that use dynamic service capabilities can adapt to changing business needs. For example, when a new attribute is added to the model it is immediately available in the UI, from the web services and also available for reporting and workflows. Typically, in a traditional application this would have required some effort to accomplish.

Customizations and full extended applications are packaged as a (Managed) Solution. ISVs can build a common solution to a problem which can be customized to fit the individual needs of their end customers in a more cost-effective way. ISVs can then publish and sell their solution in the Microsoft Dynamics Marketplace (<http://www.microsoft.com/dynamics/marketplace>).

Declarative Design Model

Microsoft Dynamics CRM 2013 contains a declarative model. This means that when the application is designed to meet specific business processes, the customizer and developer do not have to spend time creating events which perform basic create, read, update, and delete (CRUD) actions. These actions include designing a security model for the form, designing a form structure, or developing code to perform navigational and interaction events.

All these actions are abstracted from the events and the complexity of this is performed in the background by the XRM framework. This provides developers and customizers more time focusing on the specific business processes and scenarios.

The model contains the building blocks for an application; and by itself it is only a collection of related objects. However, the interaction between those objects is used to implement more extensible logic such as the quote-to-order-to-invoice processing and pricing logic.

Four-layer architecture

Microsoft Dynamics CRM has a conceptual 4-layer architecture.

Presentation

The Presentation layer is your browser, or Outlook, or mobile client.

Application

The Application layer is quite light and displays the ASPX pages you see through the browser.

The Application layer is where any JavaScript, you have added to the form, is executed. The Application layer also enforces field settings such as:

- Min/Max values
- Mandatory fields
- Field length
- Data Types validation

Platform

The Platform layer is probably the most important from the perspective of this course. The platform performs a number of key functions including:

- CRM Business Logic
- Custom Business Logic
- Security
- Workflows
- Data Imports
- Duplicate Detection

The Platform uses the Metadata extensively e.g. to convert user queries into SQL statements.

The Platform is also where the web services reside.

Data

The data layer is the SQL Server database for your organisation.

In Dynamics CRM Online and Partner Hosted deployments you will not have direct access to SQL Server.

In an on-premise deployment, access to SQL database is restricted to Filtered Views that is a) read only and b) enforces data security model.

You should never access the SQL database directly to create or amend data or tables within the database. You are allowed to create indexes but nothing else, no stored procedures or triggers.

Extensibility Platform and Features

The extensibility platform is the core of both Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online. When using Microsoft Dynamics CRM SDK, developers are building on top of this system.

When developing an application that uses Microsoft Dynamics CRM, developers use web services to communicate with the underlying XRM framework layer. Microsoft Dynamics CRM uses a metadata driven architecture to provide the flexibility to create custom entities and additional system entity attributes. This architecture is also used to make upgrades and make the transportation of solutions easier. By doing this you can make changes in the data structure without having to change code in Microsoft Dynamics CRM.

The XRM framework also controls access to data through security, controls access to the database, and raises events for workflow processes and custom business logic implementations (plug-ins). The platform layer provides for both incoming and outgoing email processing through the Microsoft Exchange Server.

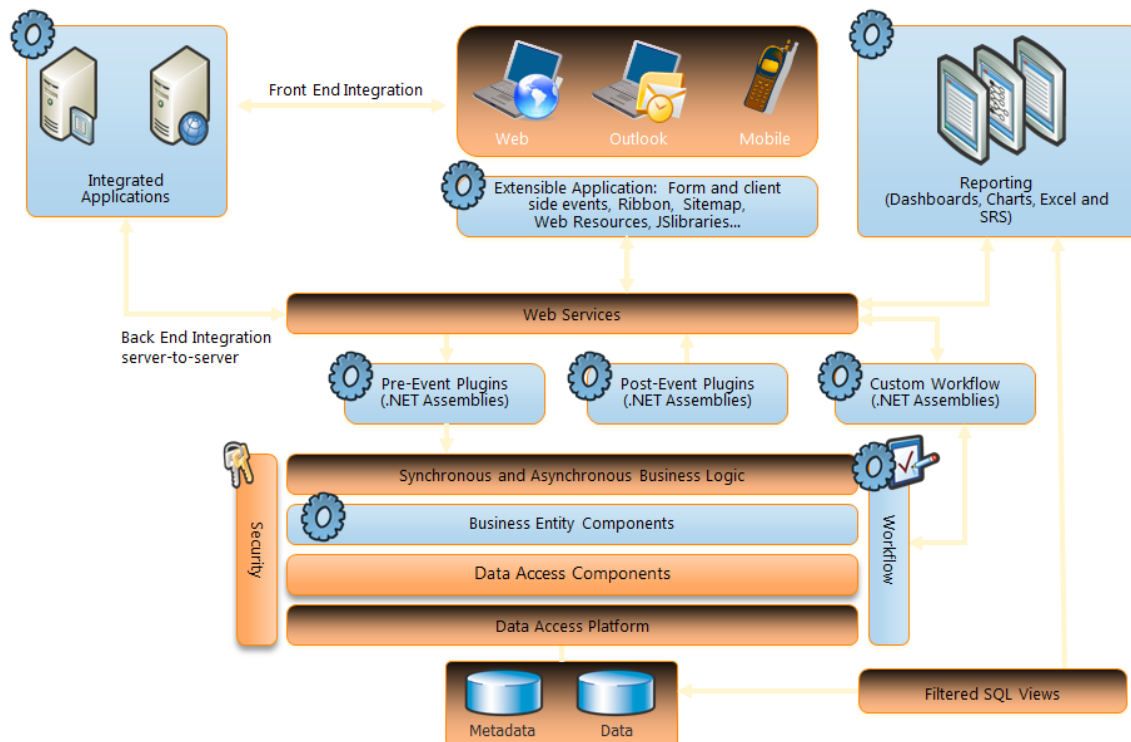


Figure 1 – Extensibility points

The above figure shows several components in the structure of a Microsoft Dynamics CRM 2013 environment. Within these components, there are several extensibility points:

- Form and Client-Side Events
- Command Bar and SiteMap
- JScript Libraries
- Plug-ins
- Web service programming
- Web Resources such as the following:
 - Web Page (HTML)
 - Data (XML)
 - Style Sheet (XSL and CSS)
 - JScript
 - Silverlight
- Microsoft SQL Server Reporting Services
- Processes

Lesson 1-3 Deployment Options

Deployment Options

There are various ways in which you can deploy Dynamics CRM 2013; on-premise, online, or a partner-hosted model.

The functionality is almost identical between the deployment options.

Clients

Microsoft Dynamics CRM is essentially a web-based application. There are various different supported browsers; Internet Explorer, Firefox, Chrome and Safari. You can just open your browser and type in the URL of the CRM server.

Another way to access it is through Outlook. The CRM Client for Outlook uses the standard Outlook interface, and enables you to access all of your records and CRM data through Outlook. It also provides you additional ability to track and sync your emails, appointments and tasks between Outlook and CRM.

New in 2013 is the Native Mobile Client; this is available for tablets and various phones through app stores. You are able to access your CRM data through a native application on your phone or tablet.

Security

User Access

All users must be authorised and authenticated before they can access Dynamics CRM.

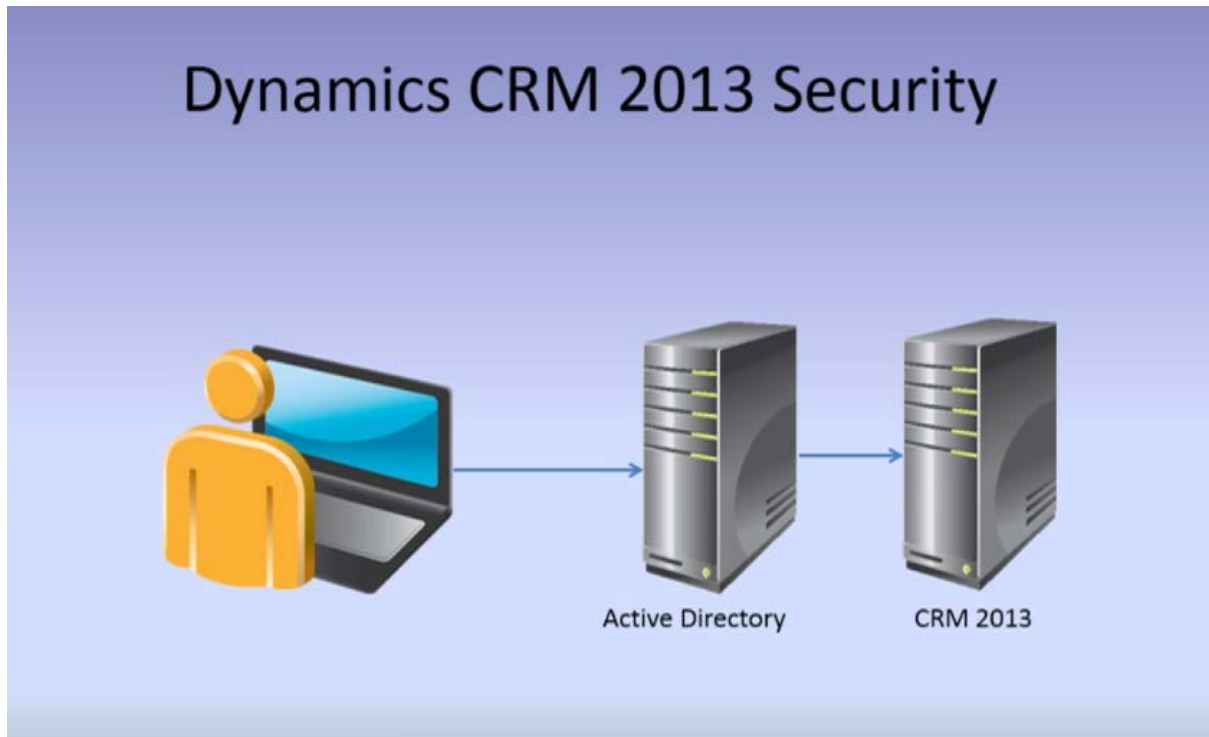


Figure 2 - Authentication

If a user with an on-premise deployment of Dynamics CRM user logs into their computer and access CRM, they are logged in with their Active Directory user name and the user is automatically logged in to CRM 2013.

For CRM Online and partner-hosted deployments, users will need to login to CRM separately using Claims Based Authentication.

Security Roles

Security roles allow different access to different parts of CRM both functionality and records. Users must be given at least one security role to access CRM.

Other Security Features

Records can be shared with other users and teams.

CRM 2013 has new functionality, access teams, to allow records to be shared with multiple users via access templates.

Users can also be provided with different forms based on their security roles.

It is also possible to restrict access to specific fields via Field Level Security.

The security model is very flexible and robust and cannot be bypassed.

Lesson 1-4 What's New in CRM 2013

Microsoft CRM 2013 new features include:

New UI and Navigation

The CRM user interface has been redesigned. The new menu bar replaces CRM 2013 Navigation pane and designed to ease user navigation, free up screen space and create a consistent look and feel across desktop and touch enabled mobile devices. The CRM 2013 ribbon has been replaced by a simpler single line action bar with a “more actions” drop down list freeing up screen space for more data, charts and information on forms. Commands are related to what you are working on and change based on data and record opened.

Quick Create form

This very basic form allows you to quickly enter a record. For example, if you're on the phone with a Lead and want to quickly capture their name and phone number in CRM without having to navigate to the lead entity. You can customise these forms to include the type of information you want.

Quick View forms

This feature allows you to bring in data from related entities. For example, if you are working on a Case record and want to view basic Account information on the Case form, you can use a Quick View form to do that.

Business Rules

This provides an intuitive way to write simple code that was otherwise reserved for someone with development / coding experience. You can do things like set field values, show/hide fields, set required/recommended fields and validate field data.

Business Process Flows

A new visual process display can be made available that guides the user through pre-defined processes. Multiple processes can be added to each record type - think sales process for inside sales vs. outside sales. A sales organization can make sure each required step in each pipeline phase is completed before moving to the next pipeline phase.

Real time / Synchronous Workflows

Processes in CRM 2013 always ran in the background requiring a user to reload the form to see the end result of the workflow. CRM 2013 now introduces processes that run immediately and update the form in real time.

Actions

A new process that allows non-programmers to define “functions” that developers can call to perform a set of actions e.g. Escalate a case

Auto Save

Once you've created a record, this new feature automatically saves information on your form triggering every 30 seconds as well as when you leave the form. This batch updates any changes made to the form without having to remember to hit the Save button. This is a system wide setting, so you either use Auto Save on all forms or you don't.

FetchXML

Fetch XML has been expanded so allow for Left Outer Joins.

Role-Based Security Teams (owner Teams) and Access Teams

With a record-based access team, CRM users can be added to a specific record and give them access. The access team is a new type of team that doesn't own records, but, has access to records. Unlike with owner teams, you can create one or more access teams to work on the same records and have team members with different levels of access rights to the record.

Native Mobile Apps

MoCA otherwise known as "Mobile Client Application". MoCA refers to the (free) mobile app that can be downloaded to your phone, tablet, or even desktop.

This new mobile app renders your existing CRM forms as mobile forms. These mobile forms are limited and are restricted to 5 tabs or 75 fields and 10 lists, whichever one comes first. IFrames are not supported.

Global search is supported in the mobile client so you can now search across multiple entities, whereas, in the full web client the 'Quick Find' only searches within a specific entity.

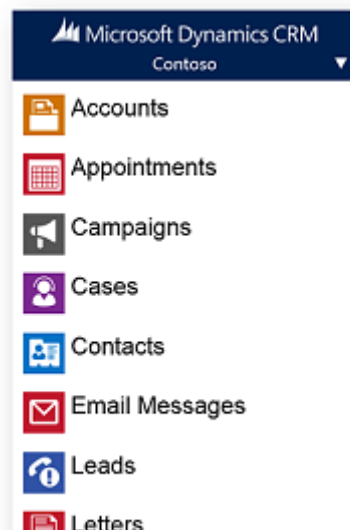
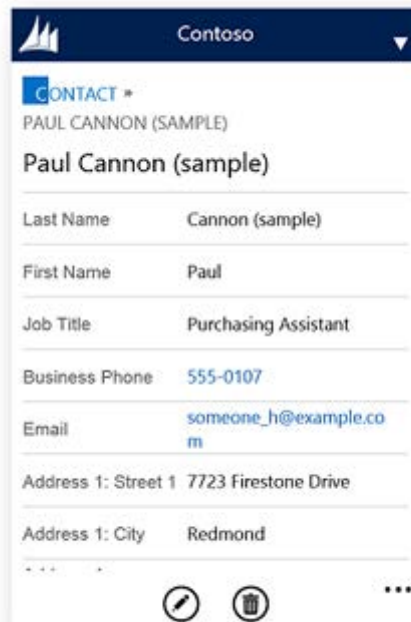
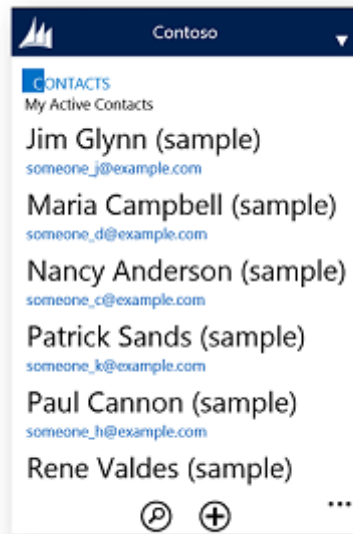


Figure 3 - Mobile Client



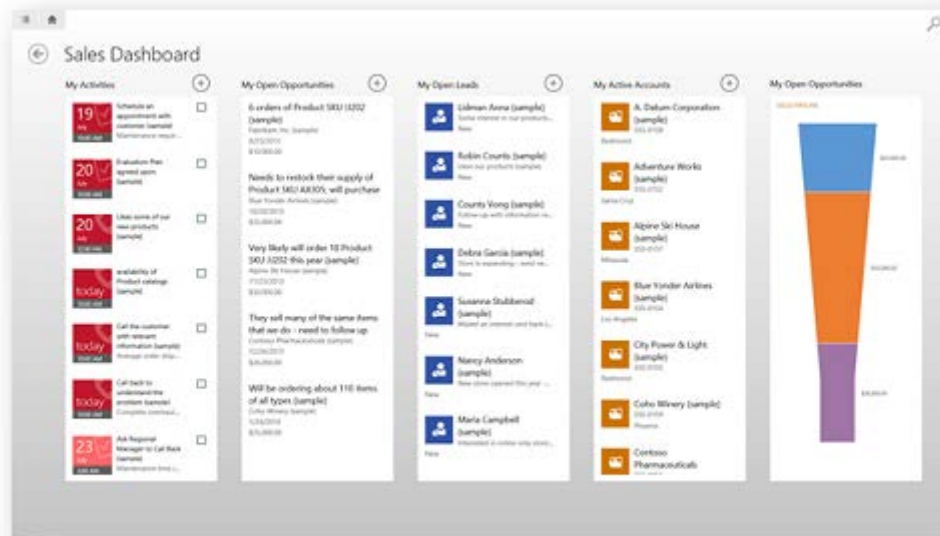


Figure 4 - Tablet Dashboard

Other new features

- Database encryption: database encryption is enabled for a set of default entity attributes that contain sensitive information, such as user names and email passwords.
- Get started pane: removed in CRM 2013 to provide further screen space for information
- Popups: there are no more pop ups in CRM 2013.
- Inline Editing: you can now click or tap a field to update info for a record right inline.
- Yammer Integration: users can participate in social conversations directly within Microsoft Dynamics, through the Yammer web and desktop applications as well as apps running on Microsoft (Windows Phone), Apple (iOS) and Google (Android) mobile devices
- Skype & Lync Integration: enabling direct dialling from any phone number field in Dynamics CRM
- Bing Maps: integrated Maps Integrated dynamically show the map for the primary record address
- Duplicate Detection: detection during record Create and Update has been removed from CRM 2013
- Server Side Synch: Server-Side Sync is used to integrate emails, activities and tasks with CRM 2013. Server-Side Sync does the same work as the Email Router and more. Currently, Server-Side Sync will only work in the following scenarios:
 - Microsoft Dynamics CRM Online → Microsoft Office 365
 - Microsoft Dynamics CRM On-Premise → Microsoft Exchange On-Premise

- Base and extension table merge: In CRM 2013 the entitynameBase and entitynameExtensionBase tables are merged into a single entitynameBase table. This can improve overall performance of CRM
- One image field per entity

New Functionality

Microsoft is releasing additional functionality as add-on license options.

- Marketing Pilot (Dynamics Marketing)
- Parature (Customer Service)
- [Social Listening](#)
- Unified Service Desktop (CCA)
- Enterprise License

Find more in the [Spring 2014 Preview Guide](#)

Lesson 1-5 Resources

Microsoft has always been supportive of developers and the following resources are available:

- CRM Developer Center msdn.microsoft.com/dynamics/crm
- Microsoft Dynamics CRM 2013 Customization Guide
- CRM Help Center www.microsoft.com/en-us/dynamics/crm-help-center
- SDK www.microsoft.com/en-us/download/details.aspx?id=40321
- Codeplex www.codeplex.com
- Forums social.microsoft.com/Forum
- Marketplace dynamics.pinpoint.microsoft.com/en-GB/home
- [Training and Adoption Kit](#)

Module 2 –Platform Operations

The key objectives of this module are to understand how to connect to the WCF web services using .NET and to execute methods to access data.

This module covers connecting to the CRM server and does not include client side access.

Objectives

The key objectives of this module are to:

- Discuss the use and benefits of the Windows Communication Foundation web services
- Describe the purpose of Discovery Service web service and how it is used
- How to use the Organization web service
- Understand the differences between early and late-bound classes
- Use the Create, Retrieve, Update and Delete Methods
- Use Request/Response method
- How to handle exceptions
- Using Meta Data

Lesson 2-1 Windows Communications Foundation

Windows Communication Foundation

Microsoft uses the Windows Communication Foundation (WCF) programming model to build service-oriented applications. Developers use it to build secure and reliable solutions that integrate across platforms and interoperate with existing investments.

The Microsoft Dynamics CRM SDK simplifies the use of WCF technology and claims-based authentication. It provides helper proxy classes making it easy to write applications that connect to and authenticate with the Microsoft Dynamics CRM web services. These helper classes can be used in applications so that developers will have to access any Microsoft Dynamics CRM deployment using the same code and without having to become an expert in claims-based security or WCF programming.

Microsoft Dynamics CRM 2013 expands upon the streamlined WCF endpoint introduced in CRM 2011 which replaced prior Web services. The WCF endpoint is a consolidated API that includes both CRM service and metadata service API in earlier versions.

Benefits

There are a number of key benefits to WCF:

- Web Services

- Streamlined API
- Improved performance
- .NET data types
- Claims based authentication

Web Services

WCF include abstracts the underlying transport technology, supports standard communications which improves interoperability, and that it is completely extendable to support any new standard or protocol.

The main WCF web service is only available for use with .NET clients. WCF also provides a SOAP web service for use when not connecting with .NET.

The WSDL, service contract definition, is static and the same or any organization.

Streamlined API with a Focused Set of Methods

Each organization has the same service contract with typed classes. The service contract ensures the entity related API will use a generic Entity class.

Standard .NET Types

The API uses standard .NET nullable types for most of the data types.

Claims Based Authentication

Authentication Methods

Microsoft Dynamics CRM 2013 supports two authentication methods:

- Basic Claims Authentication
- Active Directory Authentication

Basic Claims Authentication

The concept of a “one size fits all” security for applications accessing from different locations with multiple devices does not work well and so Microsoft created the Windows Identify Foundation (WIF) to address this challenge.

WIF is a framework for implementing claims-based authentication used by Microsoft Dynamics CRM 2013 and other Microsoft applications such as SharePoint.

Claims based authentication uses an industry standard protocol so in theory we can use any Identify Provider to authenticate users. In essence the authentication of users is handled by a third party.

Claims Based Authentication works together with WCF to provide secure user authentication and a communication channel with a Microsoft Dynamics CRM server. All Microsoft Dynamics CRM editions support claims-based authentication.

The following are some scenarios activated by moving to claims-based authentication:

- Support for any Security Assertion Markup Language (SAML) compliant provider
- Active Directory Federation Services to access Microsoft Dynamics CRM remotely using their existing identities with the need for a VPN.

Secure Token Service

Claims-based authentication requires the availability of a Secure Token Service (STS) running on a server. An STS server can be based on Active Directory Federation Services (ADFS) v2, or any platform that provides the official STS protocol.

Claims-based authentication is made up of a set of WS-* standards that describe the use of a Security Assertion Markup Language (SAML) token. The SAML token is either:

- Passive mode (when WS-Federation is used with both Microsoft Dynamics CRM 2013 and the Microsoft Dynamics CRM Online web application).
- Active mode (where WS-Trust is used with Windows Communication Foundation (WCF) clients).

Trusts

Microsoft Dynamics CRM 2013 needs to trust the third party identity provide and accept users passed from the identity provider's STS and has no need to perform further authentication.

To use Claims-based authentication you must first create a trust between Dynamics CRM and the STS.

Note: How to setup claims based authentication is covered in the MCRM course or refer to the following topics in the Microsoft Dynamics CRM 2013 Implementation Guide.

- Configure Microsoft Dynamics CRM for an Internet-facing deployment
- Claims-based authentication and Internet-Facing Deployment (IFD) requirements
- Configure relying parties for claims-based authentication

How Claims-based Authentication Works

To access a claims configured Microsoft Dynamics CRM 2013 server by using the Microsoft Dynamics CRM SDK, you must first install Windows Identity Foundation (WIF) on the development computer. The Windows Identity Foundation installs the Microsoft.IdentityModel assembly. This is referenced by the Microsoft Dynamics CRM SDK assemblies at run-time.

A request to authenticate a user is sent from Microsoft Dynamics CRM 2013 or Microsoft Dynamics CRM Online or a custom application to the STS server. The STS server determines whether the user should be authenticated, and if this is the case, issues a signed and encrypted SAML token that contains user authentication information.

How Active Directory Authentication Works

A request to authenticate a user is sent from Microsoft Dynamics CRM 2013 or a custom application to Active Directory. The WCF stack manages the authentication process for Microsoft Dynamics CRM SDK API calls from an application, whereas Internet Information Services (IIS) manages authentication for a web application.

Kerberos tickets created and passed between the computers and contain the user authentication information.

Lesson 2-2 Microsoft Dynamics CRM Web Services

Web Services provided by Dynamics CRM

There are four web services provided by Dynamics CRM:

- Discovery web service
- Organization web service
- Deployment web service
- OData (REST) web service

In CRM 2013, the OData endpoint is now available outside the application to provide authentication. This provides functionality for external apps to use the modern app endpoint for authentication.

The OData web service is used with client side coding and is covered in Module 7.

Discovery Web Service

The Discovery Service web service is used to determine the organizations with a CRM deployment that a user is a member of, and to identify the endpoint address URL needed to access the Organization Service web service for each of those organizations.

Because Microsoft Dynamics CRM 2013 works in a multi-tenant environment the discovery service is necessary so that an application can determine the endpoint address URL to access the target organization's business data.

For Microsoft Dynamics CRM 2013 installations, server and organization allocation may change as part of datacenter management and load balancing. The Discovery Service web service lets you discover which Microsoft Dynamics CRM server is serving an organization at a given time.

The Discovery Service returns a list of Organisations that the user is entitled to use and contains the names of the organisations and URL endpoints.

Assemblies required

The Discovery service requires the following assemblies:

- Microsoft.Xrm.Sdk.dll - Microsoft.Xrm.Sdk.Client namespace
- System.ServiceModel - for WCF
- System.Runtime.Serialization
- System.Security

Discovery Service URL

The URL for the discovery service on-premise is:

```
http[s]://<hostname[:port]>/XRMServices/2011/Discovery.svc
```

The URL for the discovery service using IFD is:

```
https://dev.<hostname[:port]>/XRMServices/2011/Discovery.svc
```

The URL for the discovery service for CRM Online in EMEA is:

```
https://disco.crm4.dynamics.com/XRMServices/2011/Discovery.svc
```

EndpointAccessType Enumeration

When instantiating the Discovery Service web service, you must identify the type of authentication approach. The Microsoft.Xrm.Sdk assembly has an enumeration called the EndpointAccessType. This contains three items:

- Default - Default access. Used for on-premise deployments but the actual access is determined by the endpoint URL (Value = 0)
- Internet - Internet access. Used for Microsoft Dynamics CRM Online (Value = 1)
- Intranet- Intranet access. Used for Internet-Facing (IFD) deployments (Value = 2)

Organization Web Service

In Microsoft Dynamics CRM 2013, the primary web service accessing data and metadata for your organization is the Organization Service web service. This contains the methods that are required to write code that use all the data and metadata in Microsoft Dynamics CRM.

Assemblies required

The Organisation service requires the following assemblies:

- Microsoft.Xrm.Sdk.dll - Microsoft.Xrm.Sdk.Client namespace

- System.ServiceModel - for WCF
- System.Runtime.Serialization
- System.Security

Organization Service URL

The URL for the organization service on-premise is:

```
http[s]://<hostname[:port]>/<Organisation Name>/XRMServices/2011/Organization.svc
```

The URL for the organization service using IFD is:

```
https://<Organisation Name>.<hostname[:port]>/ ]>/XRMServices/2011/Organization.svc
```

The URL for the organization service for CRM Online in EMEA is:

```
https://<Organisation Name>crm4.dynamics.com/XRMServices/2011/Organization.svc
```

Deployment Web Service

The Deployment web service is used to manage the Microsoft Dynamics CRM deployment i.e. organisations and servers. With the Deployment web service you can create organisations, disable organisations, disable servers etc.

The Microsoft Dynamics CRM Deployment Manager MMC and Microsoft.Crm Powershell cmdlets utilise this web service.

Authentication by Using the Client Proxy Classes

The easiest way to authenticate with Microsoft Dynamics CRM web services is by using the OrganizationServiceProxy and DiscoveryServiceProxy classes in the applications you are writing. The four-parameter constructor of these classes supports Microsoft Dynamics CRM 2013 deployments. This is the preferred method of authenticating with the server and is used by most of the Microsoft Dynamics CRM SDK samples. These proxy classes automatically handle claims or Active Directory authentication and also manage resource limits on the WCF channel endpoint.

Classes	Description and usage
Helper code	<p>The classes in the sample code demonstrate how to connect to the web services and authenticate the user. You can use the helper code as a basis of your own custom authentication code.</p> <p>This code is easy to use and supports all Microsoft Dynamics CRM deployment types. It also supports storing users' passwords in the Windows Credential Manager for later reuse.</p> <p>Full source code is provided so you can copy and customize it for your needs.</p>

Developer Extensions	<p>These assemblies are provided to simplify and accelerate the development of applications that interact with Microsoft Dynamics CRM. The extensions extend the functionality of the core Microsoft Dynamics CRM SDK specifically around the use of the OrganizationServiceContext class.</p> <p>For an easy method that does the hard work for you in a few lines of code, use the CrmConnection class.</p> <p>This code is easy to use and supports all Microsoft Dynamics CRM deployment types.</p>
Xrm client	<p>For advanced developers who need to customize the Windows Communication Foundation (WCF) service channel management and the authentication process, use the IServiceManagement and OrganizationServiceProxy classes in the Microsoft.Xrm.Sdk.Client namespace.</p> <p>Using these classes directly can provide better connection and authentication performance, and more flexibility. However, they require more advanced knowledge of the WCF service channel and server authentication. In addition, you must write more code to handle all Microsoft Dynamics CRM deployment types.</p>

Lesson 2-3 Early Bound and Late Bound Classes

Entity class

Developers can interact with the API using the Entity class in a late-bound fashion (working directly with the attribute collection). This kind of access is best for situations where you do not have prior knowledge of the entities your application will be working with.

Microsoft Dynamics CRM 2013 provides a utility which uses the platform metadata to generate typed classes for each entity. The generated class is used in early-bound development. These classes inherit from the generic Entity class and developers can work with the known properties. This is known as early-bound.

As a developer you can use either early or late binding in your .NET code. If you are using SOAP and a non .NET language you are restricted to late binding.

Early-Bound Entity Class

In Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online, the code generation tool (CrmSvcUtil) creates early-bound entity classes that you can use to access business data in Microsoft Dynamics CRM. These classes include one class for each entity in your organisation, including custom entities. Every time that a customizer makes customisations to a system and those fields or entities have to be used programmatically, these classes must be regenerated. The classes can be used in any project type or built into a class library. Developers can use early-bound entity classes when they create applications that use Microsoft Dynamics CRM, plug-ins and custom workflow activities.

Advantages of an Early-Bound Entity Class

The advantages to using early-bound entity classes is that all type references are checked at compile time. The compiled executable contains the code that is needed to invoke the types, properties, methods, and events.

The class created by the code generation tool includes all the entity's attributes and relationships. By using the class in their code, developers can access these attributes and be type safe. A class with attributes and relationships is created for all entities in an organization. There is no difference between the generated types for system and custom entities.

The following sample shows how to set the email address of a contact entity. For this example, assume that a query has found the record to be changed and the logical name is Contact:

```
Contact contact = New Contact();
contact.EMailAddress1 =marykay@contoso.com;
```

The following sample shows how to use custom entities and attributes:

```
new_bankaccount bankaccount = New new_bankaccount();
bankaccount.new_accountnumber ="12345";
```

Create Early-Bound Entity Classes with the Code Generation Tool (CrmSvcUtil.exe)

To use early-bound classes, custom code must know what the object model is for the deployment it is intended to work with. To obtain this information a class file will have to be created through the use of the CrmSvcUtil.

CrmSvcUtil.exe is a command-line code generation tool in Microsoft Dynamics CRM 2013 and is included in the SDK. It generates early-bound .NET Framework classes that represent the entity data model entities inside Microsoft Dynamics CRM. Running the tool is the first step in using an entity data model to develop applications for Microsoft Dynamics CRM.

The CrmSvcUtil.exe tool creates a Microsoft Visual C# or Microsoft Visual Basic .NET file that contains strongly typed classes for the entities in your organization, including custom entities and attributes. This output file contains one class for each entity. It provides early-binding and IntelliSense to aid you as you write custom code. The generated classes are partial classes that can be extended with custom business logic in separate files.

Using Early Bound

When employing early bound proxy class you must specify the following line after connecting to the Organisation service.

```
serviceProxy.ServiceConfiguration.CurrentServiceEndpoint.Behaviors.Add(new ProxyTypesBehavior());
```

or,. more simply

```
_serviceProxy.EnableProxyTypes();
```

Late-Bound Entity Class

In Microsoft Dynamics CRM 2013, a developer can use the Entity class to work with entities. This lets a developer use late binding so that he or she can work with types such as custom entities and custom attributes not available when compiling the application. When initialized, the Entity class contains the logical name of an entity and a property-bag array of the entity's attributes.

The key difference between early and late binding involves type conversion. Whereas early binding provides compile-time checking of all types so that no implicit casts occur, late binding checks types only when the object is created or an action is performed on the type. The Entity class requires types to be explicitly specified to prevent implicit casts.

When a developer works with the Entity class and uses late binding, he or she can work with the entity and logical attribute name. This contrasts with early binding where a person works with the entity and attribute schema name.

To create a new entity, first create a new instance of the Entity class and pass it an entity name. The following code example shows how to create a new account record.

```
Entity account = new Entity("account");
account["name"] = "Fourth Coffee";
_accountId = _orgService.Create(account);
```

In the preceding example, a new entity object of the type "account" is created, attributes are set, and then the `IOrganizationService.Create` method is called to create the new record.

Data Types

The programming model now uses native .NET types whenever possible.

AttributeTypeCode	CRM 2013 type
BigInt	long
Boolean	bool
CalendarRules	EntityCollection or CalendarRules[]
Customer	EntityReference
DateTime	System.DateTime
Decimal	decimal
Double	double
EntityName	string
Integer	int
Lookup	EntityReference

ManagedProperty	BooleanManagedProperty
Memo	string
Money	Money
Owner	EntityReference
PartyList	EntityCollection or ActivityParty[]
Picklist	OptionSetValue
Uniqueidentifier (Formerly PrimaryKey)	System.Guid
String	string
State	OptionSetValue or enumeration generated for the entity state
Status	OptionSetValue
Uniqueidentifier	System.Guid
Virtual	Not used in records.

Null Values

To set a field to null in most types, just set the value to null.

For strings set the field to String.Empty.

For Entity References set the value to Guid.Empty.

Lesson 2-4 Create, Retrieve, Update and Delete Methods

Create method

The Create method is used to create a new entity instance (object) in Microsoft Dynamics CRM. The parameters specify the entity that will be created and they return a GUID that contains the ID of the newly created entity.

A GUID can be specified before calling the Create method, and Microsoft Dynamics CRM will use that GUID for the new object.

Retrieve method

The Retrieve method is used to obtain an object from Microsoft Dynamics CRM and it accepts the following parameters:

- **entityName:** The logical name of the entity specified in the entityId

- `id`: The ID (GUID) of the record that you want to retrieve.
- `columnSet`: A query that specifies the set of columns, or attributes, to retrieve.

An entity is returned containing the fields specified in the `columnSet`.

ColumnSet

To specify a list of fields to retrieve, you must declare a `ColumnSet` variable and then set its `Attributes` property or create an `AllColumns` variable. Use the `ColumnSet` variable except when all attributes of the entity must be returned.

Using `AllColumns`, or `ColumnSet(true)`, returns all the data fields associated with the entity therefore it should be used sparingly, especially because it sends large quantities of unnecessary data over the network.

Many fields on an entity are references to other entities. The Microsoft Dynamics CRM platform performs the join, and then returns an object that has the `Type`, `ID`, and `Name` properties populated. That means for each reference field, Microsoft Dynamics CRM is being asked to perform a join, create an instance of a class, and populate three properties of the class. On a `Contact` object, this occurs at least five times.

When using `ColumnSet`, the `Attributes` property takes an array of strings that represent the names of the desired fields. Note: The ID column is always returned.

Update method

The `Update` method is used to update information for a specific object. When you use `Update`, you must specify the GUID for the object that will be updated.

An entity is the input parameter. Nothing is returned.

Be wary of updating a record of an entity retrieved as CRM will treat all fields in the entity as being updated even if the value has not changed. This can trigger plugins and workflows unnecessarily and cause audit logs to be written. This is particularly relevant where `AllColumns` has been used to retrieve the record. Best practice is to instantiate a new instance of the entity and only specify the fields you need to change.

Delete method

The `Delete` method is used to delete an object. When an object is deleted all appended objects are also deleted. Appended objects include any child objects for the entity being deleted, for example, addresses on an `Account`. If any objects within the parent hierarchy cannot be deleted, the delete operation rolls back and no changes are made. Objects without parental relationships are not deleted. The `Delete` method accepts a GUID for the object that will be deleted.

The `Delete` method accepts the following parameters:

- `entityName`: The logical name of the entity specified in the `entityId`
- `id`: The ID of the record that you want to retrieve.

Nothing is returned.

In CRM, it is best practice to deactivate records using `SetState` rather than delete them.

Lesson 2-5 Execute Method

Execute Method

To perform any operation other than the common Create, Retrieve, Update, Delete and RetrieveMultiple (see next module) methods, use the Execute method. The Create, Retrieve, Update, Delete and RetrieveMultiple methods provide wrappers that internally use the Execute method.

The Execute method is designed around a message-based approach to development instead of a method-based approach. Instead of having a method for every operation, there is a single method, Execute. This is passed a message (embodied as a `OrganizationRequest` class) that describes the operation to perform.

The Execute method supports a message class hierarchy to group similar operations. Execute takes a message request class as a parameter and returns a message response class. Every request message has a corresponding response message. The request describes what is needed from the platform and the response is how the platform provides it.

The difference between using a message-based approach and a method-based approach is best illustrated by the difference between the Execute method that relies on messages as parameters, and the Create, Update, Delete, Retrieve and RetrieveMultiple methods.

The SDK provides the five common methods because they are used frequently. However, each of those operations can also be performed by using Execute.

There are almost 200 operations that can be performed by using Execute paired with the appropriate Request and Response objects. In a method-based SDK, this would require almost 200 different methods with their own unique syntax.

Requests and Responses

The Execute method is a message-based method. This method takes, as its sole parameter, a Request class.

The Request class is a message sent to the server for processing. It returns a Response class which contains the results of the request.

Non-Entity Specific Requests

Many request messages are not bound to particular entities. They do not require an entity or they can be applied to any number of different entity types.

When they can be applied to different entity types, they contain a `Target` property that is not bound to a specific entity type. Otherwise, they do not have a `Target` property.

NOTE: For details about Simple Generic Requests, refer to "XRM Messages in the Organization Service" in the SDK.

Entity Specific Requests

Entity specific requests are messages that always apply to only one type of entity. These messages have the entity name in the class name, such as "GenerateQuoteFromOpportunityRequest." They do not have a Target property, instead they take entity-identifying information such as an ID.

NOTE: The terms "Non-Entity Specific Request" is not used in the Microsoft Dynamics CRM SDK. It is introduced here to clarify the difference between Requests that do not apply to a specific kind of entity and Request messages that do.

Generic Requests

There are several Execute messages that are simple generic requests. These do not have their Target property. The two most common are WhoAmI and QueryExpressionToFetchXml.

Lesson 2-6 Exceptions

Microsoft.Xrm.Sdk.OrganizationServiceFault

Many exceptions can be returned from a Microsoft Dynamics CRM web service method call. Your application design must catch and appropriately handle these exceptions. In the Microsoft Dynamics CRM SDK, all web service method calls use a communication channel to the server based on the Windows Communication Foundation (WCF) technology. In WCF terms, exceptions returned from the channel are called faults.

To access the Microsoft Dynamics CRM faults being produced by WCF, add System.ServiceModel to the header of the code. This will capture contractually-specified SOAP faults. To track those faults related specifically to Microsoft Dynamics CRM, include the following code into the catch statement:

```
catch (FaultException<Microsoft.Xrm.Sdk.OrganizationServiceFault> ex)
```

If you are accessing the discovery Web service, your code should catch DiscoveryServiceFault instead of the OrganizationServiceFault.

Other Exceptions

In addition to these exceptions and faults, code should also handle the following exceptions:

- SecurityTokenValidationException
- ExpiredSecurityTokenException
- SecurityAccessDeniedException
- MessageSecurityException
- SecurityNegotiationException

When connecting to Microsoft Dynamics CRM Online, a SecurityAccessDeniedException exception can be thrown if you use a valid Microsoft Account ID and your Microsoft Account is not associated

with any Microsoft Dynamics CRM Online organization. A `MessageSecurityException` can be thrown if your Microsoft Account ID is not valid or Microsoft Account did not authenticate.

Lesson 2-7 Metadata

Metadata

The metadata layer abstracts the underlying data storage details, such as schema and data access from the higher-level constructs of domain logic implementation and user interface and provides additional information about the data.

The organization service web service provides access to metadata of the Microsoft Dynamics CRM organization.

As a developer, you have to choose whether to take additional steps that are required to incorporate the Microsoft Dynamics CRM metadata into your application so that these changes are reflected in the application.

The alternative is to use hard-coded values and tolerate potential inconsistencies or rebuild the applications when they must reflect changes in the metadata. It is recommended to use the metadata so the applications reflect changes in Microsoft Dynamics CRM.

The metadata for an organization is stored in tables in the Microsoft SQL Server in their organization's database. These tables contain the entity, attribute, and relationship definitions for each organization. This includes the metadata for customisations.

The organization service web service contains the messages used to read or write the definitions for all the entities in a Microsoft Dynamics CRM installation with the `Microsoft.Xrm.Sdk.Metadata` name space.

Read and Write Actions

The organization service can be used to perform read and write actions.

The read actions can include the following:

- retrieve all the metadata to create a metadata cache in a client application;
- determine whether the metadata has changed since the previous retrieve;
- retrieve all the entities and determine which ones are custom entities;
- retrieve the metadata for a specific entity, system, or custom;
- retrieve the attributes for an entity;
- retrieve the metadata for a specific attribute such as possible state, names or opposite values on an attribute.

Write actions include the following:

- Create a custom entity;
- add or update an attribute for an entity, system or custom;
- create or delete a relationship between two entities;

- add or remove an option from the option set attribute;
- write an installation and install-uninstall program for the custom solution.

When accessing the metadata on the organization service, the user has to have sufficient privileges to perform the required action. Many built-in security roles do not include the required privileges on the entity, attribute and relationship securable, and this might have to be modified.

Creating fields programmatically is conceptually similar to creating them through the interface. Each data type has potentially different properties that can or have to be set during the creation process. Additionally, when creating several new fields, it is helpful to add them to a list. This can be done by creating an instance of the attribute metadata list.

When you programmatically create customizations to an environment such as modifying the form, the customizations will have to be published.

Conceptually, this is the same process as if you were doing it through the interface. You can publish an entity, multiple entities, or all entities and components.

Finally, for efficiency and performance, make sure to use the retrieve metadata changes message. This message allows you to detect changes and only retrieve what has been added or deleted, allowing for much more effective cache management. Look up more on the SDK help under Retrieve and detect changes to metadata.

Metadata Browser

The solution explorer provides access to all the entities that you can customize, but this is just a fraction of all the entities that define the metadata used for Microsoft Dynamics CRM. For most basic customization tasks the information presented within the solution explorer is going to be all you need.

Developers frequently need information about metadata and an easy way to see the metadata. If you need to have in-depth discussions with developers about metadata or if you just want to have a deeper understanding of the metadata, use the Metadata Browser solution that is included in the Microsoft Dynamics CRM SDK.

The Metadata browser is a managed solution containing only HTML web resources that you can install that will let you view all the metadata and filter entities and fields to gain a better understanding of what the metadata contains

Module 3 – Querying Data

The key objectives of this module are to provide an understanding of different methods of querying CRM data.

Objectives

The key objectives of this module are to:

- Understand the different query methods

Lesson 3-1 Queries in Dynamics CRM

Query Options

There are several ways to create queries in Microsoft Dynamics CRM 2013. Developers can use .NET Language-Integrated Query (LINQ) in early and late binding scenarios, write queries by using FetchXML, the proprietary Microsoft Dynamics CRM query language, or build a query by using Query Expression and the QueryExpression class.

Developers can also use OData in client side code. OData is covered in Module 7.

Lesson 3-2 Query Expression

Overview

The QueryExpression class can be used to build complex queries for use with the IOrganizationService.RetrieveMultiple method or the RetrieveMultipleRequest message.

Query parameters can be passed to the QueryExpression by using the ConditionExpression, ColumnSet, and FilterExpression classes.

If values from multiple entities are needed, QueryExpression can be used if there are related entities. For example, if a developer loops through a list of contacts and returns the names of all those accounts for whom that contact is a primary contact, a QueryExpression can handle this through the use of LinkedEntities.

However, if a developer wants to retrieve a list of Opportunities, Quotes, Orders and Invoices that are created on or before a particular date, QueryExpression would not handle this. Alternatively, the developer could use Filtered Views or potentially FetchXML. Depending on the specifics of the organisation and the requirements, the use of Filtered Views could be easier.

Restrictions

QueryExpression is limited to .NET code only.

QueryExpression can only retrieve records.

QueryExpression cannot perform aggregations.

QueryExpression is proprietary to Microsoft Dynamics CRM and is not used elsewhere.

Benefits

QueryExpression can retrieve values from multiple related entities.

QueryExpression can be used with both early and late bound classes.

If QueryExpression is used with early bound proxy class then you will have compile time validation.

Entity Collection

The entity collection class is a collection of business entities and is usually viewed as a result of a retrieved multiple method such as QueryExpression.

The EntityCollection class exposes the following members

Name	Description
Entities	Gets the collection of entities.
EntityName	Gets or sets the logical name of the entity.
ExtensionData	Gets or sets the structure that contains extra data.
Item	Gets or sets an item in the collection.
MinActiveRowVersion	Gets or sets the lowest active row version value.
MoreRecords	Gets or sets whether there are more records available.
PagingCookie	Gets or sets the current paging information.
TotalRecordCount	Gets the total number of records in the collection. ReturnTotalRecordCount was true when the query was executed .
TotalRecordCountLimitExceeded	Gets or sets whether the results of the query exceeds the total record count.

Lesson 3-3 QuerybyAttribute

Overview

The QueryByAttribute class can be used to build queries that test a set of attributes against a set of values. Whereas the QueryExpression enables more complex queries, the QueryByAttribute class is a simple option when it searches for records whose attributes have particular values. Use this class with the RetrieveMultiple method or the IOrganizationService.RetrieveMultipleRequest method.

QueryByAttribute has the same Benefits and Restrictions as QueryExpression.

Lesson 3-4 LINQ

Overview

In Microsoft Dynamics CRM 2013 , developers can use .NET Language-Integrated Query (LINQ) to write queries.

Developers can use the `OrganizationServiceContext` class or a derived class created by the `CrmSvcUtil` tool to write LINQ queries that access the Organisation endpoint (`Organization.svc`).

The `OrganizationServiceContext` class contains an underlying LINQ query provider that translates LINQ queries from Microsoft Visual C# or Microsoft Visual Basic .NET syntax into the query API used by Microsoft Dynamics CRM.

Restrictions

LINQ queries are built using standard language but internally uses `QueryExpression` so is limited to the features of `QueryExpression`.

The LINQ query provider supports a subset of the LINQ operators. Not all conditions that can be expressed in LINQ are supported

LINQ is limited to .NET code only.

LINQ cannot perform aggregations.

Benefits

LINQ can create, retrieve, update and delete records.

LINQ can retrieve values from multiple related entities.

LINQ can be used with both early and late bound classes.

If LINQ is used with early bound proxy class then you will have compile time validation.

Early Binding and LINQ

The `CRMSvcUtil` can generate the LINQ Service Context by providing the optional tag `/serviceContextName:<Data Context Name>` eg

```
/serviceContextName:MyDataContext
```

The LINQ Service Context that is produced by taking this approach is a gateway to the work with the LINQ provider. When the `MyDataContext` class is instantiated, an instance of the `Organization Service` will be passed:

```
var = new MyDataContext (_orgService)
```

This produces a LINQ context that code can work with. This context has public, queryable properties for each entity set.

Lesson 3-5 FetchXML

Overview

FetchXML is a proprietary query language that is used in Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online based on a schema that describes the capabilities of the language. The FetchXML language supports similar query capabilities as query expressions.

To execute a FetchXML query in Microsoft Dynamics CRM 2013, an XML query string must first be built. After creating the query string, use the `IOrganizationService.RetrieveMultiple` method to execute the query string. The privileges of the logged on user affects the set of records returned. Only records for which the logged on user has read access will be returned.

FetchXML is used as a serialized form of query internally within CRM, to save a query as a user-owned saved view in the `userquery` entity and as an organization-owned saved view in the `savedquery` entity.

A FetchXML query can be converted to a query expression with the `FetchXmlToQueryExpressionRequest` message and back again with the `QueryExpressionToFetchXmlRequest`

Not only can FetchXML be used to return data from the data tier, but its use is extended in Microsoft Dynamics CRM 2013. Developers can use FetchXML aggregations such as sum, max, min, count, and other examples to perform calculations in their FetchXML queries.

Additionally, although it is not covered in this course, Microsoft Dynamics CRM customers can create custom reports using FetchXML queries. If a report designer or developer is new to FetchXML, they can use the standard Advanced Find functionality to help construct their FetchXML query and use the Download FetchXML button.

Restrictions

FetchXML is limited to .NET code, and SQL Reporting Services as long as a) the Dynamics CRM Reporting Extensions are installed on the SQL Reporting Services server and the Dynamics CRM Reporting Authoring Extension (BIDS) is installed on the developers machine.

FetchXML can only retrieve records.

FetchXML does not have compile time validation.

FetchXML is proprietary to Microsoft Dynamics CRM and is not used elsewhere.

Benefits

FetchXML can perform aggregations.

FetchXML can perform outer joins – new in CRM 2013.

FetchXml and Aggregations

In Microsoft Dynamics CRM 2013, FetchXML includes grouping and aggregation features that allow code to calculate sum, average min, max and count.

The following aggregate functions are supported:

- sum
- avg
- min
- max
- count(*)
- count(attribute name)

Only one aggregate attribute can be specified in a query. The distinct keyword cannot be used. To create an aggregate attribute, set the keyword aggregate to true, then specify a valid entity name, attribute name, and alias (variable name).

In addition, data results can be grouped using FetchXML. The following Group By options are available:

- Group By With Linked Entity
- Group By Year
- Group By Quarter
- Group By Month
- Group By Week
- Group By Day
- Multiple Group By

Lesson 3-6 Filtered Views

Overview

Microsoft Dynamics CRM provides database views that are named filtered views, for all entities. This includes custom entities. Filtered Views are special SQL Views that make it easier to query against the Microsoft Dynamics CRM database. These views denormalise the data in the system into a single table. The views also already have security applied to them. Therefore, they only return records that the user can access through the Microsoft Dynamics CRM application.

When customizing Microsoft Dynamics CRM's schema or adding custom entities, Microsoft Dynamics CRM updates or creates the corresponding filtered views. Use filtered views to query the Microsoft Dynamics CRM data store directly using standard SQL commands. These commands are executed using the methods that are provided by the technology accessing the database. For example, when writing .NET code use a SQL Client Object, and when using SQL Server Reporting Services set the Microsoft Dynamics CRM database as a Data Source.

When to Use Filtered Views

Filtered views are best used:

- To retrieve a DataSet (containing one table, a DataTable) instead of a collection of objects. For example, to bind to a grid or use a consumer such as SQL Server Reporting Services, Excel or Microsoft® Word that expects DataSets.
- To do something that cannot be done in QueryExpression, LINQ or FetchXML. Anything done with SQL must be done with Filtered Views.
- For custom reports for On-Premise organizations.

Security and Filtered Views

When a user obtains data from filtered views, the user's Microsoft Dynamics CRM security privileges and access rights determine the data that is returned. Additionally, field level security profiles associated with a user or a team a user is part of also affect the data returned when querying a filtered view.

When calling filtered views, they must be called in the context of a Microsoft Dynamics CRM user using integrated Windows authentication. Any attempt to use SQL authentication fails.

Using filtered views from custom workflow activities and plugins

Additional steps must be taken to use filtered views within custom workflow activities and plugins. These use a network service account by default. If using filtered views to retrieve data for custom workflow activities or from a plugin, include credentials of a valid Microsoft Dynamics CRM user.

Restrictions

FilteredViews are limited to on-premise deployments only.

FilteredViews can only retrieve records.

FilteredViews cannot be used within Sandboxed plugins and custom workflow assemblies

Benefits

FilteredViews can perform aggregations.

FilteredViews can be used within SQL Reporting Services reports.

Lesson 3-7 Reporting Options

Overview

A business must be able to analyse data and visualize it in meaningful ways. Through the use of charts, dashboards, and reports, combined or individually, users and developers can build a complete business story through visualizations. Data can be displayed at a summary level and users can then drill down and take action on individual rows. Users can use the built-in tools to create the visualizations starting from scratch or from the templates that are provided by the ISV or a partner. Developers can continue from where the users started, or they can create new visualizations from

scratch. Developers can also use the additional tools that are provided by Microsoft Business Information Development Studio (BIDS) to experience a familiar Microsoft Visual Studio editing experience. The reporting capabilities include the following:

- Dashboards
- Visualizations (Charts)
- Microsoft Office Excel
- SQL Reporting Services Reports

Dashboards

Dashboards are a way to view a lot of data from an application in a single location. They help the user save time from having to visit several different locations to view key metrics. Dashboards can contain data from several entities, and they show saved charts, views, IFrames, and web resources all in one screen.

There are two types of dashboards:

- System dashboards- viewed by all users.
- User-specific dashboards- tailored by a user to his or her specifications.

Dashboards are based on the Form Storage Model (FormXml) and are included as part of a Solution package. This means that ISVs can include one or more dashboards as part of their products.

Custom dashboards can contain and display data that uses the standard xRM Framework visualization charts, grid views of data records, IFrames, and web resources such as Silverlight and HTML. Each dashboard can contain displays from several sources and still be shown together in a single user view. Additionally, the dashboard items can bring their native functionality with them; charts still have drill-down ability and grids can still be sorted or searched.

Visualizations (Charts)

Charts and dashboards provide users a higher level perspective of their business data. One or more charts can be defined and associated with an entity. They appear alongside a grid of entity data, and then the users receive an in-context visual representation of the grid data. Microsoft Dynamics CRM 2013 provides a built-in Chart designer used to build common charts in the grid area.

Charts support multiple levels of drill down into the different segments of data. When the user drills down the view shown in the grid is updated and he or she can select different chart types that are appropriate for the lower level of data.

Users can create new chart visualizations from the View Ribbon and then start the Chart Wizard to customize and select the type of chart to display. Charts can be imported, exported and they can also be shared with other users as a whole when made as part of a solution.

Developers can further customize the exported chart to perform formatting and other changes more advanced than those supported in the wizard. This includes using the full set of chart styles supported by the ASP.Net charting controls library.

Microsoft Office Excel

Another reporting option in Microsoft Dynamics CRM is Microsoft Office Excel. Users can export to static or dynamic worksheets and pivot tables.

SQL Reporting Services Reports

Microsoft Dynamics CRM provides many out-of-box reports for viewing business data. Custom reports can be created by using one of these reports as a template and modifying it within Business Intelligence Development Studio or they can be created from scratch using the built-in Report Wizard or the Business Intelligence Development Studio.

There are two types of reports in Microsoft Dynamics CRM:

SQL-based

These reports use SQL queries to securely retrieve data for reports from filtered views defined by the system.

Fetch-based

Fetch-based reports use FetchXML queries to retrieve data for reports. Custom fetch-based reports can be deployed to Microsoft Dynamics CRM Online and to Microsoft Dynamics CRM 2013 on-premise or hosted. All reports that are created using the Report Wizard in Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online are Fetch-based reports.

Module 4 – Processes

The key objectives of this module are to provide an understanding of the different process in Dynamics CRM.

Objectives

The key objectives of this module are to:

- Understand process types
- Workflows
- Dialogs
- Business Process Flows
- Actions

Lesson 4-1 Dynamics CRM Processes

Business Processes

Defining and enforcing consistent business processes is one of the main reasons people use Microsoft Dynamics CRM. Processes are a group of features that you can use to define and enforce consistent processes for your organisation. These consistent processes help make sure people using the system can focus on their work and not on remembering to perform a set of manual steps. Processes can be simple or complex and can change over time.

Processes are designed to be used by people who aren't developers. The rules that are defined in processes contain similar logic that a developer may apply using code, but you don't need to call in a developer each time you want to change the rules. However, you do need to have a clear understanding of the logic in the rules and understand the capabilities of each type of process. You still need to apply processes carefully and test the results to make sure you get what you want

Business processes are therefore an important part of any enterprise software application. There are three types of business processes:

- **Automated:** Relies only on communication among applications based on a set of rules.
- **Interactive:** Relies on individuals to initiate and run the process and make the appropriate decisions during the running of the process.
- **Combination:** A combination of both the automated and interactive processes.

With Microsoft Dynamics CRM 2013, you can create and manage your automated and interactive processes. The entity used to implement a process still includes workflows and dialogs and now it includes business process flows and actions.

Microsoft Dynamics CRM offers developers a way to extend and customize the standard behaviour of processes by developing custom components. The Microsoft Dynamics CRM process is based on

the Windows Workflow Foundation (WWF) programming model. The WWF provides a runtime engine, framework, base library of activities, and default implementations of the runtime services. The WWF runtime engine manages process execution, and supports processes that can remain active for extended periods of time. It also preserves the state of process execution during computer shutdown and restart.

Processes in Dynamics CRM

There are four process types in Dynamics CRM

- Workflows
- Dialogs
- Business Processes Flows (Guided Business Processes) – new in CRM 2013
- Actions – new in CRM 2013

The following table provides an overview of when to use each category of process.

Process category	Description
Workflow	<p>Use workflows to automate business processes behind the scenes. Workflows are typically initiated by system events so the user doesn't need to be aware that they are running, but they can also be configured for people to manually initiate them.</p> <p>Workflows can operate in the background (asynchronously) or in real-time (synchronously). These are referred to separately as background workflows or real-time workflows.</p>
Dialogs	<p>Use dialogs to create a user interface that will guide people through a script for customer interaction or a wizard to perform complex actions consistently.</p>
Business Process Flows	<p>Use business process flows to define the steps in which people should enter data to achieve an outcome. Business process flows add a control to the top of a form that show people what data they need to enter to move forward to the next stage and ultimately to completion of a business process. A business process flow can span multiple entities.</p>
Actions	<p>Use actions to expand the vocabulary available for developers to express business processes. With core verbs like Create, Update, Delete, and Assign provided by the system, an Action uses those core verbs to create more expressive verbs like Approve, Escalate, Route, or Schedule. If the definition of a business process changes, someone who isn't a developer can edit the Action so the code doesn't need to be changed.</p>

Note: Business process flows are different from other types of processes. All processes use the same underlying technology and information about them is stored in the Process entity. Business process flows have a different configuration experience and behaviour compared to other types of processes

Processes and Entities and Events

Processes are bound to a specific entity set when the process is created.

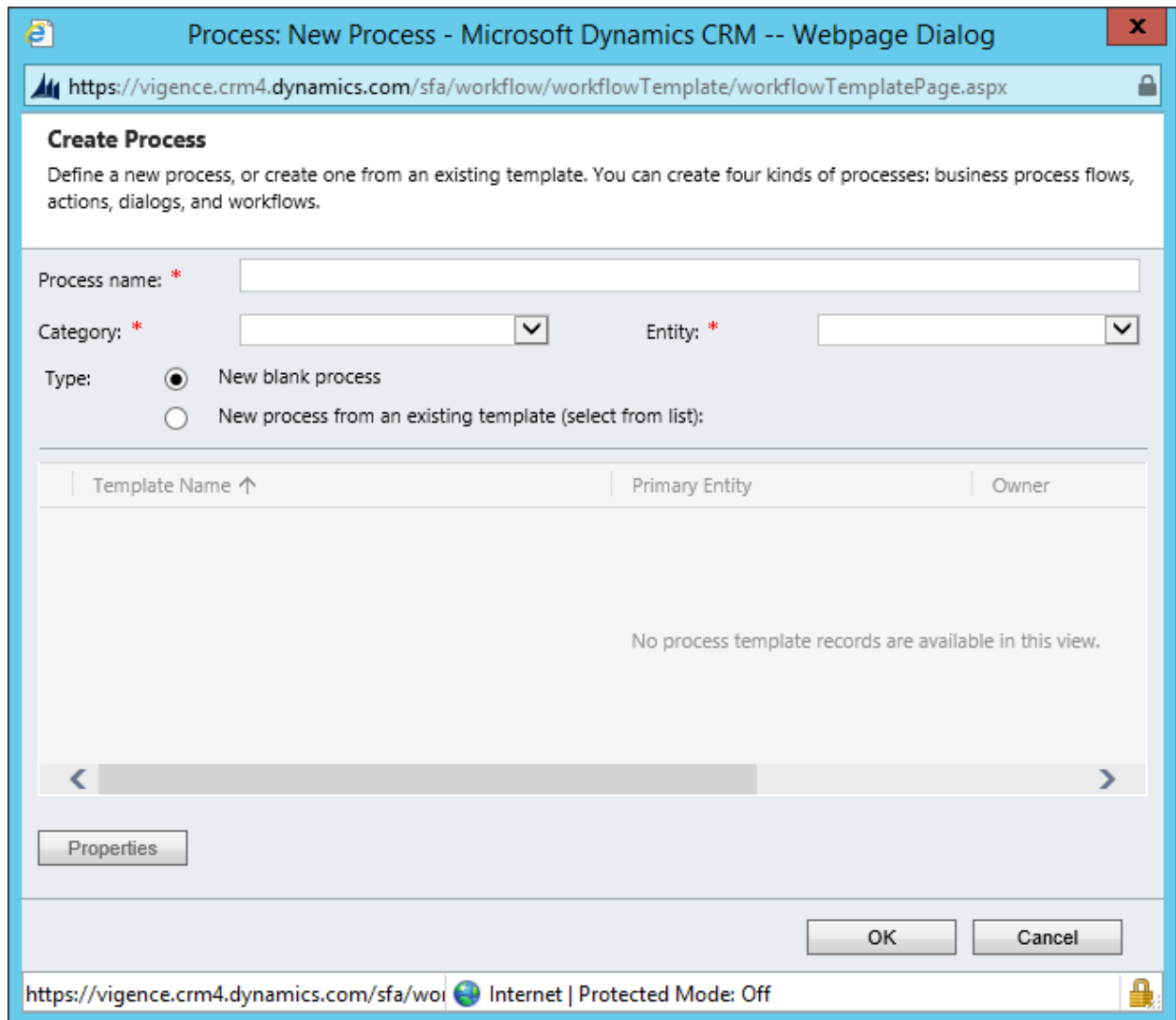


Figure 5 - New process form

When creating a new process you specify the name, process type (category) and entity. Once saved you cannot change the entity,

Process are initiated either automatically or manually. The following table describes the events that can trigger a process.

Event	Process type	Description
Create	Workflow, Business Process Flow	Creation of a record
Update	Workflow	Update of one a selection of fields for a record
Delete	Workflow	Deletion of a record
Change Status	Workflow	Change of statecode of a record
Assign	Workflow	Change of ownership of a record
On Demand	Workflow, Dialog	User initiates the process through the Run Workflow or Run Dialog option in the commend bar
Child	Workflow, Dialog	Process is called from another process
Custom	Action	Execution of the action from code

Lesson 4-2 Dynamics CRM Process Steps

Process Steps

Processes can check conditions, apply branching logic, and perform actions. They perform these actions in a series of steps.

Business process flows contain stages and control advancement to stages, but they don't provide any of the other capabilities.

The following table describes the available steps in workflow, dialog, and action processes.

Step	Process type	Description
Stage	Workflow, Dialog, Action	Stages make the workflow logic easier to read, and explain the workflow logic. However, stages don't affect the logic or behaviour of workflows. If a process has stages, all the steps

Step	Process type	Description
		in the process must be contained with a stage.
Check Condition	Workflow, Dialog, Action	<p>A logical "if-<condition> then" statement.</p> <p>You can check values for the record that the workflow is running on, any of the records linked to that record in an N:1 relationship, or any records created by earlier steps. Based on these values you can define additional steps when the condition is true.</p>
Conditional Branch	Workflow, Dialog, Action	<p>A logical "else-if-then" statement, the editor uses the text "Otherwise, if <condition> then:"</p> <p>Select a check condition you have previously defined and you can add a conditional branch to define additional steps when the check condition returns false.</p>
Default Action	Workflow, Dialog, Action	<p>A logical "else" statement. the editor uses the text "Otherwise:"</p> <p>Select a check condition, conditional branch, wait condition, or parallel wait branch that you have previously defined and you can use a default action to define steps for all cases that don't match the criteria defined in condition or branch elements.</p>
Wait Condition	Background Workflow Only	Enables a background workflow

Step	Process type	Description
		to pause itself until the criteria defined by the condition have been met. The workflow starts again automatically when the criteria in the wait condition have been met.
Parallel Wait Branch	Background Workflow Only	Defines an alternative wait condition for a background workflow with a corresponding set of additional steps that are performed only when the initial criterion is met. You can use parallel wait branches to create time limits in your workflow logic. They help prevent the workflow from waiting indefinitely until the criteria defined in a wait condition have been met.
Assign Value	Dialog, Action	Sets a value to a variable or output parameter in the process.
Create Record	Workflow, Dialog, Action	Creates a new record for an entity and assigns values to attributes.
Update Record	Workflow, Dialog, Action	You can update the record that the workflow is running on, any of the records linked to that record in an N:1 relationship, or any records created by earlier steps.
Assign Record	Workflow, Dialog, Action	You can assign the record that the workflow is running on, any of the records linked to that record with an N:1 relationship, or any records created by earlier steps.

Step	Process type	Description
Send Email	Workflow, Dialog, Action	Sends an email. You can choose to create a new email message or use an email template configured for the entity of the record that the workflow is running on or any entities that have an N:1 relationship with the entity, or the entity for any records created by earlier steps.
Start Child Workflow	Workflow, Dialog, Action	Starts a workflow process that has been configured as a child workflow.
Change Status	Workflow, Dialog, Action	Changes the status of the record that the process is running on, any of the records linked to that record with an N:1 relationship, or any records created by earlier steps.
Stop Workflow/Stop Dialog	Workflow, Dialog, Action	Stops the current workflow, dialog, or action. You can set a status of either Succeeded or Canceled and specify a status message.
Page	Dialog	A container for prompt and response steps in a dialog.
Prompt and Response	Dialog	Displays a prompt in a dialog page and may provide a field to capture data from a response.
Query CRM Data	Dialog	Defines a query that returns data to provide options for a response in a prompt and response step of a dialog.
Link Child Dialog	Dialog	Starts a dialog process that has been configured as a child dialog.

Step	Process type	Description
Custom Step	Workflow, Dialog, Action	<p>Provides extensions to the logical elements available by default in CRM. Steps can include conditions, actions, other steps, or a combination of these elements. Developers can create custom workflow steps. By default, there are no custom steps available in CRM.</p> <p>Custom workflow activities are covered in the next module.</p>

Lesson 4-3 Workflows

Workflows

Workflows automate business processes without a user interface. People usually use workflow processes to initiate automation that doesn't require any user interaction.

Each workflow process is associated with a single entity. When configuring workflows you have four major areas to consider:

- When to start them?
- Should they run as a real-time workflow or a background workflow?
- What actions should they perform?
- Under what conditions actions should be performed?

Use this process to model and automate real world business processes. These processes can be configured to run in the background or in real time. Workflow processes can start automatically based on specified conditions or can be started manually by a user.

Workflows consist of conditions (If then Else and Wait) and actions (Create record, update record, send email, assign record).

Workflows are triggered by (some) changes (creation of a record, assigning of a record to a user, change of status, updating of a field) to data. Works can be run manually (on-demand). Workflows

Workflows run on the CRM server

Workflow Properties

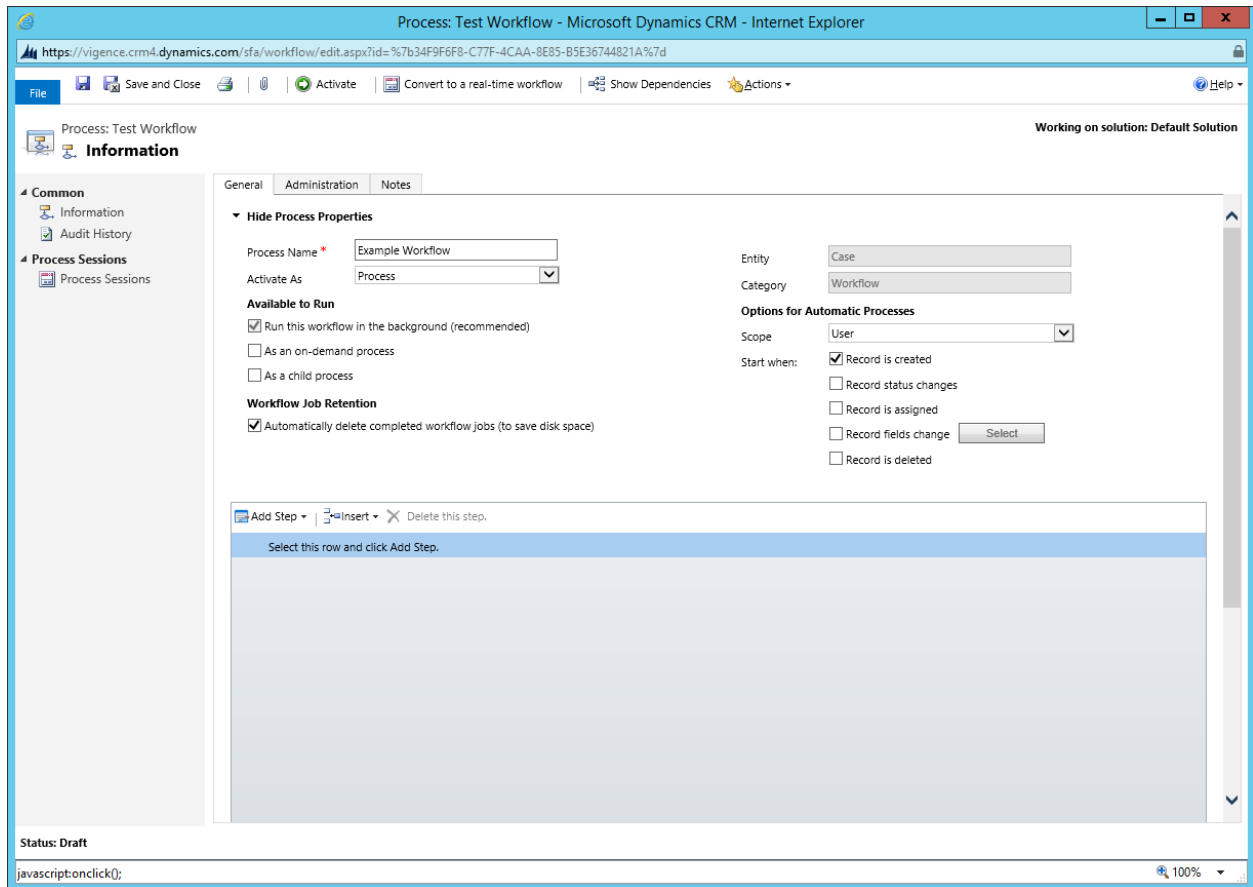


Figure 6 - Workflow Designer

Property	Description
Activate As	<p>You can choose Process template to create an advanced starting point for other workflow. If you choose this option, after you activate the workflow it will not be applied but instead it will be available to select in the Create Process dialog if you select Type: New process from an existing template (select from list)</p> <p>Process templates are convenient when you have a number of similar workflow processes and want to define them without duplicating the same logic</p>
Run this Workflow in the background (recommended)	<p>This check box reflects the option you selected when you created the workflow. This option is disabled, but you can change it from the Actions menu by choosing either Convert to a real-time workflow or Convert to a background workflow.</p>
As an on-demand	<p>Choose this option if you want to allow users to run this workflow from the</p>

Property	Description
process	Run Workflow command
As a child process	Choose this option if you want to allow the workflow to be available to be started from another workflow or dialog.
Scope	<p>For user-owned entities, options are Organization, Parent: Child Business Units, Business Unit, or User. For Organization-owned entities the only option is Organization.</p> <p>If scope is Organization, then the workflow logic can be applied to any record in the organization. Otherwise, the workflow can only be applied to a subset of records that fall within the scope.</p> <p>Note: The default scope value is User. Make sure you verify that the scope value is appropriate before you activate the workflow.</p>
Start When	<p>Use the options in this section to specify when a workflow should start automatically. You can configure a real-time workflow to be run before certain events. This is a very powerful capability because the workflow can stop the action before it occurs.</p> <p>The options are:</p> <ul style="list-style-type: none"> • Record is created • Record status changes • Record is assigned • Record fields change • Record is deleted
Execute As	This option is only available if you unselected the Run this workflow in the background (recommended) option when you created the workflow or if you later converted a background workflow to be a real-time workflow

Real Time Workflows

In CRM 2011 workflow processes always ran in the background requiring a user to reload the form to see the end result of the workflow. CRM 2013 has introduced the option for processes to run immediately and update the form in real time.

You can configure real-time workflows but you should use them with care. Background workflows are generally recommended because they allow the system to apply them as resources on the server are available. This helps smooth out the work the server has to do and help maintain the best performance for everyone using the system. The drawback is that actions defined by background workflows are not immediate. You can't predict when they will be applied, but generally it will take a

few minutes. For most automation of business processes this is fine because people using the system don't need to be consciously aware that the process is running.

Use real-time workflows when a business process requires someone to immediately see the results of the process or if you want the ability to cancel an operation. For example, you may want to set certain default values for a record the first time it is saved, or you want to make sure that some records are not deleted.

You can change a real-time workflow into a background workflow by choosing Convert to a background workflow on the toolbar.

You can change a background workflow into a real-time workflow by choosing Convert to a real-time workflow on the toolbar. If the background workflow uses a wait conditions it will become invalid and you won't be able to activate it until you remove the wait condition.

When you configure Options for Automatic Processes for real-time workflows, the Start When options for the status changes event let you select After or Before for when status changes. The default option is After.

When you select Before you are saying that you want the logic in the workflow to be applied before data changing the status is saved. This provides you with the ability to check the values before other logic has been applied after the operation and prevent further logic from being performed. For example, you may have additional logic in a plugin or custom workflow action which could initiate actions on another system. By stopping further processing you can avoid cases where external systems are affected. Applying real-time workflows before this event also means that other workflow or plug-in actions in Microsoft Dynamics CRM that may have saved data don't need to be "rolled back" when the operation is cancelled.

When you apply a Stop Workflow action in a workflow you have the option to specify a status condition that can be either Succeeded or Canceled. When you set the status to canceled, you prevent the operation. An error message containing the text from the stop action status message will be displayed to the user with the heading Business Process Error.

Security Context

When a background workflow is configured as an on-demand process and is started by a user using the Run Workflow command, the actions that the workflow can perform are limited to those the user could perform based on the privileges and access levels defined by the security role(s) set for their user account.

When a background workflow starts based on an event the workflow operates in the context of the person who owns it, usually the person who created the workflow.

For real-time workflows you have the Execute As option and you can choose whether the workflow should apply the security context of the owner of the workflow or the user who made changes to the record. If your workflow includes actions which all users would not be able to perform based on security constraints, you should choose to have the workflow run as the owner of the workflow.

Lesson 4-4 Dialogs

Dialogs

Use this process to create an interactive step-by-step data entry form that requires user input to start and run to completion. When you start the dialog process, a wizard-like interface is presented so you can make appropriate selections or enter data as you progress through each page of the wizard.

Dialogs are wizards that allow you to collect input from a user and then create or update records. They are particularly useful in initial data capture.

Dialogs consist of Pages with one or more Prompt/Responses to capture user input.

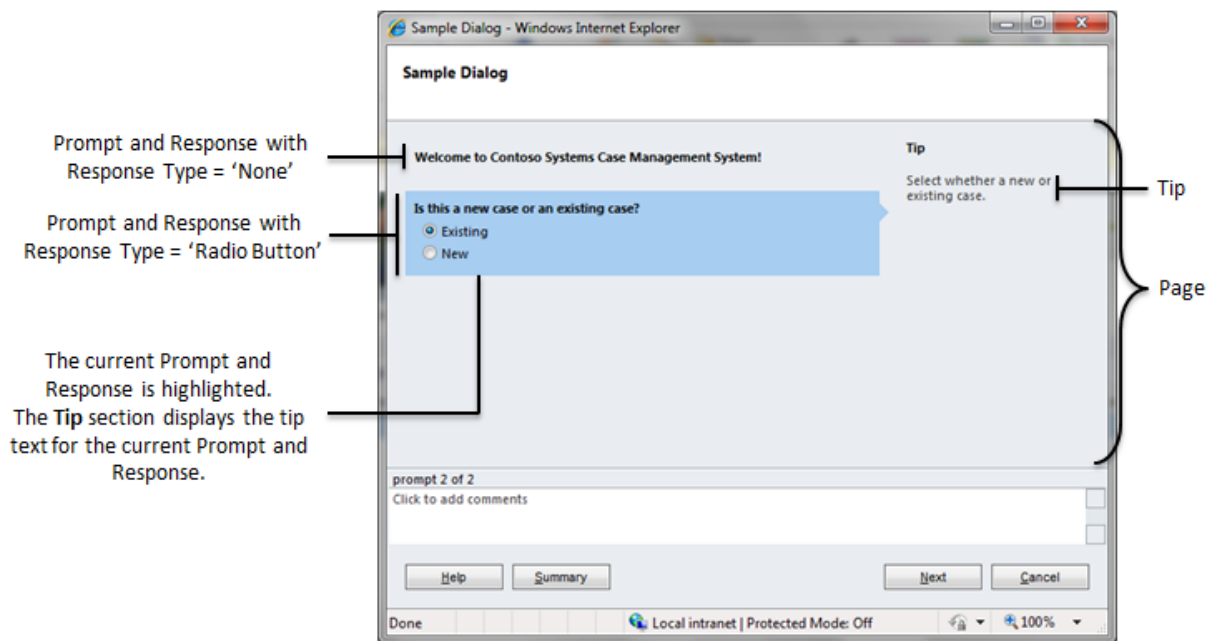


Figure 7 - Dialog page

Dialogs provide:

- Consistent customer interactions and interactive user tasks.
- Consistent information entry into your organization's database.
- A way for people in your organization to focus on growing your business, instead of performing repetitive tasks.

Dialogs share many of the conditions and actions with workflows.

Unlike workflow processes, dialogs do not have scope. They are available to the entire organization. If a user runs a dialog that creates or updates record, the user must have privileges to perform those actions outside the dialog. Each dialog will create a Dialog session record and the user must have privileges to create and update those records

Unlike workflow processes, a dialog can only be applied to one record at a time.

Dialog Properties

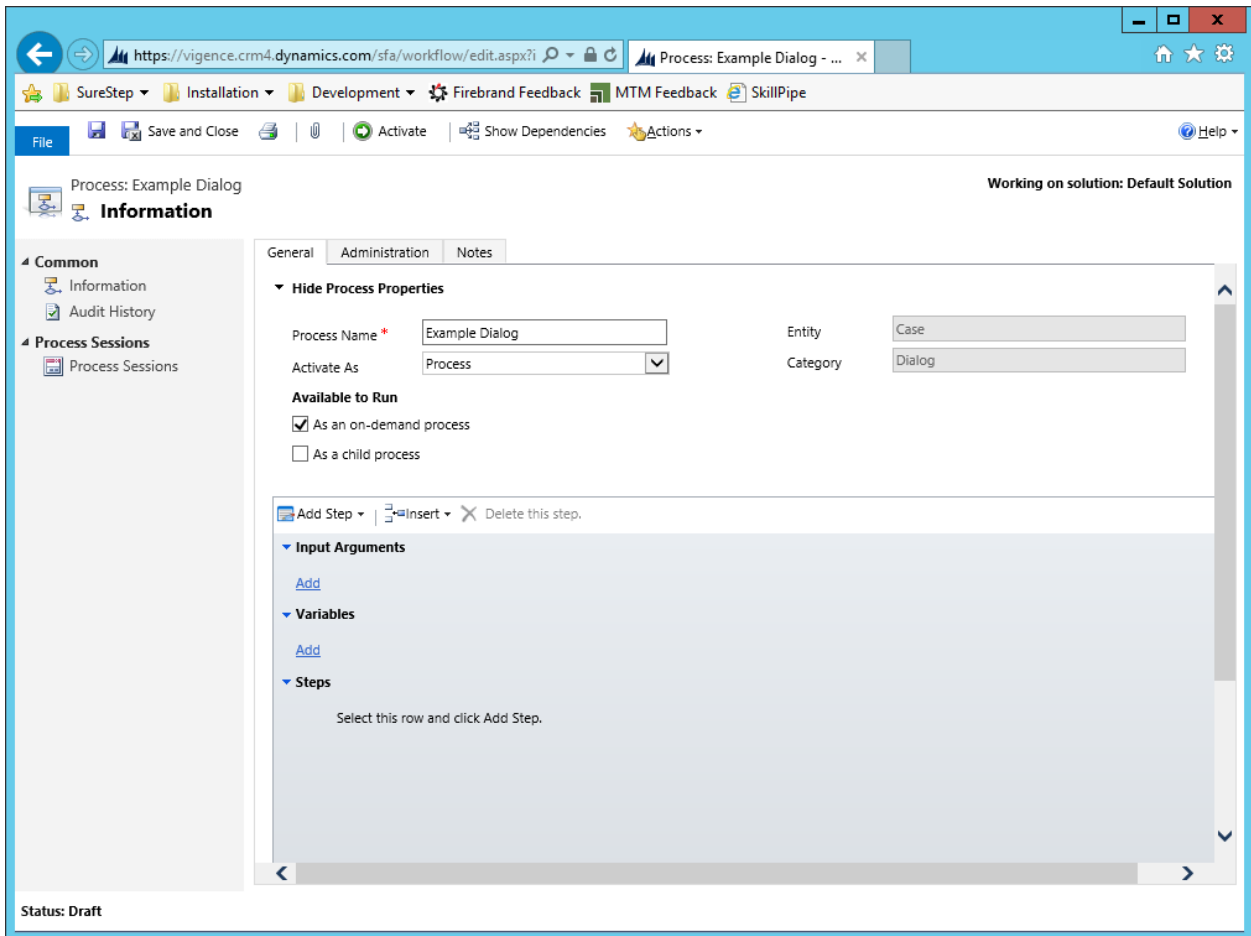


Figure 8 - Dialog Designer

Property	Description
Activate As	<p>You can choose Process template to create an advanced starting point for other Dialogs. If you choose this option, after you activate the Dialog it will not be applied but instead it will be available to select in the Create Process dialog if you select Type: New process from an existing template (select from list)</p> <p>Process templates are convenient when you have a number of similar dialog processes and want to define them without duplicating the same logic</p>
As an on-demand process	Choose this option if you want to allow users to run this workflow from the Run Workflow command
As a child	Choose this option if you want to allow the workflow to be available to be started

Property	Description
process	from another workflow or dialog.

Input Arguments

Unlike workflows, dialogs can have input arguments that allow one dialog process to pass values to a child dialog process.

If you attempt to enter Input arguments for a process configured as an on-demand process, you will be prompted to change the Available to Run value to As a Child process. After you enter Input arguments, you will not be able to set the process to be an on-demand process until all the input arguments have been removed.

Input arguments can be of the following types:

- Single Line of Text
- Whole Number
- Floating Point Number
- Date and Time
- Date Only
- Lookup

With each type, you can set a default value to be used if the calling dialog doesn't provide data to the input argument.

Variables

Unlike workflows, dialogs can have variables that can be set using the Assign Value step.

Variables are useful when a process gathers data through the course of several pages and this data may be used to perform calculations. For example, a dialog might be used to calculate a standard rating value based on the answers to several questions.

Prompt and Response

A Prompt and Response step has the following properties:

Property	Description
Statement Label	The statement label should provide an appropriate heading for the Prompt Text. The Statement Label is visible in the dialog session when viewing the summary during or after the dialog is completed.
Prompt Text	Prompt Text may represent something the person using the dialog should say to the customer or it could include instructions about how to complete a step of a complex procedure.

Property	Description
Tip Text	Tip text provides additional information to support the Prompt Text.
Response Type	<p>Choose one of the following Response Types:</p> <p>None - a prompt without a response.</p> <p>Single Line - A single line can represent a text, integer or float data by setting the Data Type.</p> <p>Option Set (Radio Buttons)</p> <p>The results are presented as a set of Radio buttons. Use this option when there are just a small number of options to choose from.</p> <p>The data selected can be set to text, integer or float data by setting the Data Type.</p> <p>You can choose to define static values or query CRM Data to provide a list of options.</p> <p>Options Set (Picklist) - This is exactly like Option Set (Radio Buttons), except that the options are displayed as a list.</p> <p>Multiple Lines (Text Only) - Provides an area to type text with multiple lines.</p> <p>Date and Time - provides a control to set a date and a time.</p> <p>Date Only - Provides a control to set a date.</p> <p>Lookup - This option will present one of the lookup fields used in the application.</p>
Data Type	<p>When you select a Response Type of Single Line, Option Set (radio buttons), or Option Set (picklist), you can choose to have the data set in the control be expressed using one of the following data types:</p> <ul style="list-style-type: none"> • Text • Integer • Float <p>When you select a Response Type of Lookup, the Data Type field is replaced by the Reference Entity field.</p>
Log Response	<p>When you choose to not log responses you will still be able to access the responses as variables within your dialog, but the data in the response will not be saved with the dialog session. This is a security feature. Consider if you have a dialog that requires some personal information to be entered and processed. If the response is not logged it will not be saved with the dialog session record that contains the data in the dialog summary</p>

Property	Description
Default Value	Use default value to set a value to indicate that the data in the response was not provided or represents a very common response which would only need to be edited if it was different.

Lesson 4-5 Business Process Flows

Business Process Flows

Use this process to create a visualisation of the business process flow. Users are guided through various stages of the sales or customer service processes. At each stage, you complete specific steps and then move to the next stage. You can customize the process flow by adding or removing steps, changing the order stages, or adding new entities to the process flow.

Multiple processes can be added to each record type - think sales process for inside sales vs. outside sales. A sales organization can make sure each required step in each pipeline phase is completed before moving to the next pipeline phase.

What can business process flows do

With business process flows, you define a set of stages and steps that are then displayed in a control at the top of the form.

As a user you will see a process bar at the top of the screen for many of the record types. With business processes, each stage for working with a customer is clearly outlined. Steps to complete your work are easy to follow.

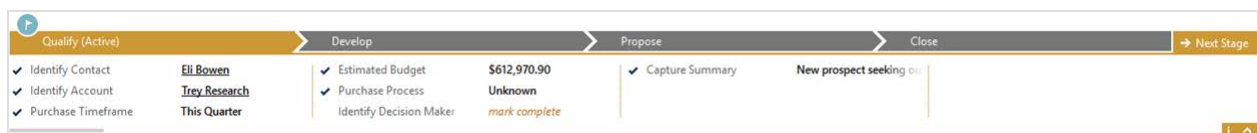


Figure 9 - Process Flow

Each stage contains a group of steps. Each step represents a field where data can be entered. People advance to the next stage by using the Next Stage button. You can make a step required so that people must enter data for the corresponding field before they can proceed to the next stage. This is commonly called "stage-gating".

Business process flows appear relatively simple compared to other types of processes because they do not provide any conditional business logic or automation beyond providing the streamlined experience for data entry and controlling entry into stages. However, when you combine them with other processes and customizations, they can play an important role in saving people time, reducing training costs, and increasing user adoption.

System business process flows

Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online include three business process flows:

- Lead to Opportunity Sales Process
- Opportunity Sales Process
- Phone to Case Process

Multiple entities in business process flows

You can use a business process flow for a single entity or span multiple entities. For example, you may have a process that begins with an opportunity, then continues to a quote, an order, and then an invoice, before finally returning to close the opportunity.

You can design business process flows that tie together the records for up to five different entities into a single process so that people using Microsoft CRM can focus on the flow of their process rather than on which entity they are working in. They can more easily navigate between related entity records.

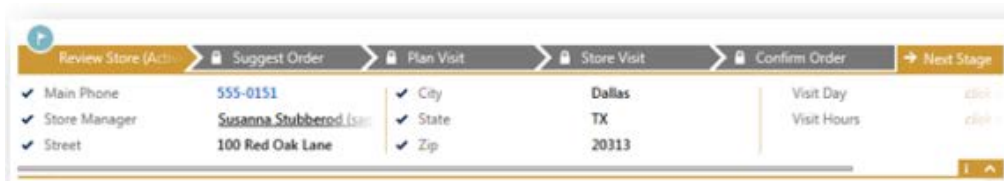


Figure 10 - Process flow over multiple entities

Multiple business process flows are available per entity

Not every user in an organization may follow the same process and different conditions may require that a different process be applied. You can have up to 10 active business process flows per entity to provide appropriate processes for different situations.

Control which business process flow will be applied

You can associate business process flows with security roles so that only people with those security roles can see or use them. You can also set the order of the business process flows so that you can control which business process flow will be set by default. This works in the same way that multiple forms for an entity are defined.

When someone creates a new entity record, the list of available activated business process flows is compared to the business processes flows that the person's security role will show them. The first activated business process flow in that list is the one that will be applied by default. If more than one active business process flow is available, people can choose Switch Process from the command bar to apply a different process. Whenever someone switches processes, the current process stage will be set to the first stage of the newly applied business process flow.

Each record can have only one business process flow at a time. When any user applies a different process, that process is the one that the next user to view the record will see. If someone’s security roles do not allow them to use a specific business process flow, the current business process flow will be visible, but disabled.

Business process flow limitations

You can define business process flows only for those entities that support them.

To ensure acceptable performance and the usability of the user interface, there are some limitations you need to be aware of when you plan to use business process flows:

- There can be no more than 10 activated business process flow processes per entity.
- Each process can contain no more than 30 stages.
- Multi-entity processes can contain no more than five entities

Business Process Flow Properties

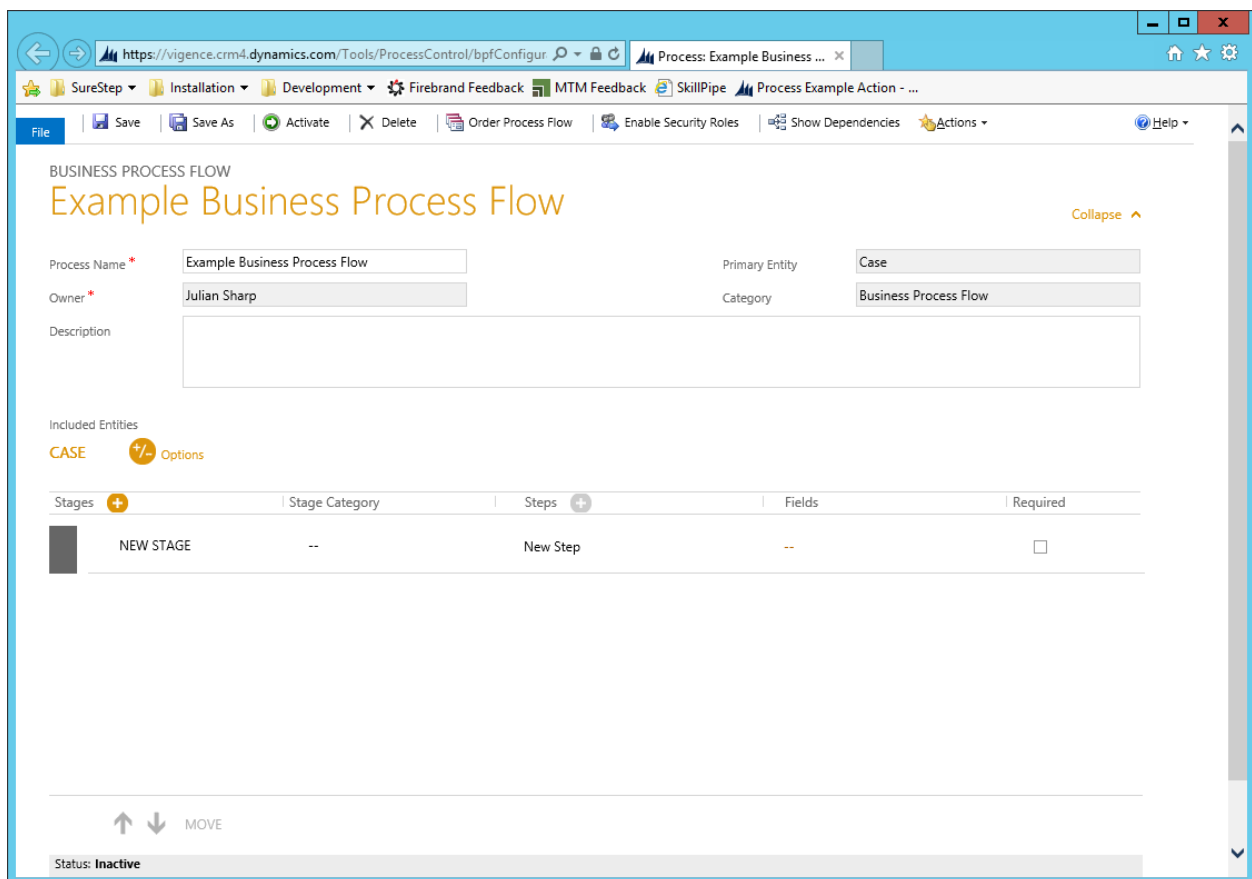


Figure 11 - Business Process Flow Designer

Business process flows integrated with other customizations

When you or your user enters data using business process flows, the data changes are also applied to form fields so that any automation provided by business rules or form scripts can be applied immediately.

Steps can be added that set values for fields that are not present in the form and these fields will be added to the Xrm.Page object model used for form scripts. Any workflows that are initiated by changes to fields included in a business process flow will be applied when the data in the form is saved. If the automation is applied by a real-time workflow, the changes will be immediately visible to the user when the data in the form is refreshed after the record is saved.

Although the business process flow control in the form does not provide any direct client-side programmability, changes applied by business rules or form scripts are automatically applied to business process flow controls. If you hide a field in a form, that field will also be hidden in the business process flow control. If you set a value by using business rules or form scripts, that value will be set within the business process flow.

Lesson 4-6 Actions

Actions

Use this process to create a new operation that is not available in a stock Microsoft Dynamics CRM installation or to combine multiple disparate operations into a single operation. For example, in the case of a support call centre, you could combine create, assign, and setstate operations into a single new “escalate” operation.

This provides an intuitive way to write simple code that was otherwise reserved for someone with development / coding experience.

Why use actions

Actions open a range of possibilities for developers and people who compose business logic. Before Actions, the primary way that developers could implement business processes was limited to plug-ins or custom workflows. With these, developers can perform operations composed of verbs like Create, Update, Delete, Assign, and SetStatus. Each of these messages is based on actions taken on an entity instance. So if the goal of a process is to create a record, then update it, then assign it, there are three separate steps. Each step is defined by the capabilities of the entity – not necessarily your business process.

Actions provide the ability to define a single verb (or message) that matches an operation you need to perform for your business. These new messages are driven by a process or behaviour rather than what can be done with an entity. These messages can correspond to verbs like Escalate, Convert, Schedule, Route, or Approve – whatever you need. The addition of these verbs helps provide a richer vocabulary for you to fluently define your business processes. You can apply this richer vocabulary from clients or integrations rather than having to write the action within clients. This also makes it easier because you can manage and log the success or failure of the entire action as a single unit.

Effectively, Developers can add new "messages" to the CRM Event Pipeline.

Configurable messages

Once an action is defined and activated, a developer can use that message like any of the other messages provided by the Microsoft Dynamics CRM platform. However, a significant difference is that now someone who is not a developer can apply changes to what should be done when that message is used. You can configure the action to modify steps as your business processes change. Any custom code that uses that message does not need to be changed as long as the process arguments do not change.

Workflow processes and plugins continue to provide similar capabilities for defining automation. Workflow processes still provide the capability for a non-developer to apply changes. But the difference is in how the business processes are composed and how a developer can write their code. An action is a message that operates on the same level as any of the messages provided by the Microsoft Dynamics CRM Platform. Developers can even create plugins for Actions.

Global messages

Unlike workflow processes or plugins, an action doesn't have to be associated with a specific entity. You can define "global" Actions that can be called on their own.

Actions limitations

For Microsoft Dynamics CRM 2013, Actions can be called only from code. You can't call an action from a workflow or other process. So for now the most common ways to invoke Actions will be:

- From code that executes within a plugin or custom workflow.
- From a command that is placed in the application and executes the operation using JavaScript code.
- From an integration with another system that uses the Microsoft Dynamics CRM web services.
- From a custom client application that uses the Microsoft Dynamics CRM web services.

Configure actions

You may need to create an action so that a developer will use it in code or you may need to edit an action that was previously defined. Like workflow processes, consider the following:

- What actions should they perform?
- Under what conditions actions should be performed?

Unlike workflow processes, you don't need to set the following options:

- Start When: Actions start when code calls the message generated for them.

- Scope: Actions always run in the context of the calling user.
- Run in the background: Actions are always real-time workflows.

An action also has something that workflow processes don't have – input and output arguments.

Action Properties

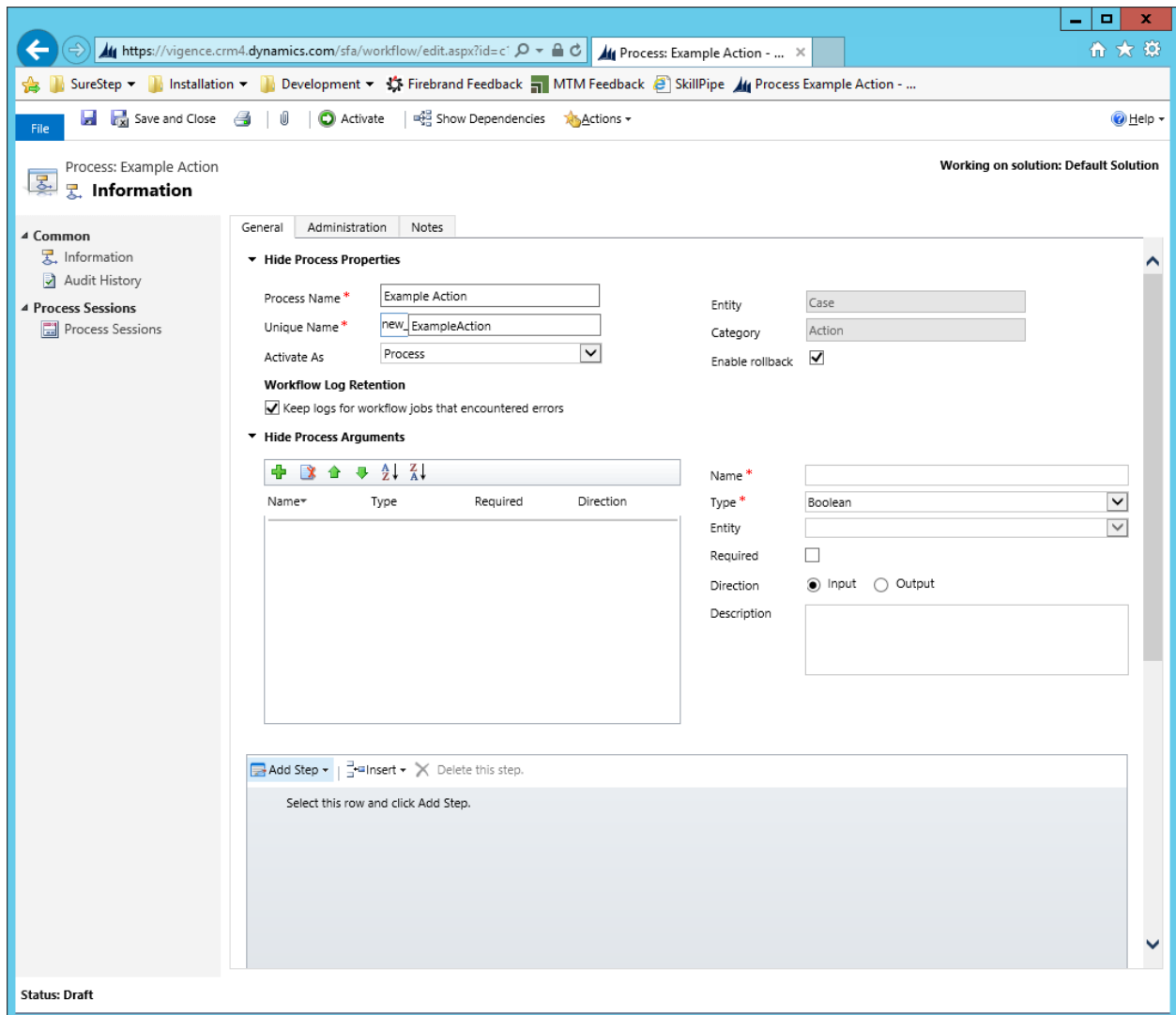


Figure 12 - Action Designer

Property	Description
Activate As	<p>You can choose Process template to create an advanced starting point for other Actions. If you choose this option, after you activate the Action it will not be applied but instead it will be available to select in the Create Process dialog if you select Type: New process from an existing template (select from list)</p> <p>Process templates are convenient when you have a number of similar action processes</p>

Property	Description
	and want to define them without duplicating the same logic
In transaction	<p>Generally, processes that support transactions will “undo” (or rollback) the entire operation if any part of them fails. There are some exceptions to this. Some actions developers might do in code initiated by the action might not support transactions. For example, if the code perform actions in other systems that are beyond the scope of the transaction. Those can’t be rolled back by the action running in Microsoft Dynamics CRM. Some messages in the CRM platform don’t support transactions. But everything you can do just with the user interface of the action will support transactions. All the actions that are part of a real-time workflow are considered in transaction, but with actions you have the option to opt out of this.</p> <p>You should consult with the developer who will use this message to determine whether it must be in transaction or not. Generally, an action should be in transaction if the actions performed by the business process don’t make sense unless all of them are completed successfully. The classic example is transferring funds between two bank accounts. If you withdraw funds from one account you must deposit them in the other. If either fails, both must fail.</p>

Process Arguments

When a developer uses a message they may begin with some data that they can pass into the message and use. For example, if you want to create a new case record, you may have the case title value that will be passed in as an argument. This would be an input argument.

When the message is finished the developer may need to pass some data that was changed or generated by the message to another operation in their code. These must be defined as an output argument.

Both input and output arguments must have a name, a type, and some information about whether the argument is always required. You can also provide a description.

The name of the message and the information about all the process arguments represent the “signature” for the message. After an action is activated and it is being used in code, the signature must not change. Changing this signature will cause any code that uses the message to fail. The only exception to this may be changing one of the parameters so that it is not always required.

Changing the order of the arguments by sorting them or moving them up or down doesn’t make a difference because the arguments are identified by name, not by the order. Changing the description will not break code using the message.

The following table describes the action process argument types.

Type	Description
Boolean	A true or false value.
DateTime	A value that stores date and time information.
Decimal	A number value with decimal precision. Used when precision is extremely important.
Entity	A CRM record for the specified entity. When you select Entity, the drop-down is enabled and allows you to select the entity type.
EntityCollection	A collection of entity records.
EntityReference	An object that contains the name, id, and type of an entity record that uniquely identifies it. When you select EntityReference, the drop-down is enabled and allows you to select the entity type.
Float	A number value with decimal precision. Used when data comes from a measurement that isn't absolutely precise.
Integer	A whole number.
Money	A value that stores data about an amount of money.
Picklist	A value that represents an option for an OptionSet attribute.
String	A text value.

Module 5 – Server Side Code

The key objectives of this module are to provide an understanding of the different types of server side code that can be used in Dynamics CRM.

Objectives

The key objectives of this module are to:

- Understand Plugins
- Understand Custom Workflow Activities
- When to use
- Debugging

Lesson 5-1 Plugins

Overview

The functionality of most Microsoft Dynamics CRM events can be extended by writing custom plug-ins.

A plug-in is a custom business logic that can integrate with Microsoft Dynamics CRM 2013 to modify or augment standard behaviour of the platform.

Plug-ins subscribe to a set of events and run when these events occur. Plugins can run code both before an event performs its operations and after an event completes its operations. Plugins are implemented in code held in a custom assembly.

These events occur regardless of the method that is used to perform the activity so they are independent of the user interface.

Registration of plugins is performed through custom code that uses classes available in the SDK. The SDK provides a tool to assist with plug-in registration.

Basics

Plugins have many uses. This includes the following:

- Performing complex platform level data validation
- Performing auto-number generation
- Providing integration with other applications
- Executing complex business logic

Any number of plug-ins can be associated with a given entity and event. When multiple plugins are registered for the same event on the same entity, they are called in a sequence based on the order specified on the step registration. This value is specified as the Rank and it is supplied when the plug-in is registered. This allows developer control over the sequence.

Plug-ins can be written in any Microsoft .NET 4.0 CLR-compliant language i.e., Microsoft Visual C# and Microsoft Visual Basic .NET.

Authentication

Plug-ins not executed by the sandbox or asynchronous service execute under the security context of the Microsoft Dynamics CRM application pool, CrmAppPool, as defined in Internet Information Services.

By default, CrmAppPool uses the Network Security account identity. Therefore, it is important to set credentials correctly. Many times, it might be necessary to impersonate the user whose actions caused the plug-in to run.

Lesson 5-2 Event Framework

Event Framework

The event framework is the term that is used to describe the technology and mechanisms available in Microsoft Dynamics CRM to extend or customize functionality with custom business logic.

The event framework enables you to create rich vertical and horizontal solutions on top of Microsoft Dynamics CRM by supporting the development and integration of custom business logic with Microsoft Dynamics CRM in a reliable and portable way. After your custom business logic has been integrated into Microsoft Dynamics CRM, it can be executed synchronously as part of the main Microsoft Dynamics CRM execution path, or asynchronously from a managed queue. Business data can be passed to your custom code, which can then perform actions based on the nature of the information, or modify the information itself.

The Event Framework provides the following key features:

- An improved event processing subsystem. This subsystem provides a unified method of executing both plug-ins and workflow activities, which results in improved reliability, an enhanced feature set, and plug-in portability.
- An event framework API for extending the Microsoft Dynamics CRM platform through the development of custom business logic in the form of plug-ins and workflow activities.
- An API for the deployment of plug-ins and custom workflow activities to the Microsoft Dynamics CRM database. Deployment of plug-ins and workflow activities to the database enables automatic distribution of your plug-ins and custom workflow activities to servers running Microsoft Dynamics CRM services throughout a datacenter.
- Backwards compatibility for Microsoft Dynamics CRM 2011 plug-ins.
- Synchronous and asynchronous execution of plug-ins. Synchronous plug-ins are executed in a pre-defined order as part of the main Microsoft Dynamics CRM event processing. Asynchronous plug-ins are queued and executed independently.

Architecture

The Microsoft Dynamics CRM event processing subsystem executes plug-ins based on a message pipeline execution model. A message is sent to the organization Web service when there's an SDK method call by a plug-in or other application. The message contains business entity information and core operation information.

The message is passed through the event execution pipeline where it can be read or modified by the platform core operation and any registered plug-ins.

The following figure illustrates the overall architecture that Microsoft Dynamics CRM platform with respect to synchronous and asynchronous processing.

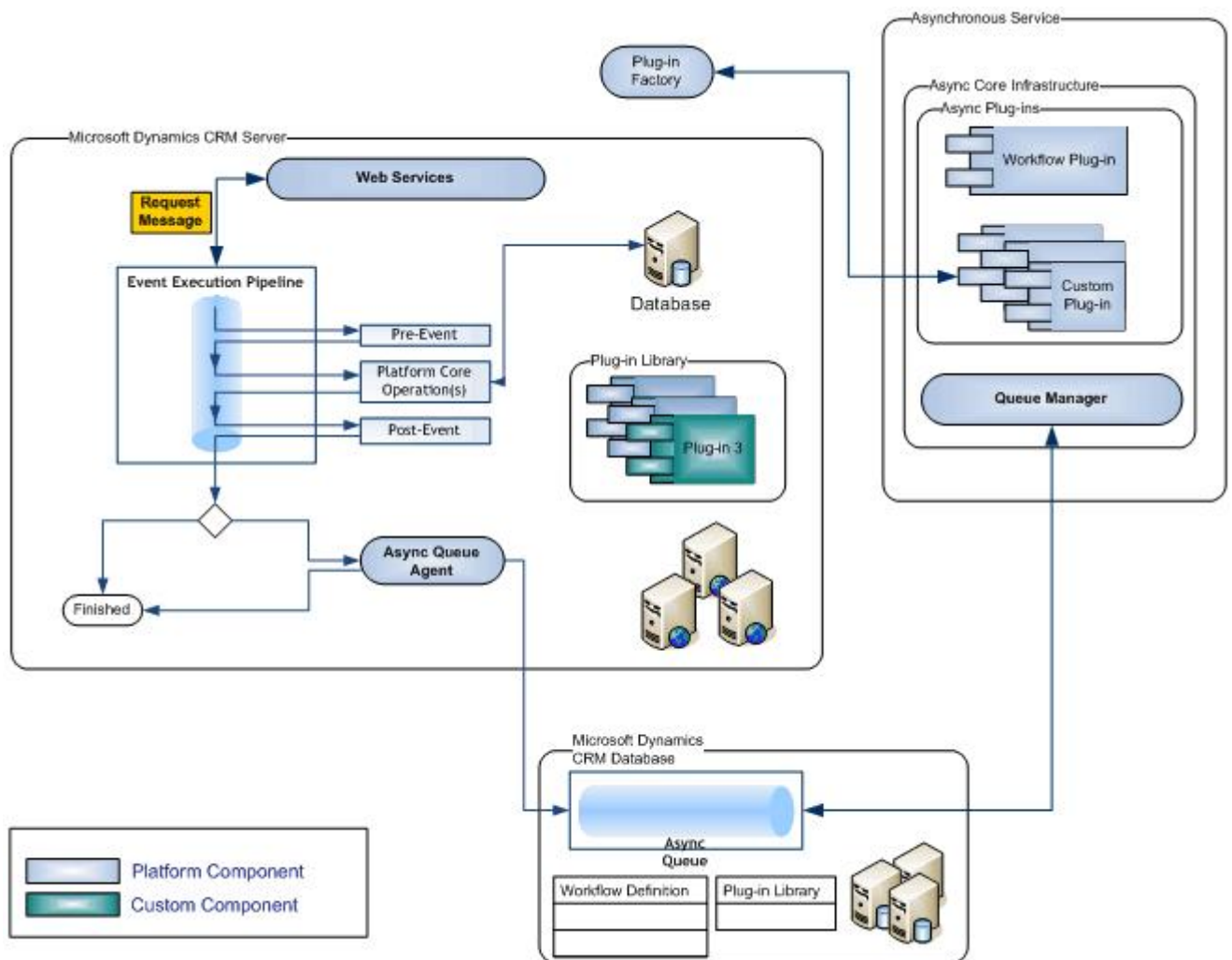


Figure 13 - Event Execution Pipeline

The event execution pipeline can process events synchronously or asynchronously. The platform core operation and any plug-ins registered for synchronous execution are executed immediately.

Synchronous plug-ins registered for the event are executed in a well-defined order.

Plug-ins registered for asynchronous execution are queued with the asynchronous service and executed in the future.

Pipeline Stages

The event pipeline is divided into multiple stages with four of them available to register custom developed plug-ins. Multiple plug-ins that are registered in each stage can be further ordered or ranked in that stage during the plug-in registration. Any particular plug-in registered can reside in only one particular stage. This means that if a plug-in has to operate in multiple stages it has to be registered one time for each stage.

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. Plug-ins registered in this stage are executed within the database transaction.
Platform Core Operation	MainOperation	30	In-transaction 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.

Lesson 5-3 Plugin Isolation, Trusts and Statistics

Isolation

Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online support the execution of plug-ins and custom workflow activities in an isolated environment. In this isolated environment, also known as a sandbox, a plug-in or custom activity can make use of the full power of the Microsoft Dynamics CRM SDK to access the organization web service. Access to the file system, system event log, certain network protocols, registry, and more is prevented in the sandbox. However, sandbox plug-ins and custom activities do have access to external endpoints like the Windows Azure cloud service.

Microsoft Dynamics CRM collects run-time statistics and monitors plug-ins and custom workflow activities that execute in the sandbox. If the sandbox worker process that hosts this custom code exceeds threshold CPU, memory, or handle limits or is otherwise unresponsive, that process will be killed by the platform. At that point any currently executing plug-in or custom workflow activity in that worker process will fail with exceptions. However, the next time that the plug-in or custom

workflow activity is executed it will run normally. There is one worker process per organization so failures in one organization will not affect another organization.

In summary, the sandbox is the recommended execution environment for plug-ins as it is more secure, supports run-time monitoring and statistics reporting, and is supported on all Microsoft Dynamics CRM deployments. In addition, Microsoft Dynamics CRM Online only supports execution of plugins and custom workflow activities if they are registered in the sandbox.

Trusts

Developers have the option of registering their plug-ins in the sandbox, known as partial trust, or outside the sandbox, known as full trust. Full trust is supported for on-premises and Internet-facing Microsoft Dynamics CRM deployments. For a Microsoft Dynamics CRM Online deployment, plug-ins or custom workflow activities must be registered in the sandbox (partial trust) where they are isolated as previously described.

Run-time statistics

The Microsoft Dynamics CRM platform collects run-time information about plug-ins and custom workflow activities that execute in the sandbox. This information is stored in the database using PluginTypeStatistic entity records. These records are populated within 30 minutes to one hour after the sandboxed custom code executes. See the PluginTypeStatistic attributes to find out what information is collected. You can retrieve this information by using the retrieve message or method.

Access

Sandboxed plug-ins and custom workflow activities can access the network through the HTTP and HTTPS protocols. This capability provides support for accessing popular web resources like social sites, news feeds, web services, and more. The following web access restrictions apply to this sandbox capability.

- Only the HTTP and HTTPS protocols are allowed.
- Access to localhost (loopback) is not permitted.
- IP addresses cannot be used. You must use a named web address that requires DNS name resolution.
- Anonymous authentication is supported and recommended. There is no provision for prompting the logged on user for credentials or saving those credentials

Lesson 5-4 Developing Plugins

IPlugin Interface

Plug-ins are custom classes that implement the IPlugin interface. You can write a plug-in in any .NET Framework 4 CLR-compliant language such as Microsoft Visual C# and Microsoft Visual Basic .NET.

The `IServiceProvider` parameter of the `Execute` method is a container for several service useful objects that can be accessed within a plug-in. The service provider contains instance references to the execution context, `IOrganizationServiceFactory`, `ITracingService`, and more

Context

The execution context contains a wealth of information that describes the run-time environment that the plug-in is executing in, information related to the execution pipeline, and entity business information.

When a system event is fired for which a plug-in is registered, the system creates and populates the context and passes it to a plug-in through the proceeding sample that mentions classes and methods.

The execution context is passed to each registered plug-in in a pipeline when they are executed.

Each plug-in in the execution pipeline can modify writable properties in the context.

For example, given a plug-in registered for a pre-event, another for a plug-in registered for a post-event, the post-event plug-in can receive a context that has been modified by the pre-event plug-in. The same situation applies to plug-ins that registered in the same stage.

All the properties in the `iPlug-in` execution context are read only. However, plug-ins can modify the contents of these properties that are collections.

The `input parameters` property contains the data that is in the request message currently being processed by the event execution pipeline. Plug-in code can access this data.

The `properties` is a type parameter collection where the keys to the access that requested data are the names of the actual public properties in the request.

For example, if the `create` request method is used, one property of that request is named "target," which is of the type `entity`. This is the entity currently being operated on by the platform.

To access the data of the entity, use the name "target" as the key in the `input parameters` collection. It is necessary to cast the returned instance.

Similarly, the `output parameters` property contains the data that is in the response message, such as a "create" response currently being passed through the event execution pipeline.

However, only synchronous post-event and asynchronous registered plug-ins have output parameters populated because the response is the result of the core platform operation.

Note: If a plug-in is registered as a pre-event, the `output parameters` property bag does not contain a value for the ID key because the core operation has not yet occurred.

Assemblies Required

To be able to compile plug-in code, you must add Microsoft.Xrm.Sdk.dll and Microsoft.Crm.Sdk.Proxy.dll assembly references to your project. These assemblies can be found in the SDK\Bin folder of the SDK.

AutoSave

Your plug-in design should take into account a web form feature known as auto-save. In the web application entity forms, there is no Save button. The Microsoft Dynamics CRM web application automatically saves changed data in the form when needed. Depending on how you registered your plug-in, this may result in your plug-in being called very frequently for individual field changes instead of one plug-in invocation for all changes. The auto-save feature only applies to web forms for contact, opportunity, leads, account, and case entities.

It is a best practice to register your plug-in or custom workflow activities on entities and specific fields that matter most. If you can avoid it, do not register your plug-in for changes to all entity fields.

Assembly Versioning and Solutions

Plug-in assemblies can be versioned using a number format of Major.Minor.Build.Revision defined in the Assembly.info file of the Microsoft Visual Studio 2010 project. Depending on what part of the assembly version number is changed in a newer solution, the following behaviour applies when an existing solution is updated through import.

- The build or revision assembly version number is changed.

This is considered an in-place upgrade. The older version of the assembly is removed when the solution containing the updated assembly is imported. Any pre-existing steps from the older solution are automatically changed to refer to the newer version of the assembly.

- The major or minor assembly version number, except for the build or revision numbers, is changed.

When an updated solution containing the revised assembly is imported, the assembly is considered a completely different assembly than the previous version of that assembly in the existing solution. Plug-in registration steps in the existing solution will continue to refer to the previous version of the assembly. If you want existing plug-in registration steps for the previous assembly to point to the revised assembly, you will need to use the Plug-in Registration tool to manually change the step configuration to refer to the revised assembly type. This should be done before exporting the updated assembly into a solution for later import.

Lesson 5-5 Deploying Plugins and Impersonation

Signing Assemblies

All plug-in assemblies must be signed before they can be registered in Microsoft Dynamics CRM 2013

Storage

Plug-ins not registered in the sandbox can be stored in the Microsoft Dynamics CRM server's database or the on-disk file system. It is strongly recommended to store production-ready plug-ins in the Microsoft Dynamics CRM database, instead of on-disk.

Plug-ins stored in the database are automatically distributed across multiple Microsoft Dynamics CRM servers in a data centre cluster and are included in database backups and redeployments. On-disk storage of plug-ins is useful for debugging plug-ins using Microsoft Visual Studio.

Plug-ins registered in the sandbox must be stored in the database.

For on-premise or Internet-Facing Microsoft Dynamics CRM installations, when deploying plug-ins on-disk, the plug-in assembly must be manually copied to the server before registration. The assembly must be deployed to the following folder:

```
<installdir>\Program Files\Microsoft CRM\server\bin\assembly
```

Impersonation

Impersonation is used to execute business logic on behalf of a Microsoft Dynamics CRM system user to provide desired feature or service for that user.

Impersonation During Registration

One method to impersonate a system user in a plug-in is by specifying the impersonated user during plug-in registration. When registering a plug-in programmatically, if the `ImpersonatingUserId` property of `SdkMessageProcessingStep` is set to a specific Microsoft Dynamics CRM system user, web service calls made by the plug-in execute on behalf of the impersonated user.

If `ImpersonatingUserId` is set to a value of `null` or `Guid.Empty` during plug-in registration, the initiating user or the standard "system" user is the impersonated user.

Whether the calling/logged on user or "system" user is used for impersonation is dependent on the request being processed by the pipeline

Impersonation During Execution

Impersonation that is defined during plug-in registration can be altered in a plug-in at run time. Even if impersonation was not defined at plug-in registration, plug-in code can still use impersonation.

The platform passes the impersonated user ID to a plug-in at run time through the `UserId` property. This property can have one of three different values as shown in the table below

UserId Value	Condition
Initiating user or "system" user	The SdkMessageProcessingStep.ImpersonatingUserId attribute is set to null or Guid.Empty at plug-in registration.
Impersonated user	The ImpersonatingUserId property is set to a valid system user ID at plug-in registration.
"system" user	The current pipeline was executed by the platform, not in direct response to a service method call.

Another option is to impersonate the user whose actions initiated the plug-in execution. This can be done by using the InitiatingUserId property

The InitiatingUserId property of the execution context contains the ID of the system user that called the service method that ultimately caused the plug-in to execute.

Steps

Each plugin requires at least one step. A step defines:

- Message (Event)
- Entity
- Impersonation
- Rank
- Stage
- Synchronous or Asynchronous
- Availability in Outlook Offline

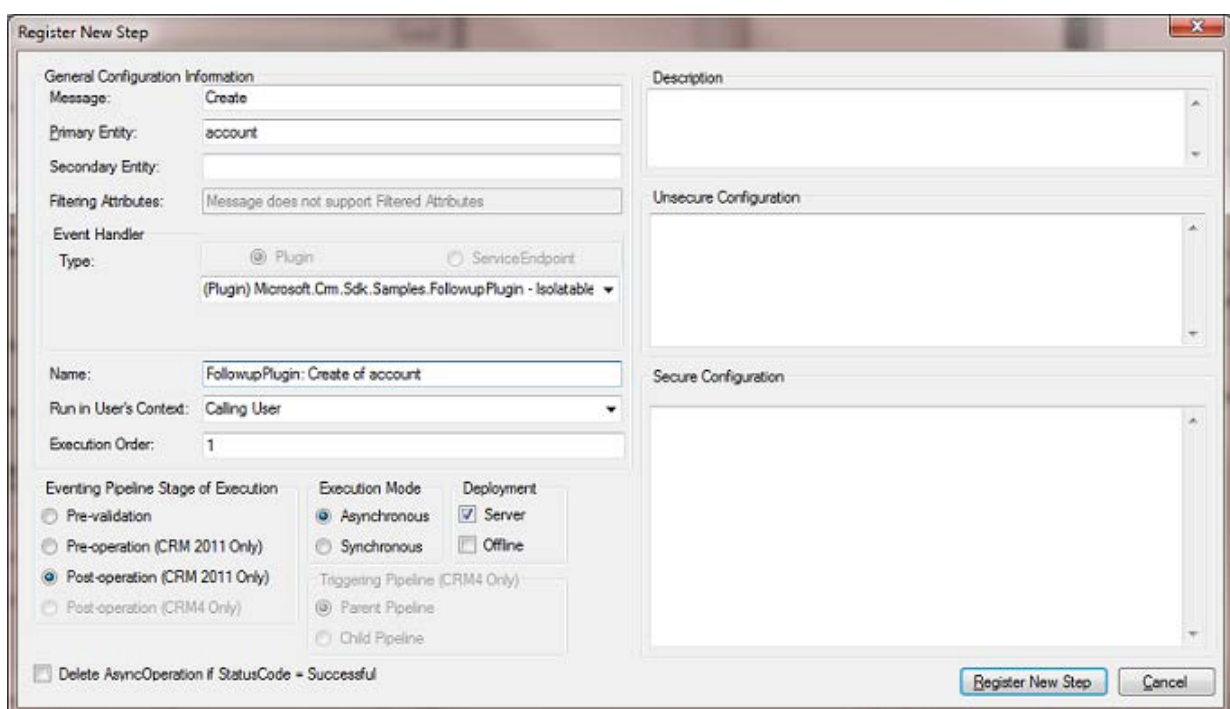


Figure 14 - Register Plugin Step

Configuration

Secure Configuration and Unsecure Configuration can be used to hold configuration information that a plugin can reference such as:

- usernames and passwords for other systems
- URL to web services
- XML configuration data

The plugin can access this information at runtime. This you can write generic plugins or plugins that need to call access different services e.g. different URLs for DEV, TEST and PROD

Unsecure information is accessible through the User Interface under Solutions.

Secure information is accessible through a separate privilege.

Pre and post entity images

A step can have Pre and Post entity images specified

PreEntityImages and PostEntityImages contain snapshots of the primary entity's attributes before (pre) and after (post) the core platform operation. Microsoft Dynamics CRM populates the pre-entity and post-entity images based on the security privileges of the impersonated system user. Only entity attributes that are set to a value or null are available in the pre or post entity images.

You can specify to have the platform populate these PreEntityImages and PostEntityImages properties when you register your plug-in. The entity alias value you specify during plug-in registration is used as the key into the image collection in your plug-in code.

There are some events where images are not available. For example, only synchronous post-event and asynchronous registered plug-ins have PostEntityImages populated. The create operation does not support a pre-image and a delete operation does not support a post-image. In addition, only a small subset of messages support pre and post images as shown in the following table

Message Request	Property	Description
AssignRequest	Target	The assigned entity.
CreateRequest	Target	The created entity.
DeleteRequest	Target	The deleted entity.
DeliverIncomingEmailRequest	EmailId	The delivered email ID.
DeliverPromoteEmailRequest	EmailId	The delivered email ID.
ExecuteWorkflowRequest	Target	The workflow entity.
MergeRequest	Target	The parent entity, into which the data from the child entity is being merged.

MergeRequest	SubordinateId	The child entity that is being merged into the parent entity.
SendEmailRequest	EmailId	The sent entity ID.
SetStateRequest	EntityMoniker	The entity for which the state is set.
UpdateRequest	Target	The updated entity

Registering for pre or post images to access entity attribute values results in improved plug-in performance as compared to obtaining entity attributes in plug-in code through RetrieveRequest or RetrieveMultipleRequest requests.

Plugin Registration

Plugins, Steps and Images must be registered in Dynamics CRM. There are SDK calls for you to build your own code to register your plugin. However, there are easier tools in the SDK.

Plugin Registration Tool

The Plug-in Registration tool provides a graphical user interface and supports registering plugins and custom workflow activities with Microsoft Dynamics CRM.

However, plug-ins and custom workflow activities can only be registered in the sandbox (isolation mode) of Microsoft Dynamics CRM Online.

The tool can be added to the Visual Studio Tools menu as an external tool to speed up the development process.

Developer Toolkit

If using the Developer toolkit, you can register a plugin from within Visual Studio using Deploy option on the CrmPackage.

Lesson 5-6 Debugging Plugins

Debugging Steps

To effectively troubleshoot errors, it is important to know how to appropriately debug plug-ins.

Register and deploy the plug-in assembly

If there is another copy of the assembly at the same location and you cannot overwrite that copy because it is locked by Microsoft Dynamics CRM, you must restart the service process that was executing the plug-in. Refer to the table below for the correct service process.

Copy the symbol file (.pdb) of the compiled plug-in assembly to the server\bin\assembly folder on the CRM server.

Configure the debugger

Attach the debugger to the process on the Microsoft Dynamics CRM server that will run your plug-in. Refer to the following table to identify the process.

Plug-in Registration Configuration	Service Process
Full Trust	w3wp.exe
Outlook Client Offline	Microsoft.Crm.Application.Host.exe
asynchronous registered plug-ins (or custom workflow assemblies)	CrmAsyncService.exe
sandbox (isolation mode)	Microsoft.Crm.Sandbox.WorkerProcess.exe

If there are multiple processes running the same executable file, for example multiple w3wp.exe processes, attach the debugger to all instances of the running executable process. Next, set one or more breakpoints in your plug-in code.

Test the plug-in

Run the Microsoft Dynamics CRM application, or other custom application that uses the SDK, and perform whatever action is required to cause the plug-in to execute. For example, if a plug-in is registered for an account creation event, create a new account.

Debug your plug-in code

Make any needed changes to your code so that it performs as you want. If the code is changed, compile the code into an assembly and repeat steps 1 through 4 in this procedure as necessary. However, if you change the plug-in assembly's major or minor version numbers, you must unregister the earlier version of the assembly and register the new version.

Register the plug-in in the database

After the edit/compile/deploy/test/debug cycle for your plug-in has been completed, unregister the (on-disk) plug-in assembly and then reregister the plug-in in the Microsoft Dynamics CRM database.

Debugging in the sandbox

It is important to perform these steps before the first execution of a sandboxed plug-in. If the plug-in has already been executed, either change the code of the assembly, causing the hash of the assembly to change on the server, or restart the Microsoft Dynamics CRM Sandbox Processing Service on the sandbox server.

Configure the Server

The sandbox host process monitors the sandbox worker process which is executing the plug-in. The host process checks if the plug-in stops responding, if it is exceeding memory thresholds, and more. If the worker process doesn't respond for than 30 seconds, it will be shutdown. In order to debug a

sandbox plug-in, you must disable this shutdown feature. To disable the shutdown feature, set the following registry key to 1 (DWORD):

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSCRM\SandboxDebugPlugins
```

Register and deploy the plug-in assembly

Register the plug-in in the sandbox (isolation mode) and deploy it to the Microsoft Dynamics CRM server database.

Copy the symbol file (.pdb) of the compiled plug-in assembly to the server\bin\assembly folder on the server running the sandbox worker process named Microsoft.Crm.Sandbox.WorkerProcess.exe. This is the server hosting the Sandbox Processing Service role.

Follow the instructions in steps above to debug the plugin.

Error Handling

For synchronous plug-ins, the Microsoft Dynamics CRM platform handles exceptions passed back to the platform by displaying an error message in a dialog of the web application user interface. For asynchronous plug-ins, the exception message is written to a System Job (AsyncOperation) record.

For plug-ins not registered in the sandbox, the exception message (System.Exception.Message) is also written to the Application event log on the server that runs the plug-in. The event log can be viewed by using the Event Viewer Administrative Tool. Because the Application event log is not available to sandboxed plug-ins, sandboxed plug-ins should use tracing.

Optionally, the plug-in can display a custom error message in the web application by having it throw an InvalidPluginExecutionException exception with the custom message as the Message property value.

Logging and tracing

An alternative method to debug a plug-in is to use tracing. Tracing assists developers in troubleshooting plug-ins by providing run-time plug-in information as an aid in diagnosing the cause of plug-in failure. Tracing is especially useful to debug Microsoft Dynamics CRM Online registered plug-ins as it is the only supported debugging method for that scenario.

The tracing discussed here is different from ASP.NET tracing. Tracing is implemented in the Microsoft Dynamics CRM SDK through the use of the tracing service ITracingService. Developers add Trace statements to their plug-in code, then build and deploy the plug-in. During execution and only when that plug-in passes an exception back to the platform at run-time, tracing information is displayed to the user. For a synchronous registered plug-in, the tracing information is displayed in a dialog box of the Microsoft Dynamics CRM web application. For an asynchronous registered plug-in, the tracing information is shown in the Details area of the System Job form in the web application. The amount and nature of this information is up to you as a developer to code into your plug-ins.

The main reason for implementing this type of tracing is to support the isolated (sandboxed) plug-in and custom workflow activities capability in Microsoft Dynamics CRM. Sandboxed custom code is

not able to write information to the system event log or the file system. The tracing service was implemented to provide sandboxed plug-ins and custom workflow activities with a means to output run-time information when an exception is thrown. In addition, tracing is also supported in plug-ins that are not sandboxed.

An important design feature of tracing is that the tracing information is only made available when an exception is passed from the plug-in or custom workflow activity back to the platform. When the error dialog box is displayed in the web application, the user must click the Download Log File button to view the log containing exception and trace output. If an exception is not thrown or is caught by the plug-in/custom activity code, any tracing information generated by your custom code is lost.

An alternate approach is to create a custom entity to track any run-time plug-in information that you want to record. When the plug-in executes, have the plug-in store its exception information in a record of the custom entity. In this way, you are logging your run-time plug-in information to the Microsoft Dynamics CRM database. This method works for any plug-in regardless of how it is registered. However, if the plug-in executes within the database transaction and an exception occurs that causes a transaction rollback, any entity data changes by the plug-in will be undone.

Lesson 5-7 Custom Workflow Activities

Overview

A developer can create custom code which can accept information from a workflow or dialog, process it, and send information back to it.

Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online supports the registration and execution of custom workflow activities in addition to the out-of-box activities provided by Windows Workflow Foundation. Windows Workflow Foundation includes an activity library that provides activities for control flow, sending and receiving messages, doing work in parallel, and more. However, to build applications that satisfy your business needs, you may need activities that perform tasks specific to that application. To make this possible, Windows Workflow Foundation supports the creation of custom workflow activities.

You can write custom workflow activities in Microsoft Visual C# or Microsoft Visual Basic .NET code by creating an assembly that contains one or more classes derived from the Windows Workflow Foundation `CodeActivity` class. This assembly is annotated with .NET attributes to provide the metadata that Microsoft Dynamics CRM uses at runtime to link your code to the workflow engine.

Custom workflow activities are supported for workflows and dialogs (processes) when built using Windows Workflow Foundation 4. Custom workflow activities created by using Windows Workflow Foundation 3.5 can only be used with workflows (not dialogs) in Microsoft Dynamics CRM 2013 (on-premises and IFD), and only when not registered in the sandbox (isolation mode).

If you want to use your custom workflow activities with both workflows and dialogs, you must use Windows Workflow Foundation 4 to create the custom workflow activities, or update your older custom activities code to work with Windows Workflow Foundation 4. In addition, to register

custom workflow activities in the sandbox (partial trust), your custom workflow activities must be built using Microsoft .NET Framework 4 PU3 (platform update 3) or .NET 4.5.

CodeActivity

To create a custom workflow activity, create a class that inherits from the CodeActivity workflow base class. This class is available in the System.Activities namespace. You will need to add a reference to System.Activities in your project.

Activities that inherit from the CodeActivity class can override the Execute method to produce custom functionality.

Context

As with Plugins, the execution context is available that contains a wealth of information that describes the run-time environment that the custom assembly is executing in.

Assemblies Required

Custom workflow assemblies require the Microsoft.Xrm.Sdk.Workflow.dll assembly reference as well as the Microsoft.Xrm.Sdk.dll and Microsoft.Crm.Sdk.Proxy.dll assembly references to be added your project. These assemblies can be found in the SDK\Bin folder of the SDK.

You will also need to add a reference to System.Activities.

Parameters

A custom workflow assembly can include input and output parameters that are accepted by the code, processed, and then passed as a value back to the workflow. This allows the workflow or dialog to use that returned output within the rest of the workflow being processed.

Data Types

The Microsoft Dynamics CRM types are found in the Microsoft.Xrm.Sdk namespace. Use the InputAttribute and OutputAttribute classes to specify input and output properties.

The following types are supported for custom workflow activities:

- Bool
- DateTime
- Decimal
- Double
- EntityReference
- Int
- Money
- OptionSetValue
- String

Apart from the Input, Output, and Default attributes, some of the supported Microsoft Dynamics CRM types in the custom workflow activities require you to specify additional attributes such as ReferenceTarget and AttributeTarget.

Input Parameter

The following sample shows how to add the input attribute to a Money parameter used in a custom workflow activity. It also shows how to specify a default value for the property.

```
[Input("Money input")]
[Default("232.3")]
public InArgument<Money> MoneyParameter { get; set; }
```

Output Parameter

The following sample shows how to add the output attribute to a Money parameter used in a custom workflow activity. It also shows how to specify a default value for the property.

```
[Output("Money output")]
public OutArgument<Money> MoneyParameter { get; set; }
```

Input and Output Parameter

The following sample shows how to add the input and output attributes to a Money parameter used in a custom workflow activity. It also shows how to specify a default value for the property.

```
[Input("Money input")]
[Output("Money output")]
[Default("232.3")]
public InOutArgument<Money> MoneyParameter { get; set; }
```

DefaultAttribute

You can use the DefaultAttribute class to specify a default value for an input parameter. The following examples show how to set the default value for different types using the Default attribute.

```
[Input("Bool input")]
[Default("True")]
public InArgument<bool> Bool { get; set; }
```

and

```
[Input("Int input")]
[Default("2322")]
public InArgument<int> Int { get; set; }
```

and for an OptionSet

ReferenceTargetAttribute

The EntityReference attribute type requires you to specify the entity type being referenced using the ReferenceTargetAttribute class. The following sample shows how to add the input attribute to an AccountReference parameter in a custom workflow activity by using the ReferenceTarget attribute.

```
[Input("EntityReference input")]
[ReferenceTarget("account")]
[Default("3B036E3E-94F9-DE11-B508-00155DBA2902", "account")]
public InArgument<EntityReference> AccountReference { get; set; }
```

AttributeTargetAttribute

The OptionSetValue attribute type requires you to specify the entity and the attribute being referenced using the AttributeTargetAttribute class. The following sample shows how to add the input and output attributes to an OptionSetValue parameter in a custom workflow activity by using the AttributeTarget attribute.

```
[Input("OptionSetValue input")]
[AttributeTarget("account", "industrycode")]
[Default("3")]
public InArgument<OptionSetValue> OptionSetValue { get; set; }
```

RequiredArgumentAttribute

You can use the System.Activities.RequiredArgumentAttribute class to specify that an input parameter is required.

```
[Input("Update Next Birthdate for")]
[RequiredArgument]
[ReferenceTarget("contact")]
public InArgument<EntityReference> Contact { get; set; }
```

Deploying

After you have created an assembly that contains one or more custom workflow activities, you register this assembly with Microsoft Dynamics CRM. This process is similar to registering a plug-in. The custom workflow activity can then be incorporated into a workflow or dialog in the Process form in Microsoft Dynamics CRM.

Debugging

To debug a custom workflow activity, copy the .pdb file for the assembly to the %installldir%\server\bin\assembly folder.

The assembly can be deployed as on-disk or stored in the database. The recommended deployment is in the database, but for debugging you should choose on-disk.

Next, attach the debugger to the CrmAsyncService.exe process.

Make sure that you remove the .pdb file when you have finished debugging because it uses memory to have it loaded.

Lesson 5-8 Declarative Workflows

Overview

Microsoft Dynamics CRM 2013 on-premise deployments let users create custom XAML workflows. Custom XAML workflows, also called declarative workflows, allow developers the ability to harness the power of Windows Workflow Foundation to create reusable workflows for Microsoft Dynamics CRM.

Using the Microsoft Visual Studio Workflow Designer, workflows can be created or modified by dragging workflow activities from the toolbox onto the design surface, create variables, and set properties of these activities to implement the workflow's functionality. Workflow creators can use built-in Windows Workflow Foundation activities or the process activities that are specific to Microsoft Dynamics CRM.

Additionally, through declarative workflows, logic such as while loops can be incorporated into a Microsoft Dynamics CRM workflow logic flow where the native Microsoft Dynamics Workflows do not enable designers to do this through the interface.

Prerequisites for working with custom XAML workflows

You must use Microsoft Visual Studio to develop custom XAML workflows for Microsoft Dynamics CRM.

To work with the XAML workflows that are created or modified outside of Microsoft Dynamics CRM, make sure that:

Your user account has the Deployment Administrator privilege in Microsoft Dynamics CRM.

Declarative workflows are enabled on the Microsoft Dynamics CRM server. By default they are not enabled. You can use Windows PowerShell to enable or disable XAML workflows.

Enable XAML workflows

Open a Windows PowerShell command window.

```
Add-PSSnapin Microsoft.Crm.PowerShell
$setting = get-crmsetting customcodesettings
$setting.AllowDeclarativeWorkflows="True"
set-crmsetting $setting
```

Disable XAML workflows

Open a Windows PowerShell command window.

```
Add-PSSnapin Microsoft.Crm.PowerShell
$setting = get-crmsetting customcodesettings
$setting.AllowDeclarativeWorkflows=0
set-crmsetting $setting
```

Using the Visual Studio Workflow Designer

Using the Visual Studio Workflow Designer, you can do the following:

- Visually create workflows without having to write code. You can create a workflow project in Visual Studio by using the built-in Visual Studio Activity Library template, and then use a Workflow activity as the root activity.
- Add the Microsoft Dynamics CRM workflow activities in the toolbox.
- After a workflow project is created, you can add activities to the root activity.
- Move activities in a workflow. Some activities, for example, the Sequence activity, can contain multiple child activities. You can create variables to hold values and references that are important to your workflow by using the variables designer.
- Modify pre-existing workflows. In this case, workflows must first be exported from Microsoft Dynamics CRM, and can be modified in Visual Studio before being imported back into Microsoft Dynamics CRM.

Lesson 5-9 Outlook and Plugins

Offline working

Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access enables you to continue your work when you are disconnected from the server.

The event and plug-in infrastructure lets you leverage development investments across solutions by using the same APIs and programming model. The `IOrganizationService` methods and the Microsoft Dynamics CRM OData service methods function both online and offline. When using a method such as Create or Update offline, the data is written locally and then when the user connects to the server, the actions are played back to the server.

To find out whether a message is supported offline, see `Microsoft.Crm.Sdk.Messages` in the SDK. You can also determine whether an `IOrganizationService` message works offline by checking the `SdkMessage.Availability` attribute for the desired message. If the message works for multiple entity types, you must also check the `SdkMessageFilter.Availability` attribute to see whether the message is available offline for the entity you want to work with. For example, the Create message is available offline, but not for the queue, user, or site entities.

Tracing can be enabled on the Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access for debugging.

Deployment

Deployment of plugins to Outlook clients is handled automatically by the Go Offline process.

CrmOutlookService

Microsoft Dynamics CRM contains the assembly `Microsoft.Crm.Outlook.Sdk`, which you can use when customizing Microsoft Outlook. This includes methods to determine which client is being used and whether the user is online or offline.

The Microsoft.Crm.Outlook.Sdk assembly provides programmatic support for basic Microsoft Outlook actions such as synchronization, going offline or online, and CRM for Outlook state verification.

Methods

The following table lists the methods that are available in CrmOutlookService:

Method	Description
GoOffline	Takes Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access into an offline state and triggers a synchronization of the offline database with the online server. This method cannot be called from a plug-in.
GoOnline	Synchronizes Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access with the online Microsoft Dynamics CRM server. This method cannot be called from a plug-in.
SetOffline	Sets Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access into an offline state without triggering a synchronization of the offline database with the online server.
Sync	Triggers a synchronization event between Microsoft Outlook and the Microsoft Dynamics CRM server

Properties

The following table lists the properties available in CrmOutlookService:

Property	Description
IsCrmClientLoaded	Gets a value that indicates whether CRM for Outlook is loaded by Microsoft Outlook.
IsCrmClientOffline	Gets a value that indicates whether Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access is offline.
IsCrmDesktopClient	Gets a value that indicates whether CRM for Outlook is installed.
ServerUri	Gets the server URI to use to connect to the Microsoft Dynamics CRM server based on the state of CRM for Outlook.
State	Gets the state of Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access.

Lesson 5-10 Azure

Overview

Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online supports integration with Windows Azure. By integrating Microsoft Dynamics CRM with Windows Azure, developers can register plug-ins with Microsoft Dynamics CRM that can pass run-time message data, known as the execution context, to one or more Windows Azure solutions in the cloud. This is especially important for Microsoft Dynamics CRM Online because Windows Azure is one of two supported solutions for communicating run-time context obtained in a plug-in to external line of business (LOB) applications. The other solution is the external custom endpoint access capability from a plug-in registered in the sandbox.

The Windows Azure Service Bus combined with the Windows Azure Access Control Service (ACS) provides a secure communication channel for Microsoft Dynamics CRM run-time data to external line of business applications. This capability is especially useful in keeping disparate Microsoft Dynamics CRM systems or other Microsoft Dynamics CRM servers synchronized with Microsoft Dynamics CRM business data changes.

Integration

Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online can be integrated with the Windows Azure platform by coupling the Microsoft Dynamics CRM event execution pipeline to the Windows Azure Service Bus in the cloud. In essence, the Microsoft Dynamics CRM pipeline connects to the Windows Azure Service Bus enabling the data that has been processed as part of the current Microsoft Dynamics CRM operation to be posted to the bus. Windows Azure Service Bus solutions that are “CRM aware” can listen for and read the data that is posted on the service bus by Microsoft Dynamics CRM. The posted data is stored in a RemoteExecutionContext class instance that is an extended version of IExecutionContext passed at run time to Microsoft Dynamics CRM asynchronous plug-ins.

This integration between Microsoft Dynamics CRM and the Windows Azure platform provides a secure channel for communicating Microsoft Dynamics CRM run-time data to external cloud-based line-of-business applications.

Key elements of the integration

The key elements that implement the integration between Microsoft Dynamics CRM and the Windows Azure Service Bus are as follows.

Asynchronous Service

The asynchronous service is responsible for posting the Microsoft Dynamics CRM remote execution context to the Windows Azure Service Bus. Each post is performed by a system job of the asynchronous service. A user can view the status of each system job using the Microsoft Dynamics CRM web application.

Plug-ins

There are two kinds of asynchronous registered plug-ins supported by the integration feature: out-of-box (OOB), and custom.

An Azure aware plug-in is provided with Microsoft Dynamics CRM. This OOB plug-in executes in full trust with the Microsoft Dynamics CRM platform. When registered with Microsoft Dynamics CRM, the plug-in can notify the asynchronous service to post the current request's context to the Windows Azure Service Bus. A developer needs to register a step on this plug-in that identifies the target message and entity in order to enable the service bus posting functionality.

You can also write your own custom plug-in that is "Windows Azure aware". The custom plug-in executes in partial trust mode in the sandbox and can call any Microsoft Dynamics CRM SDK methods. A custom plug-in can initiate posting of the Microsoft Dynamics CRM context to the service bus by including some standard lines of code that notifies the asynchronous service to post the request context. This cloud-specific code makes the plug-ins "Windows Azure aware".

Custom Workflow Activities

Custom workflow activities can be written to post the current request's data context to the Windows Azure Service Bus.

Windows Azure Service Bus

The service bus relays the remote execution context between Microsoft Dynamics CRM and Windows Azure Service Bus solution listeners. The Windows Azure Access Control Service (ACS) manages claims based authentication security.

Windows Azure Solution

For the CRM-Azure integration feature to work there must be at least one solution in a Windows Azure Service Bus solution account, where the solution contains one or more service endpoints. For a relay endpoint contract, a listener that is "CRM aware" must be actively listening on the endpoint for the Microsoft Dynamics CRM request on the service bus. For a queue endpoint contract, a listener does not have to be actively listening. A listener is made "CRM aware" by linking it to the Microsoft.Xrm.Sdk assembly so that type RemoteExecutionContext is defined.

The solution rules must be configured to allow the Microsoft Dynamics CRM remote execution context to be posted to the service bus. To enable this posting, ACS needs to recognize the Microsoft Dynamics CRM deployment as a supported issuer.

Basic Microsoft Dynamics CRM to Azure service bus scenario

As a pre-requisite, ACS has been configured to recognize Microsoft Dynamics CRM as the supported issuer and the Windows Azure Service Bus solution configured with rules to allow Microsoft Dynamics CRM to post to the endpoint on which the listener is listening.

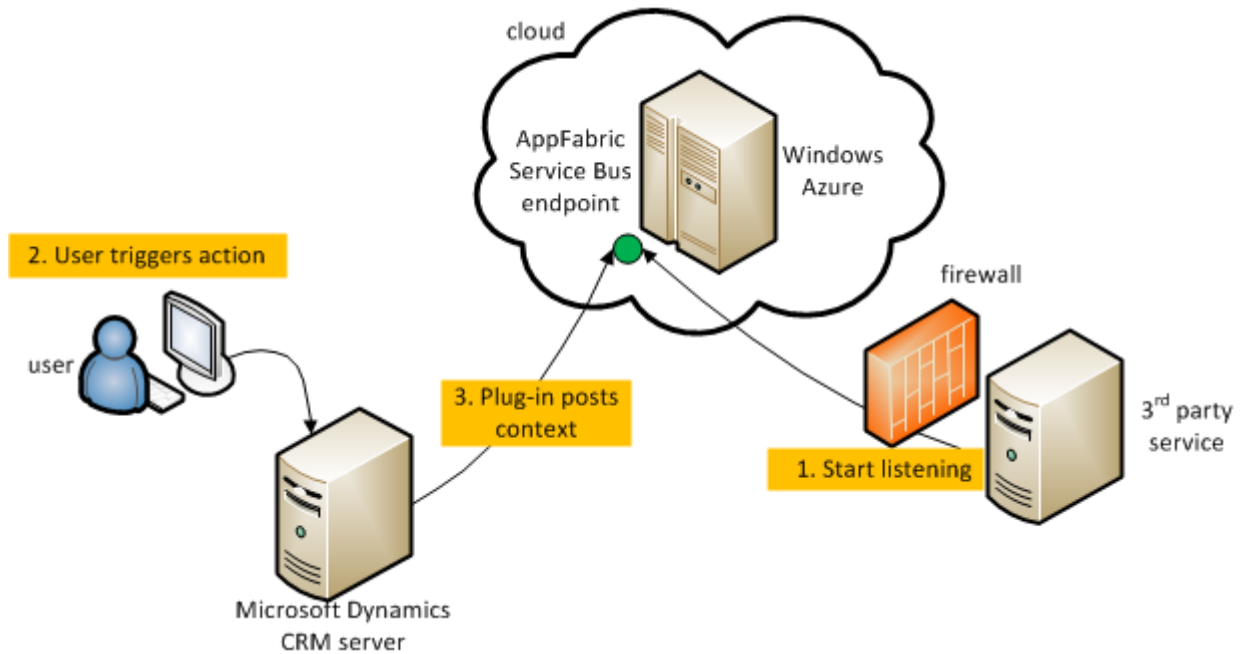


Figure 15 - Azure Service Bus Integration

The sequence of events as identified in this diagram are as follows:

1. A listener is registered on a Windows Azure Service Bus solution endpoint and begins actively listening for the Microsoft Dynamics CRM remote execution context on the service bus.
2. A user performs some operation in Microsoft Dynamics CRM that triggers execution of the registered OOB plug-in or a custom Windows Azure aware plug-in. The plug-in initiates a post, through an asynchronous service system job, of the current request context to the service bus.
3. ACS authenticates the claims posted by Microsoft Dynamics CRM. The service bus then relays the remote execution context to the listener. The listener processes the context information and performs some business related task with that information. The asynchronous service is notified, by the service bus, of a successful post and sets the related system job to a completed status

Contracts

For each solution endpoint, you configure a contract that defines the handling of these remote execution context “messages” on the service bus and the security that should be used on that endpoint. Service bus messages are received at an endpoint using one of the supported contracts listed below.

Queue

A queue contract provides a message queue in the cloud. With a queue contract, a listener does not have to be actively listening for messages on the endpoint. For queues, there is a destructive read and a non-destructive read. A destructive read reads an available message from the queue and the message is removed. A non-destructive read does not remove a message from the queue.

There are two types of queues supported in Microsoft Dynamics CRM: a message buffer queue, and a persistent queue. For message buffer queues, messages in the queue are automatically deleted if not read within a pre-configured length of time that typically is less than 10 minutes. Persistent queues have a much longer message availability duration that can be specified in code.

One-way

A one-way contract requires an active listener. If there is no active listener on an endpoint, the Microsoft Dynamics CRM post to the service bus fails. Microsoft Dynamics CRM will retry the post in exponentially larger and larger time spans until the asynchronous system job that is posting the request is eventually aborted and its status is set to Failed.

Two-way

A two-way contract is similar to a one-way contract except that a string value can be returned from the listener to Microsoft Dynamics CRM.

REST

A REST contract is similar to a two-way contract except it is on a REST endpoint.

Topic

Similar to a queue except that one or more listeners can subscribe to receive messages from the topic.

Listeners

A listener is structured around what is known as ABC: address, binding, and contract. The following information identifies the ABCs of a one-way listener.

- Address: service URI
- Binding: WS2007HttpRelayBinding
- Contract: IServiceEndpointPlugin

After your listener is registered with an endpoint, the listener's Execute method is invoked whenever a message is posted to the service bus by Microsoft Dynamics CRM. The Execute method does not return any data from the method call.

A two-way listener is coded in a similar fashion as a one-way listener. The ABCs of a two-way listener are as follows:

- Address: service URI
- Binding: WS2007HttpRelayBinding
- Contract: ITwoWayServiceEndpointPlugin

For this two-way contract, the Execute method returns a string from the method call.

A REST listener is coded in a similar fashion as a two-way listener. The ABCs of a REST listener are as follows:

- Address: service URI
- Binding: WebHttpRelayBinding
- Contract: IWebHttpServiceEndpointPlugin

For the REST contract, the Execute method returns a string from the method call.

Out of the Box Azure Plugin

An internal plug-in named ServiceBusPlugin is provided with Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online. The plug-in contains the business logic to post the Microsoft Dynamics CRM message execution context to the Windows Azure Service Bus.

To use this plug-in, you need to register a Windows Azure Service Bus solution endpoint and a step for the plug-in using the Plugin Registration Tool. The step defines what message and entity combination being processed by the core Microsoft Dynamics CRM operation should trigger the plug-in to execute. Only the execution context is posted to Azure.

Note: The ServiceBusPlugin can only be registered to run asynchronously.

You can write a custom plug-in that includes the required lines of code to post to the service bus. The plug-in is registered in a similar way, except that it must be registered in the sandbox and run under partial trust.

You can also write a custom workflow activity that can post the execution context to the service bus and include this activity in your workflows.

Lesson 5-11 Plugins vs Custom Workflow Activities

Determining When to Use Plug-ins

Generally, if developing custom code and assemblies, working with plug-ins is more powerful than developing a whole solution in workflow. This is for the following reasons:

- All code is in C# or Visual Basic .NET.
- Plug-ins are not bound by the features and logic patterns that the workflow editor supports.
- It is possible to step through and debug all aspects of the code. It is not possible to step through a process, although you can step through custom workflow activities.

When faced with a situation where processes or plug-ins can be used, consider the guidelines that are shown here:

Ease of development

Best Choice: Depends whether custom workflow activities are required.

Processes: For simple operations that are supported by processes and do not have custom workflow activities, processes are the easiest.

Plug-ins: For advanced operations or operations that are not supported by processes, plug-ins could be easier.

Modifications by non-developers

Best Choice: Processes

Processes: Processes have the advantage when the logic is supported by processes natively. Non-developers can use custom workflow activities created by developers to include additional capabilities in the process rules they design.

Plug-ins: Plug-ins usually require a developer to apply changes. However a plug-in can reference an external file, registry setting or database so that a non-developer can modify the behaviour of the plug-in.

On Demand Application

Best Choice: Processes

Processes: Processes can be applied manually from the Microsoft Dynamics CRM 2013 application.

Plug-ins: Plug-ins do not apply here. Web resources and code can be developed if opting not to use workflow. These pages can be integrated in the Microsoft Dynamics CRM 2013 application through custom buttons, menu commands or IFrames.

Logic Support

Best Choice: Plug-ins.

Processes: Limited to what the editor supports unless using a custom workflow activity.

Plug-ins: This is limited only by the capabilities of the available development resources.

Immediacy

The requirement to affect data before committing it to the database so that users have instant feedback.

Best Choice: Plug-ins or Client-side code

Processes: Until Dynamics CRM 2013 event based activities were always asynchronous and users did not see the result of any data changes immediately within Microsoft Dynamics CRM application. However, with the advent of synchronous workflows there are times when workflow is the correct answer even when immediacy is needed.

Plug-ins: Plug-ins allow for data changes to be made that are visible after the form is reloaded following a Save. Pre-event plug-ins can also cancel any submitted changes.

Client-Side code: Most responsive. But because it does not occur at the platform layer, scenarios can arise where users with different interfaces get different application behaviour.

Platform Layer Data Validation

The requirement to perform data validation and return errors when requirements are not met.

Best Choice: Plug-ins

Processes: Workflow activities normally run asynchronously after the event occurs and cannot return errors to the client or cancel the event. Synchronous workflows are now possible in CRM 2013

Plug-ins: In pre-event plug-ins, validation can be performed, the event that fired the plug-in can be cancelled and an error can be returned to the client.

Lesson 5-12 Developer Toolkit

Developer Toolkit

The Developer Toolkit for Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online is included in the SDK package at SDK\Tools\DeveloperToolkit.

The Developer Toolkit is a set of Microsoft Visual Studio 2010 and Microsoft Visual Studio 2012 integration tools focused on accelerating the development of custom code for Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online. The Developer Toolkit supports creation and deployment of plug-ins, custom workflow assemblies, XAML workflows and web resources. A developer can write custom code within Visual Studio and deploy the code to an unmanaged solution on a Microsoft Dynamics CRM server.

With the Developer Toolkit, you can do the following:

- Easily generate strongly typed proxy classes without having to run CrmSvcUtil.exe.
- Get access to entity and option set definitions within Visual Studio.
- Generate plug-in code so you can immediately begin to write code for business logic.
- Edit and register plug-ins without using the Plug-in registration tool.
- Create new web resources or extract existing web resources, add them to your solution, edit them, and deploy changes all within Visual Studio.
- Create and edit workflow and dialog processes from within Visual Studio.
- Create and deploy XAML workflows in Visual Studio.
- Get easy access to security role and field security profile information in Visual Studio

Module 6– Client Side Script

The key objectives of this module are to provide an understanding of the types of client side scripting that can be used in Dynamics CRM.

Objectives

The key objectives of this module are to:

- Understand use of JavaScript in Dynamics CRM
- Understand the Xrm.Page model
- Understand use of Web Resources
- URL Addressable Forms and Parameters
- Debugging

Lesson 6-1 Use of JavaScript

Overview

Microsoft JScript libraries are JavaScript (JScript) web resources that contain functions which can be used to do the following:

- Handle form and field events.
- Perform actions for controls configured in the ribbon.
- Support other functions

Where JavaScript can be in Microsoft Dynamics CRM

You can use JavaScript to perform actions in form scripts, command bar (ribbon) commands, and web resources.

Form scripts

The most common use of JavaScript in Microsoft Dynamics CRM is to add functions as event handlers for entity form events.

Command bar (ribbon) commands

When you customize the Microsoft Dynamics CRM command bar, you can configure commands for controls that you add. These commands contain rules that control whether the control is enabled and what action is performed when the control is used.

Web resources

Microsoft Dynamics CRM provides an organization-owned entity that stores a binary representation of a file that can be accessed by using a URL. This file is called a web resource. There are several types of web resources.

A web resource that represents a JavaScript library is called a JScript web resource. You can use a webpage (HTML) web resource to provide a user interface with JavaScript libraries included just as you would for files on a web server.

Because these files are part of Microsoft Dynamics CRM, users who access them are already authenticated. Therefore, you can use Microsoft Dynamics CRM web services without having to write code to authenticate the user.

Lesson 6-2 Xrm.Page Model

Overview

Microsoft Dynamics CRM 2013 includes an enhanced client scripting model. The scripting model provides developers more flexibility when they work with the form controls and the data associated with them.

The Xrm.Page object provides a hierarchy of objects that can be used to interact with Microsoft Dynamics CRM 2013 forms in the following ways:

- Show and hide user interface elements.
- Support multiple controls for each attribute.
- Access multiple forms for each entity.
- Manipulate form navigation items.
- Get or set attribute values.

Full details are located in the SDK under the Client-Side Programming Reference topic.

Xrm.Page object hierarchy

As shown in the following diagram, Xrm.Page provides a namespace container for three objects described in the following table:

Object	Description
context	Provides methods to retrieve information specific to an organization, a user, or parameters that were passed to the form in a query string.
data	Provides access to the entity data and methods to manage the data in the form.
ui	Contains methods to retrieve information about the user interface, in addition to collections for several sub components of the form.

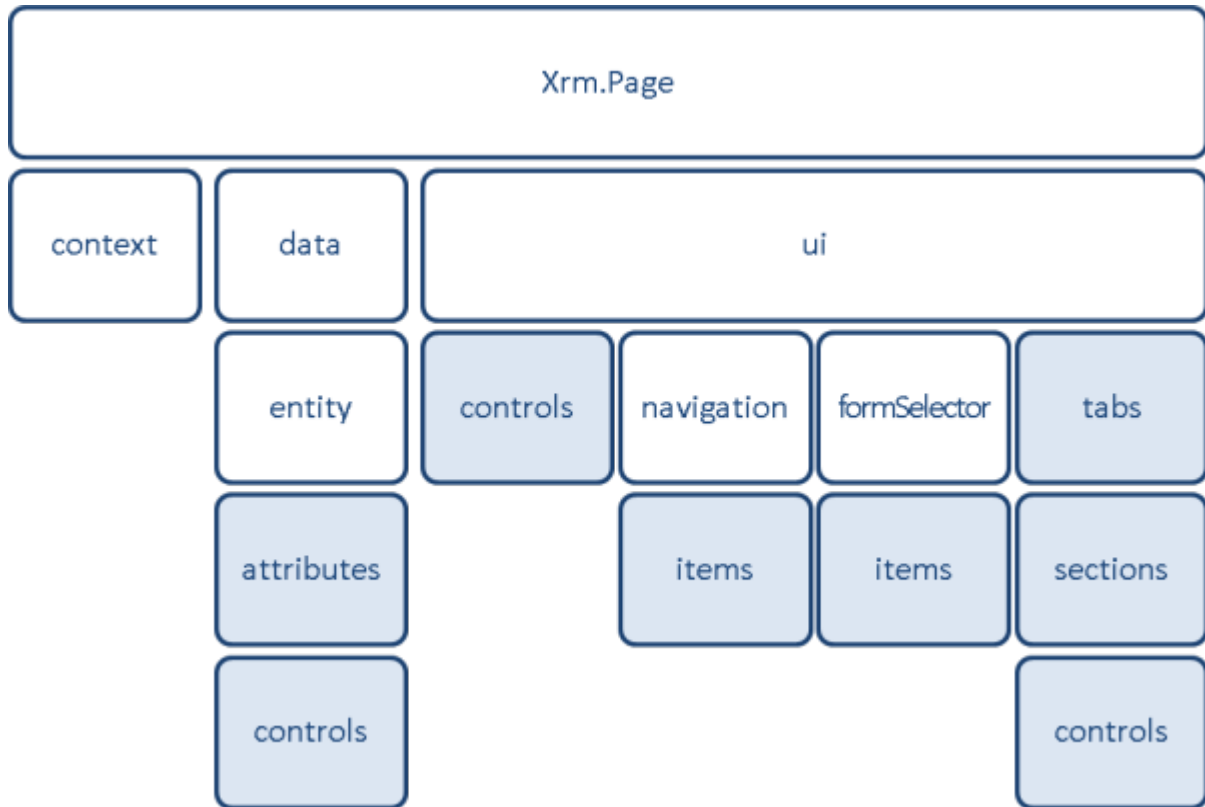


Figure 16 - Xrm.Page model

Xrm.Page.context

Xrm.Page.context provides methods that are used to retrieve information that is specific to an organization, a user, or parameters that were passed to the form in a query string.

Context Properties and Methods

The context object is available in forms by referencing Xrm.Page.context. For web resources executed outside of a form, use the GetGlobalContext function to retrieve a context object.

Properties of the context are:

client - Provides access to the getClient and getClientState methods you can use to determine which client is being used and whether the client is connected to the server.

getClientUrl - Returns the base URL that was used to access the application.

getCurrentTheme - Returns a string representing the current Microsoft Office Outlook theme chosen by the user.

getOrgLcid - Returns the LCID value that represents the base language for the organization.

getOrgUniqueName - Returns the unique text value of the organization's name.

getQueryStringParameters - Returns a dictionary object of key value pairs that represent the query string arguments that were passed to the page.

getUserId - Returns the GUID of the SystemUser

getUserLcid - Returns the LCID value that represents the provisioned language that the user selected as their preferred language.

getUserName - Returns the name of the current user.

getUserRoles - Returns an array of strings that represent the GUID values of each of the security roles that the user is associated with or any teams that the user is associated with.

isOutlookClient - Deprecated. Use client.getClient instead.

isOutlookOnline - Deprecated. Use client.getClientState instead.

prependOrgName - Prepends the organization name to the specified path.

Removed Methods

The getAuthenticationHeader and getServerUrl methods were deprecated with Microsoft Dynamics CRM 2011 and are no longer present in Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online.

Xrm.Page.data

Properties and Methods

Xrm.Page.data provides methods to work with the form. You can refresh the data in the form and save the form asynchronously:

refresh - Asynchronously refreshes and optionally saves all the data of the form without reloading the page.

save - Saves the record asynchronously with the option to set callback functions to be executed after the save operation is completed.

Xrm.Page.data.entity

Xrm.Page.data provides an entity object that provides collections and methods to manage data within the entity form.

Properties and Methods

attributes - The collection of attributes for the entity.

getDataXml - Returns a string representing the xml that will be sent to the server when the record is saved.

getEntityName - Returns a string representing the logical name of the entity for the record.

getId - Returns a string representing the GUID id value for the record.

`getIsDirty` - Returns a Boolean value that indicates if any fields in the form have been modified.

`OnSave` - Use the `addOnSave` and `removeOnSave` methods to add or remove event handlers to the save event

`getPrimaryAttributeValue` - Gets a string for the value of the primary attribute of the entity.

`save` - Saves the record with the options to close the form or open a new form after the save is completed. `save` has an optional argument:

`save()` - If no parameter is included the record will simply be saved. This is the equivalent of using the `Save` command.

`save("saveandclose")` - This is the equivalent of using the `Save and Close` command.

`save("saveandnew")` - This is the equivalent of the using the `Save and New` command.

Xrm.Page.data.entity attribute

Attributes contain the data in the form. Use the `Xrm.Page.data.entity.attributes` collection or the `Xrm.Page.getAttribute` shortcut method to access a collection of attributes

Properties and Methods

Boolean and OptionSet Attribute methods - the `getInitialValue`, `getOption`, `getOptions`, `getSelectedOption`, and `getText` methods provide ways to get information about Boolean or OptionSet attributes.

`Controls` - Access controls associated with attributes.

`getAttributeType` - Get the type of attribute.

`getFormat` - Get the attribute format.

`getIsDirty` - Determine whether the value of an attribute has changed since it was last saved.

`getIsPartyList` - Determine whether a lookup attribute represents a partylist lookup.

`getMaxLength` - Get the maximum length of string which an attribute that stores string data can have.

`getName` - Get the name of the attribute.

`getParent` - Get a reference to the `Xrm.Page.data.entity` object that is the parent to all attributes.

`getUserPrivilege` - Determine what privileges a user has for fields using Field Level Security.

Number Attribute methods - Use the `getMax`, `getMin`, and `getPrecision` methods to access information about the properties of number attributes.

OnChange Event - Use the `addOnChange`, `removeOnChange`, and `fireOnChange` methods and to manage event handlers for the OnChange event.

RequiredLevel - Use the `setRequiredLevel` and `getRequiredLevel` methods to control whether an attribute must have a value in order to save a record.

SubmitMode - Use the `setSubmitMode` and `getSubmitMode` methods to control whether the value of an attribute will be submitted when a record is saved.

Value - Use the `getValue` and `setValue` methods to determine the values set for an attribute and change the value.

Xrm.Page.ui

Xrm.Page.ui provides collections and methods that are used to manage the user interface of the form.

Properties and Methods

`close` - Method to close the form.

`controls` - A collection of all the controls on the page.

`formSelector` - Use the `formSelector.getCurrentItem` method to retrieve information about the form currently in use and the `formSelector.items` collection containing information about all the forms available for the user.

`getCurrentControl` - Method to get the control object that currently has focus on the form.

`getFormType` - Method to get the form context for the record.

`navigation.items` - A collection of all the navigation items on the page.

Form Notification - Use `setFormNotification` to display form level notifications and `clearFormNotification` to remove notifications.

`refreshRibbon` - Method to cause the ribbon to re-evaluate data that controls what is displayed in it.

`Tabs` - A collection of all the tabs on the page.

ViewPort Methods - The ViewPort is the area of the page containing form data. It corresponds to the body of the form and does not include the navigation, header, footer or form assistant areas of the page. Use the `getViewPortHeight` and `getViewPortWidth` methods to get the current size of the ViewPort.

Xrm.Page.ui controls

You access controls using the following collections: `Xrm.Page.ui.controls`, `Xrm.Page.ui.Section.controls`, or `Xrm.Page.data.entity.Attribute.controls`. The `Xrm.Page.getControl` method is a shortcut method to access `Xrm.Page.ui.controls`.

For controls that are bound to attributes it is common to access controls through the `Xrm.Page.data.entity.Attribute.controls` collection

Properties and Methods

Disabled - Detect the state and enable or disable controls using the `getDisabled` and `setDisabled` methods.

`getAttribute` - Get the attribute that the control is bound to.

`getControlType` - Get information about the type of control.

`getName` - Get the name of the control

`getParent` - Get the section object that the control is in.

Label - Get or change the label for a control using the `getLabel` and `setLabel` methods.

Lookup Control Methods and Events - Control the results displayed for a user to choose from when they set the value of a lookup control using the `addCustomFilter`, `addCustomView`, `getDefaultView`, `setDefaultView` methods. You can add or remove event handlers for the `PreSearch` Event using the `addPreSearch` and `removePreSearch` methods.

Notification - Display and remove notifications to users about a control using the `setNotification` and `clearNotification` methods.

OptionSet Control Methods - Modify the options displayed in OptionSet controls using the `addOption`, `clearOptions`, and `removeOption` methods.

Refresh - Refresh the data displayed in a subgrid.

`setFocus` - Set focus on a control.

`setShowTime` - Specify whether a date control should show the time portion of the date.

Visible - Determine which controls are visible and show or hide them using the `setVisible` and `setVisible` methods.

Web Resource and IFRAME Control Methods - Interact with web resource and IFRAME controls using the `getData`, `setData`, `getInitialUrl`, `getObject`, `setSrc` and `getSrc` methods.

Xrm.Page.ui.formSelector

Form Items are available from the `Xrm.Page.ui.formSelector.getCurrentItem` or the `Xrm.Page.ui.formSelector.items` collection. A form item represents a form that is available to a user

because it is associated with a security role that the user is also associated to. Often there will be only one form. When more than one form is available, these methods can be used to change the form the user is viewing.

Properties and Methods

getId - Returns the GUID ID of the form.

getLabel - Returns the label of the form.

Navigate - Opens the specified form.

Xrm.Page.ui.navigation

Each item represents one of the available navigation options available in the navigation bar for entities that have been updated to the new user experience or on the left side of the form for entities that have not been updated.

Properties and Methods

getId - Returns the name of the item.

Label - Get or change the label for a navigation item using the getLabel and setLabel methods.

setFocus - Sets the focus on the item.

Visible - Determine which navigation items are visible and show or hide them using the getVisible and setVisible methods.

Xrm.Page.ui tab

A tab is a group of sections on a page. It contains methods to manipulate tabs as well as access to sections within the tab through the sections collection.

Properties and Methods

DisplayState - Use the getDisplayState and setDisplayState methods to determine whether the tab is collapsed or to collapse and expand the tab.

getName - Method to get the name of the tab.

getParent - Method to return the Xrm.Page.ui (client-side reference) object that is the parent to all tabs.

Label - Use the getLabel and setLabel methods to determine the label for the tab or to hide and show the tab label.

Sections - A collection of sections within the tab.

setFocus - Method to set focus on a tab.

Visible - Use the `setVisible` and `setVisible` methods to determine the tab is visible or to hide and show the tab

Xrm.Page.ui section

A section contains methods to manage how it appears as well as accessing the tab that contains the section.

Properties and Methods

Controls - A collection of controls in the section.

`getName` - Method to return the name of the section.

`getParent` - Method to return the tab containing the section.

Label - Use the `getLabel` and `setLabel` methods to retrieve or change the label for the section.

Visible - Use the `setVisible` and `setVisible` methods to check whether the section is visible or hide and show the section.

Xrm.Utility

The `Xrm.Utility` object provides a container for useful functions not directly related to the current page.

These functions are available in every application page that supports scripting. You can use them in form scripts or in ribbon commands. For HTML web resources, they are available when you include the `ClientGlobalContext.js.aspx` page.

Properties and Methods

Dialogs - Use `alertDialog` and `confirmDialog` display messages to users and set code to execute based on their response. These functions must be used with Microsoft Dynamics CRM for tablets in place of the `window.alert` and `window.confirm` methods.

`isActivityType` - Determine if an entity is an activity entity.

`openEntityForm` - Opens an entity form.

`openWebResource` - Opens an HTML web resource.

Shortcuts

The `Xrm.Page` object includes two shortcut methods that are used to provide direct access to commonly used controls.

Xrm.Page.getAttribute

This shortcut provides direct access to the `Xrm.Page.data.entity.attributes.get` method. The following example uses the `Xrm.Page.getAttribute` shortcut method to get the accountnumber attribute.

```
var accountnumber = Xrm.Page.getAttribute("accountnumber");
```

Xrm.Page.getControl

This shortcut provides direct access to the `Xrm.Page.ui.controls.get` method. The following example uses the `Xrm.Page.getControl` to get the accountnumber control.

```
var accountnumber = Xrm.Page.getControl("accountnumber");
```

Lesson 6-3 Web Resources

Adding custom web content (for example, HTML, Images, Silverlight, JavaScript) to CRM forms is a common method that is used to extend Microsoft Dynamics CRM functionality. Adding custom validation and new applications that interact with Microsoft Dynamics CRM (for example, HTML pages displaying Microsoft Dynamics CRM data) are a few examples that use client-side technologies. Web Resources activates the storage of shared blocks of code or resources that can be reused across the Microsoft Dynamics CRM Web Application.

Web resources represent files that can be used to extend the Microsoft Dynamics CRM web application such as html files, JScript, and Silverlight applications. You can use web resources in form customizations, the SiteMap, or the application ribbon because they can be referenced by using URL syntax.

The URL syntax for web resources allows for relative path references. With your development tools, you can create a group of interdependent files on a development server by using file types compatible with web resources. Then, if you use a consistent naming convention and relative path references, the website will function after you upload all the files into Microsoft Dynamics CRM

Because Web Resources are stored in Microsoft Dynamics CRM and are solution components, they can be easily exported and installed to any Microsoft Dynamics CRM deployment. Web Resources are also available to users of Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access when offline because they are synchronized with the user's data.

Web Resources Types

The available Web Resources are:

File	File extensions	Type
Webpage (HTML)	.htm, .html	1
Style Sheet (CSS)	.css	2

File	File extensions	Type
Script (JScript)	.js	3
Data (XML)	.xml	4
Image (PNG)	.png	5
Image (JPG)	.jpg	6
Image (GIF)	.gif	7
Silverlight (XAP)	.xap	8
Style Sheet (XSL)	.xsl, .xslt	9
Image (ICO)	.ico	10

Where Web Resources can be Used

Web resources can be used in a number of different areas within Microsoft Dynamics CRM 2013. While each web resource has its own distinct use, not all web resources can be used within each functional area of the application. The following table outlines where each web resource can be used within the application.

Resource Type	Forms	Dashboards	SiteMap	Application ribbon	Other
Web Page (HTML)	x	x	x	x	
Script (JScript)	x			x	
Data (XML)					Used in the context of a Web Page (HTML) web resource
Image (PNG)	x	x	x	x	Could be used in the context of a Silverlight (XAP) or Web Page (HTML) web resource
Image (JPG)	x	x	x	x	Could be used in the context of a Silverlight (XAP) or Web Page (HTML) web resource

Resource Type	Forms	Dashboards	SiteMap	Application ribbon	Other
Image (GIF)	x	x	x	x	Could be used in the context of a Silverlight (XAP) or Web Page (HTML) web resource
Silverlight (XAP)	x	x	x	x	Could be used in the context of a Silverlight (XAP) or Web Page (HTML) web resource
StyleSheet (XSL)					Used in the context of a Web Page (HTML) web resource along with a Data (XML) web resource
Image (ICO)	x	x	x	x	Could be used in the context of a Silverlight (XAP) or Web Page (HTML) web resource
Style Sheet (CSS)					Used in the context of a Web Page (HTML) web resource.

Referencing Web Resources

There are several methods that you can use to reference web resources.

\$webresource Directive

You should always use the \$webresource directive when referencing a web resource from a ribbon control or from a SiteMap sub area. Use the \$webresource directive anywhere the XML allows a URL value. The following sample shows how to use it.

```
$webresource:<name of Web Resource>
```

Full URL

The following sample shows the style of URL you can use to view web resources

```
<Microsoft CRM URL>/WebResources/<name of web resource>
```

When you write code to reference a web resource that will need to work for either Microsoft Dynamics CRM Online or on-premises Microsoft Dynamics CRM, you should use the `getClientUrl` function.

Relative URL

When referencing a web resource from areas that do not support using the `$webresource: directive`, a relative URL can be used. To enable this, we recommend that you use a consistent naming convention for the web resources that reflect a virtual file structure. The solution publisher's customization prefix will always be included as a prefix to the name of the web resource. This can represent a virtual "root" folder for all web resources added by that publisher. You can then use the forward slash character (/) to simulate a folder structure that will be honoured by the web server.

From another web resource, you should always use relative URLs to reference each other. For example, for the web page web resource `new_/content/contentpage.htm` to reference the CSS web resource `new_/Styles/styles.css`, create the link as follows:

```
<link rel="stylesheet" type="text/css" href="../../styles/styles.css" />
```

Xrm.Utility.openWebResource

The `Xrm.Utility.openWebResource` function will open an HTML web resource in a new window with parameters to pass the name of the web resource, any query string data to be passed in the `data` parameter, and information about height and width of the window.

Limitations of Web Resources

There is no type of Web Resource that supports the capabilities of an ASP.NET (.aspx) page to execute code on the server. Web Resources are limited to static files or files that are processed in the browser. A Web Resource can contain code that is processed in the browser to execute web service calls to interact with Microsoft Dynamics CRM data.

Web Resources are available only by using the Microsoft Dynamics CRM web application security context. Only licensed Microsoft Dynamics CRM users who have the necessary privileges can access them.

The maximum size of files that can be uploaded is determined by the `Organization.MaxUploadFileSize` property. This property is set in the Email tab of the System Settings in the CRM application. This setting limits the size of files that can be attached to email messages, notes, and web resources. The default setting is 5 MB.

HTML Web Resources

Capabilities of HTML Web Resources

Because an HTML web resource is just streamed to the user's browser, an HTML web resource can include any content that is rendered on the user's browser.

Limitations of HTML Web Resources

An HTML web resource cannot contain any code that must be executed on the server. ASP.NET pages cannot be uploaded as HTML web resources.

HTML web resources can only accept one custom query string parameter called "data".

Passing Parameters to HTML Web Resources

HTML web resource can accept only the parameters in the following table.

Parameter	Name	Description
typename	Entity Name	The name of the entity.
type	Entity Type Code	An integer that uniquely identifies the entity in a specific organization.
id	Object GUID	The GUID that represents a record.
orgname	Organization Name	The unique name of the organization.
userlcid	User Language Code	The language code identifier being used by the current user.
orglcid	Organization Language Code	The language code identifier that represents the base language for the organization.
data	Optional Data Parameter	An optional value that may be passed.
formid	Form Id	The GUID that represents a form id.
pagemode		For internal use only.
security		For internal use only.
tabSet		For internal use only.

Silverlight (XAP) Web Resources

Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online support adding Silverlight 4.0 web resources to entity forms. You can display Silverlight 5.0 web resources within an HTML web resource using a hosting <object> element that is configured for that version

Silverlight support

Microsoft Silverlight web resources remain supported in Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online for backwards compatibility. For components that will be able to be presented on all clients, we recommend using HTML web resources with HTML5 instead of Silverlight.

HTML5 is the preferred client technology for the web, over web plug-ins like Silverlight and Flash. HTML5 can be consumed from any device (PC, tablet, smartphone, and more) and heavily uses JavaScript (and many powerful JavaScript libraries, such as jQuery) and CSS.

Jscript Web Resources

Use Script (JScript) web resources to create a library of JavaScript functions that can be accessed from anywhere.

Capabilities of script web resources

With Script web resources, you can more efficiently manage code used in form scripts, webpage (HTML) web resources, or ribbon commands by linking them to shared library of JavaScript functions.

Limitations of script web resources

Like all web resources, script web resources use the Microsoft Dynamics CRM web application security context. Only licensed Microsoft Dynamics CRM users who have the necessary privileges can access them

Image Web Resources

Use image Web resources to make images available for use in Microsoft Dynamics CRM.

Capabilities of Image Web Resources

With image Web resources you can add images where you need them. Common uses include the following:

- Custom entity icons
- Icons for custom Ribbon controls and SiteMap subareas
- Decorative graphics for entity forms and Web page Web resources.
- Background images that are used by CSS Web resources

Limitations of Image Web Resources

Like all Web resources, image Web resources use the Microsoft Dynamics CRM security context. Only licensed Microsoft Dynamics CRM users who have the necessary privileges can access them.

Only the four specified types of image files (*.jpg, *.png, *.gif, *.ico) are supported.

Stylesheet (XSL) Web Resources

Use Stylesheet (XSL) Web resources to transform XML data.

Data (XML) Web Resources

Use Data (XML) Web resources to save and access data.

Capabilities of XML Web resources

Use XML Web resources to cache data that you want to use in your solution.

Limitations of XML Web resources

Use XML Web resources to cache configuration settings or metadata for your solutions.

An XML Web resource does not represent a robust solution for data that is frequently updated by multiple users. While one user is updating an XML Web resource, another user (or automated process) could update the Web resource and that data would be lost when the first user saves their changes.

Stylesheet (CSS) Web Resources

Use cascading style sheet (CSS) web resources to create style sheets for use in webpage web resources.

Capabilities of CSS web resources

With CSS web resources, you can manage the appearance of webpage web resources by linking them to a shared library of CSS styles.

Limitations of CSS web resources

Like all web resources, CSS web resources are only available in the Microsoft Dynamics CRM security context. Only licensed CRM users who have the necessary privileges can access them.

Lesson 6-4 Forms and Client-Side Events

Overview

Microsoft Dynamics CRM 2013 includes an enhanced client scripting model. The scripting model provides developers more flexibility when they work with the form controls and the data associated with them.

Full details are located in the SDK under the Client-Side Programming Reference topic.

OnLoad Event

The OnLoad event occurs after the form has loaded and therefore it cannot prevent the window from loading. The OnLoad event can be used to apply logic about how the form should be displayed, to set properties on fields, and interact with other page elements. Some actions that can be performed by using the OnLoad event include the following:

- Perform calculations based on changing values
- Alert a user to a situation
- Disable fields that should not be updated

OnSave Event

The OnSave event occurs when a user clicks the Save or Save and Close buttons or other actions that cause the form to be saved, such as the Save method. The event always occurs, even when the data in the form has not changed. The OnSave event does not correspond to the standard HTML OnSubmit event.

A script can detect which action is performed to trigger the form save by using the getSaveMode method in the save event arguments retrieved from the getEventArgs method execution context.

The OnSave event can be canceled to prevent the data from being saved. Therefore it is very common to use the OnSave event to validate data.

OnChange Event

The OnChange event is available on every field. The OnChange event requires two conditions to be true:

- The data in the field must change.
- The field must lose focus.

After the event, the data in the field will be re-validated. This means that the event cannot be used to enter invalid data.

Some actions that can be performed by using the OnChange event include the following:

- Perform calculations to change other fields based on changing values.
- Change the formatting of fields, such as telephone numbers.
- Implement dynamic option sets.

TabStateChange Event

This event occurs when a tab is expanded or collapsed, and enables the execution of code to be deferred until a tab is expanded.

This event is important if a script modifies the src property of an IFRAME control. An IFRAME will be refreshed when the tab is expanded and any changes to the src property will be removed.

Therefore, if code interacts with the src property of an IFRAME, it should always be included in the TabStateChange event instead of the Onload event.

OnReadyStateComplete Event

The OnReadyStateComplete event indicates the content of the IFRAME has loaded and can be accessed in code. Any script that interacts with the contents of an IFRAME will fail unless the contents of the IFRAME has completed loading. This event provides a location to include scripts that will execute after the contents of the IFRAME have completed loading.

Using JavaScript in Forms

To use a function in a JavaScript library, the JavaScript library must first be created as a Jscript web resource.

The Jscript web resource must then be added to the form.

The required function within the Jscript web resource is then specified against the required event.

Limitations

JScript web resources can be associated to, and loaded with an entity form, up to a maximum of 50. After associating a library to the form, it is available to all events in the form. For any event, up to 50 functions can be assigned as event handlers.

Associate Functions at Run Time

Functions can be added to the OnChange attribute and the form OnSave events at run-time by using either of the following methods:

- `Xrm.Page.data.entity attribute.addOnChange`
- `Xrm.Page.data.entity.addOnSave`

When these methods are used, the function is added at the bottom of the event handler pipeline. The corresponding `removeOnChange` and `removeOnSave` methods can be used to remove functions added in this manner.

Form Types

In Microsoft Dynamics CRM, several different form types can be used to view and edit entities. The following table describes the form types available. The forms are accessed by using the `Xrm.Page.ui` object and their form type can be queried with the `Xrm.Page.ui.getFormType()` method. This returns a number that corresponds to the type of form.

Value	Form Type
0	Undefined
1	Create
2	Update
3	Read Only
4	Disabled
5	Quick Create (Deprecated)
6	Bulk Edit
11	Read Optimized (Deprecated)

Note: Quick Create forms return 1. The value 5 was used for an earlier type of quick create form that was removed in CRM 3.0.

Execution context as a Parameter

When you register a function for an event handler you have the option to pass an execution context object as the first parameter to the function.

This object contains methods that allows you to managed variables you wish to share with other event handlers and the save event.

Accessing Context Outside of Form

GetGlobalContext

The GetGlobalContext function returns the same context object found in the Xrm.Page.context.

When you need context information outside a form, include a reference to the ClientGlobalContext.js.aspx page in an HTML web resource.

ClientGlobalContext.js.aspx

You can use the GetGlobalContext function when you include a reference to the ClientGlobalContext.js.aspx page located at the root of the web resources directory.

Lesson 6-5 URL Addressable Forms

Overview

URL addressable elements enable you to include links to Microsoft Dynamics CRM forms, views, dialogs, and reports in other applications. In this manner, you can easily extend other applications, reports, or websites so that users can view information and perform actions without switching applications

URL Addressable Forms and Views

All entity forms and views are displayed in the main.aspx page. Query string parameters passed to this page control what will be displayed. For example:

To open a new account entity record form for on-premises Microsoft Dynamics CRM:

```
http://mycrm/myOrg/main.aspx?etn=account&pagetype=entityrecord
```

To open an account entity record form for Microsoft Dynamics CRM Online where the id is {91330924-802A-4B0D-A900-34FD9D790829}:

```
http://myorg.crm.dynamics.com/main.aspx?etn=account&pagetype=entityrecord&id=%7B91330924-802A-4B0D-A900-34FD9D790829%7D
```

To open the Closed Opportunities view for Microsoft Dynamics CRM Online:

```
http://myorg.crm.dynamics.com/main.aspx?etn=opportunity&pagetype=entitylist&viewid=%7B00000000-0000-0000-00AA-000010003006%7D&viewtype=1039
```

You will typically use the `getClientUrl` method to retrieve the organization root Url for both on-premises Microsoft Dynamics CRM and Microsoft Dynamics CRM Online.

To get the id value for any record, use the Send a Link button the command bar. The following is an example of what will be opened in your email application:

```
<http://mycrm/myOrg/main.aspx?etc=4&id=%7b899D4FCF-F4D3-E011-9D26-00155DBA3819%7d&pagetype=entityrecord>.
```

The id parameter passed to the URL is the encoded id value for the record. In this example the id value is {899D4FCF-F4D3-E011-9D26-00155DBA3819}. The encoded version of the GUID substitutes opening and closing brackets “{” and “}” with “%7B” and “%7D”, respectively

Get the URL for a View

Open the view you want to use.

In the command bar, click Send a Link, and then click Of Current View.

Paste the link into Notepad and edit it to extract only the URL part of the text that you want.

Opening a Dialog Process by Using a URL

A common customization is to enable a user to open a specific dialog process in the context of a specific record. For example, you might want to add a custom button to the ribbon for a specific entity using the id value for current record as an input parameter for the dialog process.

To open a dialog you need the following:

- The unique identifier for the dialog.
- The logical name for the entity the dialog is created for.
- The unique identifier for the record you want to have the dialog run against

Used as [organization url]/cs/dialog/rundialog.aspx?DialogId=[dialog unique identifier]&EntityName=[entity logical name]&ObjectId=[unique identifier for the record]

Form Parameters

By default, Microsoft Dynamics CRM lets a specified set of query string parameters to be passed to a form. You use these parameters to set default values when you create a new record in the application. Each parameter must use a standard naming convention that includes a reference to the attribute logical name.

In your applications, you may want to pass custom query string parameters to an entity form. This topic provides information about how you can define a set of specific parameter names and data types that can be accepted for a specific entity form.

There are two ways to specify which query string parameters will be accepted by the form:

- Edit form properties

- Edit form XML

Query String Parameters

The following are the standard query string parameters for the main.aspx page:

Parameter	Description
etn	<p>The logical name of the entity.</p> <p>Do not use the etc (entity type code) parameter that contains an integer code for the entity. This integer code varies for custom entities in different organizations.</p>
extraqs	<p>Optional for forms. This parameter contains encoded parameters within this parameter.</p> <p>When an entity has more than one form defined, you can use this parameter to specify which form to open by passing the encoded parameter formid with the value equal to the ID value of the form. For example, to open a form with the ID of '6009c1fe-ae99-4a41-a59f-a6f1cf8b9daf', include this value in the extraqs parameter: formid%3D6009c1fe-ae99-4a41-a59f-a6f1cf8b9daf%0D%0A.</p>
pagetype	<p>The type of page. There are two possible values:</p> <p>entityrecord Displays an entity record form.</p> <p>entitylist Displays an entity view.</p>
id	<p>Optional for forms. Use this when you open a specific entity record. Pass in the encoded GUID identifier for the entity. The encoded version of the GUID substitutes opening and closing brackets "{" and "}" with "%7B" and "%7D", respectively, for example {91330924-802A-4B0D-A900-34FD9D790829} is %7B91330924-802A-4B0D-A900-34FD9D790829%7D.</p>
viewid	<p>Required for views. This is the ID of the savedquery or userquery entity record that defines the view. The easiest way to get the URL for a view is to copy it.</p>
viewtype	<p>Defines the view type. Possible values are as follows:</p> <ul style="list-style-type: none"> • 1039 Use for a system view. The viewid represents the Id of a savedquery record. • 4230 Use for a personal view. The viewid represents the Id of a userquery record.

Custom Parameters

When you edit an entity form, on the Home tab in the Form group, click Form Properties. In the Form Properties dialog box, select the Parameters tab. When you add a parameter you just specify its name and data type.

Each name attribute must contain at least one underscore ('_') character. However, the name of the query string parameter cannot begin with an underscore. The name also cannot start with 'crm_'. It is strongly recommend that you use the customization prefix of the solution publisher as the naming convention.

A valid querystringparameter name attribute value would be 'myISV_contact_specialvalue'. If a querystringparameter element name is not unique, it may be overwritten by another parameter definition using a different data type.

Fields as Parameters

You can also pass the schema name of any field on the form as a query string parameter. CRM will automatically populate the CRM fields with the values passed.

Encoding

In URLs certain characters such as & need to be encoded so as to ensure that they don't get misinterpreted as any of the other parts of the URL.

For Query String Parameters, all characters included in the query string must be encoded using the `encodeURIComponent` method.

Common encoded URI components

- %3d (=)
- %26 (&)

Use the JScript `decodeURIComponent` method to decode the values passed.

Handling Form Parameters

When using form level parameters, understand how they can be consumed within the form. This will usually be done on the `OnLoad` event of the form using the `Xrm.Page.context.getQueryStringParameters` method

As part of the client-side context, the `getQueryStringParameters` method enables a developer to capture an array of key value pairs representing the query string arguments that were passed to the page.

Lesson 6-6 Best Practice

Best Practice

The following sections describe best practices when you use JavaScript with Microsoft Dynamics CRM.

Comments

Comment your code.

Define unique names for your JavaScript functions

When you are the only developer for an HTML page you can easily manage the names of the JavaScript functions you use. In Microsoft Dynamics CRM, other solutions may add JavaScript functions to the page where your function is used.

If two JavaScript functions on a page have the same name, the first function defined is overwritten by the second. For this reason, make sure that you define unique names for your JavaScript functions.

Use the prefix for your publisher when naming functions.

Create virtual file structure

Use a consistent naming convention for the Web resources that reflect a virtual file structure.

The solution publisher's prefix will always be included as a prefix to the name of the web resource. This can represent a virtual root folder for all Web resources added by that publisher.

You can then use the backslash character (\) to simulate a folder structure that will be honoured by the Web server.

From another web resource, you can then use relative URLs to reference other web resources e.g.

```
<link rel="stylesheet" type="text/css" href="myisv_/styles/mystyles.css" />
```

Namespaced Library Names

Associate each function with a JScript object to create a kind of namespace to use to call functions, as shown in the following example.

```
// If the MyUniqueName namespace object is not defined, create it.
if (typeof (MyUniqueName) == "undefined")
{
    MyUniqueName = {};
}

// Create Namespace container for functions in this library;
MyUniqueName.MyFunctions = {
    performMyAction: function()
    {
        // Code to perform the action.
        // Call another function in the library
        this.anotherAction();
    },
    anotherAction: function(){
        // Code in another function
    }
};
```

Then when using the function, the full name can be specified as shown in the following example.

```
MyUniqueName.MyFunctions.performMyAction();
```

If calling a function within another function, the "this" keyword can be used as a shortcut to the object that contains both functions. However, if the function is used within another object, such as the definition of a callback function in the same library, the full name should be used.

Avoid using unsupported methods

On the Internet, you can find many examples or suggestions that describe using unsupported methods. These may include leveraging undocumented internal function for page controls. These methods may work but because they are not supported you can't expect that they will continue to work in future versions of Microsoft Dynamics CRM.

Use the Custom Code Validation Tool to identify code that is using unsupported methods.

Use a cross-browser JavaScript library for HTML web resource user interfaces

A cross-browser JavaScript library, such as jQuery, provides many advantages when developing HTML web resources that must support multiple browsers. JavaScript libraries like jQuery provide a unified development experience for all browsers supported by Microsoft Dynamics CRM. These capabilities are appropriate when you are using HTML web resources to provide user interfaces. JavaScript libraries like jQuery provide consistent ways to interact with the Document Object Model (DOM).

Do not use jQuery for form script or commands

We do not recommend or support using jQuery for any pages within the application. This includes form scripts and ribbon commands.

Recognize limitations for content delivery network (CDN) libraries

Content delivery network (CDN) JavaScript libraries provide many advantages for public websites. Because these libraries are hosted on the Internet, you do not need to create web resources that contain the content of the libraries. For Microsoft Dynamics CRM you should consider the following issues before you use a CDN JavaScript library.

Users of the Microsoft Dynamics CRM for Microsoft Office Outlook with Offline Access client have the capability to work with no Internet connection while working offline. If you are depending on an Internet connection for your JavaScript libraries, your code will fail.

Some organizations will restrict Internet access for employees. Unless they configure the network to allow access to the CDN library sites, your code may fail for those organizations.

The alternative to using CDN libraries is to create a script (JavaScript) web resource with the contents of the library. Because web resources are organization-owned entities they will be synchronized when a Microsoft Dynamics CRM for Outlook with Offline Access user goes offline. Because these web resources now become part of the application they will not be blocked if an organization restricts access to the Internet.

Use feature detection when writing functions for multiple browsers

Even when you use a cross-browser library like jQuery, you need to be very aware of differences between browsers. You can generally detect which browser is being used by querying the `navigator.userAgent` property. This is called browser detection. Browser detection is not a good strategy for most cases because it can't take into account what features newer versions of a browser have. Also, some browsers provide the capability to modify the `navigator.userAgent` property so that they appear to be a different browser.

Feature detection is the recommended approach. By detecting what features are available, you can create code paths for the browsers you support without knowing exactly which browser is being used.

Do not access the DOM

JavaScript developers are used to interacting with Document Object Model (DOM) elements in code. You might use the `window.getElementById` method or the jQuery library. You are free to use these techniques in your HTML web resources, but they are not supported to access elements in Microsoft Dynamics CRM application pages or entity forms. Instead, access to entity form elements are exposed through the `Xrm.Page` object model. The Microsoft Dynamics CRM development team reserves the right to change how pages are composed, including the ID values for elements, so using the `Xrm.Page` object model protects your code from changes in how pages are implemented

Use asynchronous data access methods

When you access data by using the Microsoft Dynamics CRM web services that use the REST or SOAP endpoint for web resources, always use an `XMLHttpRequest` that is configured to execute asynchronously. The reason is that the browser operates on a single thread. If that thread is being used to execute a long-running process synchronously the browser will stop responding.

Lesson 6-7 Debugging JavaScript

Overview

Each browser provides some kind of debugging extension. Internet Explorer provides developer tools you can use to debug scripts in Microsoft Dynamics CRM.

You can also use Microsoft Visual Studio.

When you use JavaScript libraries in Microsoft Dynamics CRM, your libraries are added to a page that includes many libraries. It can sometimes be difficult to isolate your specific library in the debugging environment. When using debugging tools in Internet Explorer, on the Script tab, expand the available scripts and find the one with the name that corresponds to the name of your JavaScript web resource.

Debugging tools for different browsers have similar capabilities. Once you have found your library, you can set a break point and recreate the event that should cause your code to run.

Internet Explorer

The Internet Explorer developer tools can be opened by pressing F12 when viewing a page using Internet Explorer.

Before starting to debug client-side scripts, make sure the development settings are set correctly and the latest code from the server is being debugged instead of the cached scripts. The following sections list the recommended development settings for debugging .

In Internet Explorer, set the following options:

- Load page content = "Every time"
- Friendly errors = Off
- Script debugging = On

Enabling the Every Visit to the Page Feature

Perform these steps to enable the Every time I visit the webpage feature:

1. Start Internet Explorer.
2. Select Tools, and then Internet Options.
3. On the General tab in the Browsing history section, click Settings.
4. Enable the Every time I visit the webpage option.
5. Click OK.

Setting Script Debugging and Friendly Errors Options

In Internet Explorer, enable the script debugging option and disable the friendly errors option.

Perform these steps to set these options:

1. Start Internet Explorer.
2. On the Tools menu, click Internet Options.
3. Click the Advanced tab.
4. Clear the Disable script debugging (Internet Explorer) option.
5. Clear the Show friendly HTTP error messages option.
6. Click OK
7. Close Internet Explorer.

Other Browsers

For Google Chrome, press F12 to open developer tools. Firebug is a popular browser extension for web development using Mozilla Firefox. For Apple Safari, you must first choose the Show Develop menu in menu bar in Advanced Preferences. Then you can choose Show Web Inspector from the Develop menu.

Write messages to the console

Using the window.alert method when debugging JavaScript is still a common way to troubleshoot code in the application. But now that all modern browsers provide easy access to debugging tools, it is not a best practice, especially when others might be using the application you are debugging.

Consider writing your messages to the console instead. The following is a small function you can add to your libraries that you can use to send any messages you want to view to the console when it is open.

```
function writeToConsole(message)
{
    if (typeof console != 'undefined') {
        console.log(message);
    }
}
```

Unlike using the alert method, if you forget to remove any code that uses this function, people using the application will not see your messages.

Module 7– Client Side Code

The key objectives of this module are to provide an understanding of the how client side code that can be used in Dynamics CRM.

Objectives

The key objectives of this module are to:

- Understand OData and SOAP
- Understand JSON
- Understand JQuery

Lesson 7-1 Data access using JavaScript

One common need for developers is using JavaScript is to retrieve data for Microsoft Dynamics CRM platform.

Early editions of Microsoft Dynamics CRM used FetchXML to retrieve information from the platform on the client side whereas FetchXML is used frequently it can be challenging sometimes to construct and test these queries.

There are two web services you can use within the application to access data by using JavaScript.

OData (REST) endpoint

Previously known as the 'REST endpoint for web resources'. You can use the OData endpoint to execute HTTP requests by using a web service that is based on a Uniform Resource Identifier (URI). "RESTful" web services are popular because they can make programming easier.

The current implementation of the OData endpoint is limited to create, retrieve, update and delete operations. One of the advantages of the OData endpoint is that it implements the OData protocol, which provides the way to query and update data. When you use JavaScript, you usually retrieve objects in JavaScript Object Notation (JSON) format. This makes the results easier to work with.

The primary limitation of the OData endpoint is that you cannot use the `IOrganizationService.Execute` method to execute messages (Request and Response classes). You use the SOAP endpoint for web resources to execute messages.

SOAP endpoint

The SOAP endpoint lets you execute messages because the REST endpoint does not yet allow this. You can also call the `Create`, `Retrieve`, `Update`, `Delete` and `RetrieveMultiple` methods of this web service, but using it is not as easy as using the REST endpoint.

Which endpoint to use

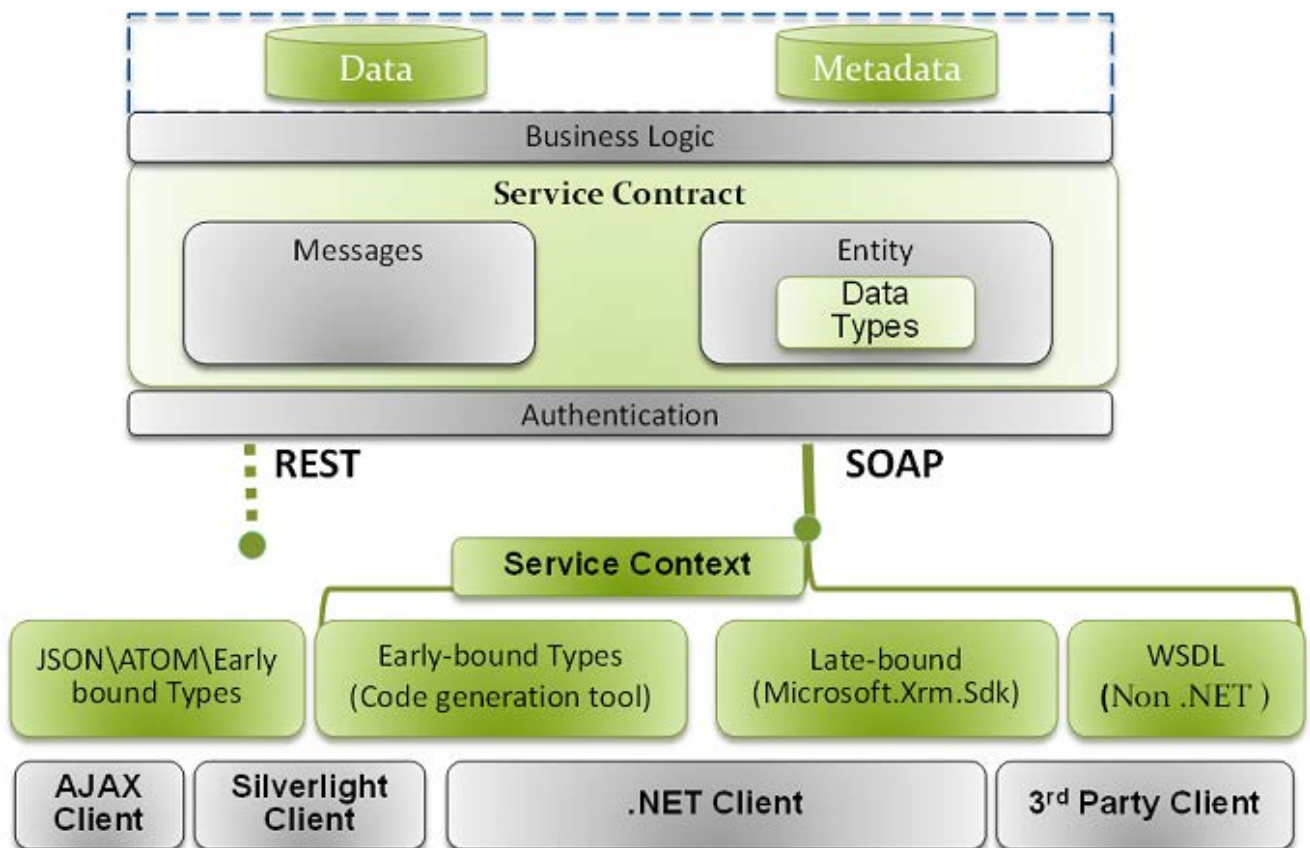


Figure 17 - Data Access Architecture

The REST endpoint provides an alternative to the windows communication foundation SOAP endpoint but there are currently some limitations be aware these limitations are not a limitation of the OData protocol. The limitations of the implementation of the OData protocol within Microsoft Dynamics CRM.

Only create, retrieve, update and delete actions may be performed on entity record messages that require the execute message cannot be performed. Associate and disassociate actions can be performed by using navigation properties.

The following table describes the appropriate web service to use depending on the task you need to perform

Task	Web Service
Create, Retrieve, Update and Delete records.	OData endpoint
Associate and Disassociate records	OData endpoint
Assign Records	Modern app SOAP endpoint
Retrieve Metadata	Modern app SOAP endpoint

Lesson 7-2 Modern app SOAP endpoint

Modern app SOAP endpoint for modern applications

Unlike the REST endpoint for web resources, the SOAP endpoint uses the Organization service. This is the same service used when writing applications that exist outside of the Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online application. The differences are:

- Requests are sent to a different URL: <organization URL>/XRMServices/2011/Organization.svc/web.
- Authentication for web resources is provided by the application.

If you use this endpoint outside the application, you must implement authentication.

The modern app SOAP endpoint provides access to all the messages defined in the Organization service. However, only the types defined in the Web Services Description Language (WSDL) will be returned. There is no strong type support.

While the modern app SOAP endpoint is also capable of performing create, retrieve, update and delete operations, the OData endpoint provides a better developer experience for client application extensions. In this release of Microsoft Dynamics CRM, the modern app SOAP endpoint provides an alternative way to perform operations that the OData endpoint isn't capable of.

Using the SOAP endpoint with JavaScript

With JavaScript, you will be using XMLHttpRequest to POST requests to the service. The body of the request must contain the XML appropriate for the message you are using. You must also parse the XML returned in a response.

With XMLHttpRequest, it is possible to make synchronous requests. However it is highly recommended to always use asynchronous requests. Because manually configuring each request is very time consuming, it is expected that you will reuse existing libraries or create your own.

Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online does not provide a comprehensive set of JavaScript libraries. The specific syntax used when calling JavaScript libraries depends on how they are designed.

The SDK provides a Microsoft Visual C# solution called SoapLogger that lets you capture the XML sent and received when you perform operations with the Microsoft Dynamics CRM web services by using Visual C#. With this information, you can build your own JavaScript libraries. Microsoft CRM MVP Jamie Miley has released a [CRM 2011 Jscript Soap Request Formatter](#) program on CodePlex that takes this a step further and actually generates JavaScript libraries for you.

Lesson 7-3 ODATA

What Is OData?

The OData endpoint uses the Open Data protocol. This protocol implements a 'RESTful' design pattern. REST represents Representational State Transfer. REST is an architectural style in which every resource is addressed by using a unique URI. In Microsoft Dynamics CRM, a resource can be an entity collection or a record.

Microsoft Dynamics CRM 2013 uses the Windows Communication Foundation, or WCF, data services framework to provide an open data protocol endpoint that is a REST-based data service. This endpoint is called the organization data service.

In CRM 2013 OData endpoint has been extended to support external authentication which is a particular import to external clients.

REST

REST represents Representational State Transfer. REST is an architectural style in which every resource is addressed by using a unique URI. In Microsoft Dynamics CRM, a resource can be an entity collection or a record.

REST works the way the Internet works. You interact with resources by using HTTP verbs such as GET, POST, MERGE, and DELETE. Various libraries can be used to process the HTTP requests and responses. REST provides a standard interface that you can use with any programming language. REST allows for either synchronous or asynchronous processing of operations. The capability to perform asynchronous operations makes REST well suited for web resources and scripts used in Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online.

Microsoft Dynamics CRM Implementation of OData

Microsoft Dynamics CRM 2013 uses the Windows Communication Foundation (WCF) Data Services framework to provide an Open Data Protocol (OData) endpoint that is a REST-based data service. This endpoint is called the Organization Data Service. In Microsoft Dynamics CRM, the service root URI is:

[Your Organization Root URL]/xrmservices/2011/organizationdata.svc

This XML document uses conceptual schema definition language (CSDL) to describe the available data. You will download this document and use it to generate typed classes when you use managed code or as a reference for available objects when you use JavaScript.

Limitations

The OData endpoint provides an alternative to the SOAP endpoint, but there are currently some limitations.

Only Create, Retrieve, Update, and Delete actions can be performed on entity records.

- Messages that require the Execute method cannot be performed.

- Associate and disassociate actions can be performed by using navigation properties.

The OData protocol is not fully implemented. Some system query options are not available.

You cannot use late binding with managed code against custom entities, attributes, or relationships that did not exist when the code was written.

You will typically use WCF Data Services Client Data Service classes while programming by using managed code. These classes allow for early binding so that you get strongly typed classes at design time. The only entities available to you are those defined in the system when the classes were generated. This means that you cannot use late binding to work with custom entities, attributes, or relationships that were not included in the WCF Data Services Client Data Service classes when they were generated

OData Entity Data Model (EDM)

To provide a consistent set of URIs that corresponds to the entities used in Microsoft Dynamics CRM, an Entity Data Model (EDM) organizes the data in the form of records of "entity types" and the associations between them.

The Microsoft Dynamics CRM EDM is described in an OData Service Metadata document available at the following path:

[Your Organization Root URL]/xrmservices/2011/organizationdata.svc/\$metadata

This XML document uses conceptual schema definition language (CSDL) to describe the available data. You will download this document and use it to generate typed classes when you use managed code or as a reference for available objects when you use JavaScript.

AJAX

AJAX (Asynchronous JavaScript and XML) is a web development technique used to create interactive web applications. Server requests are made from the browser in the background using an XMLHttpRequest object. While you can send synchronous requests, the recommended practice is to send asynchronous requests. Asynchronous requests require two JScript functions, one to send the request and a second "callback" function to process a response.

When using REST in the context of AJAX, there are a handful of topics to understand, JSON, XMLHttpRequest, and JQuery.

SDK.REST.js

The Sample: Create, retrieve, update, and delete using the OData endpoint with JavaScript includes an SDK.REST.js library that provides an example of a reusable library that can further simplify using the OData endpoint.

Accessing Microsoft Dynamics CRM Entity Data

Each Microsoft Dynamics CRM entity is represented in the conceptual schema definition language (CSDL) as a collection using the <EntitySet> element. The name of each collection follows the naming

convention of [Entity Schema Name]+ Set. This name is used in the URL to access a collection of entity records. A list of all the collections available is listed when you view the service root URI. To create a query you append your criteria to the resource path.

For example, in your browser you can view the ATOM Account entity records (called "entries") using the path in the following example

```
[Your Organization Root URL]/XRMServices/2011/OrganizationData.svc/AccountSet
```

After you view the list of account records, you can see how each of them can be referenced individually using the URL syntax in the following example

```
[Your Organization Root URL]/XRMServices/2011/OrganizationData.svc/AccountSet(guid'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

OData Query options

You can refine the results of your query by using system query options. The following table lists the query string options defined in the OData protocol that are implemented in the OData endpoint for Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online.

Option	Description
\$expand	Directs that related records should be retrieved in the record or collection being retrieved. if you want to retrieve related records, locate the name of the entity relationship that defines this relationship, you might have to view the entity relationship information in the application to correctly identify the relationship or the CDSL, the organization data service.
\$filter	Specifies an expression or function that must evaluate to 'true' for a record to be returned in the collection.
\$orderby	Determines what values are used to order a collection of records. By default the order is ascending. Use DASC to reverse the order and ASC to explicitly set the default
\$select	specifies a subset of properties to return and in the order in which the columns of data should be organized. This default return all columns is expressed as select equals star
\$skip	Sets the number of records to skip before it retrieves records in a collection.
\$top	Determines the maximum number of records to return.

Microsoft Dynamics CRM does not support using the following query system options: Inline count, count, or format. Additionally, date and arithmetic functions are not available to Microsoft Dynamics CRM 2013.

Lesson 7-4 JSON

What is JSON?

JavaScript Object Notation (JSON) format is used for serializing and transmitting structured data much in the same way that XML is typically used. Like XML, it is text based and designed to be readable by humans.

To convert regular JScript objects into the JSON format you use the `JSON.stringify` method. Because the text in JSON defines JScript objects, the text could be converted to JScript objects by using the `eval` method. However, this practice creates security vulnerabilities. You should use the `JSON.parse` method instead.

To use JSON in Microsoft Dynamics CRM, customizers or developers should create a JScript web resource that contains the minified contents of `json2.js`. This library is available at <http://www.json.org/json2.js>.

How is JSON Used in Microsoft Dynamics CRM?

Microsoft Dynamics CRM 2011 uses the Windows Communication Foundation Data Service Framework to provide an Open Data Protocol (OData) end point that is a REST-based data service.

OData sends and receives action, data by using ATOM or JavaScript object notation, JSON.

- ATOM is an XML-based format usually used for RSS feeds.
- JSON is a text format that allows for serialization of JavaScript objects.

If using JSON to use OData in Microsoft Dynamics CRM, customizers or developers should create a JScript Web resource that contains the minified contents of `json2.js`.

If JSON is being used on the form level, add the `json2` web resource to the JScript library on the form. If using it within the context of a web resource such as an HTML (Web Page) web resource, it will need to be added as a JScript web resource and referred to using the relative path.

XmlHttpRequest

Within the context of Microsoft Dynamics CRM 2013, the `XMLHttpRequest` is used when using OData to perform operations against application data. An `XMLHttpRequest` provides instructions to the server about the format of any data to be included in the response. Because the REST endpoint supports both ATOM and JSON formats you have the option to request data to be returned in the XML ATOM format. However, with JScript code the expected typical request will use JSON because it is easily consumable using JScript.

Example

The following sample uses JSON and `XMLHttpRequest` to create a new account record:

```
var account = {};  
    account.Name = "Sample Account";  
    var jsonAccount = JSON.stringify(account);
```

```

var createAccountReq = new XMLHttpRequest();
createAccountReq.open("POST", Xrm.Page.context.getClientUrl() +
"/XRMServices/2011/OrganizationData.svc/AccountSet", true);
createAccountReq.setRequestHeader("Accept", "application/json");
createAccountReq.setRequestHeader("Content-Type", "application/json; charset=utf-8");
createAccountReq.onreadystatechange = function () {
    createAccountReqCallBack(this);
};
createAccountReq.send(jsonAccount);

function createAccountReqCallBack(createAccountReq) {
    if (createAccountReq.readyState == 4 /* complete */) {
        createAccountReq.onreadystatechange = null; //avoids memory leaks
        if (createAccountReq.status == 201) {
            //Success
            var newAccount = JSON.parse(createAccountReq.responseText).d;
        }
        else {
            //Failure
            errorHandler(createAccountReq);
        }
    }
};

```

Lesson 7-5 JQuery

Use of jQuery

Use jQuery with HTML web resources

We recommend that you use jQuery together with HTML web resources to provide user interfaces because it is an excellent cross-browser library.

With HTML web resources, you control the libraries that are present and there is no restriction against manipulating the DOM. Feel free to use jQuery within your HTML Web resources.

Avoid using jQuery with form scripts or ribbon commands

We do not recommend using jQuery in form scripts and ribbon commands.

When to use JQuery

Most of the benefit provided by jQuery is that it allows for easy cross-browser manipulation of the DOM. This is explicitly unsupported within form scripts and ribbon commands. Restrict your scripts to use the Xrm.Page and Xrm.Utility libraries available in form scripts and ribbon commands.

If you decide to use the remaining capabilities of jQuery that are useful with Microsoft Dynamics CRM and include the ability to use \$.ajax, consider the following:

- For best performance, don't load jQuery in the page if you do not need it

- Using \$.ajax to perform requests against the OData and Modern Apps SOAP endpoint is supported, but there are alternatives. The alternative to using \$.ajax is to use the browsers XMLHttpRequest object directly. The jQuery \$.ajax method is just a wrapper for this object. If you use the native XMLHttpRequest object directly, you do not need to load jQuery.
- Consider using SDK.REST.js
- Each version of jQuery that is loaded in a page can be a different version. Different versions of jQuery have different behaviours and these can cause problems when multiple versions of jQuery are loaded on the same page.

Use JQuery within Microsoft Dynamics CRM

To use JQuery within Microsoft Dynamics CRM 2011 forms, the dashboard, or through web resources that are used from other parts of the application, you will need to make a reference to jquery1.5.1.js.

To do this, create a JScript web resource and upload the script file that comes with the Microsoft Dynamics CRM SDK.

The jquery.1.5.1.js is a popular JScript library that includes capabilities to perform asynchronous data operations using the \$.ajax object.

Once added as a web resource the scripts can be added to the form JScript Library (if using JQuery on the form level) or within the context of an HTML (Web Page) resource. When using both JQuery and JSON within a form JScript Library or on an HTML (Web Page) web resource, it is important to reference the jquery.1.5.1.js before the json2.js.

Example

The following sample uses JQUERY to create a new account record:

```
var account = {};
account.Name = "Sample Account";
var jsonAccount = window.JSON.stringify(account);
$.ajax({
    type: "POST",
    contentType: "application/json; charset=utf-8",
    datatype: "json",
    url: Xrm.Page.context.getClientUrl() + "/XRMServices/2011/OrganizationData.svc/AccountSet",
    data: jsonAccount,
    beforeSend: function (XMLHttpRequest) {
        //Specifying this header ensures that the results will be returned as JSON.
        XMLHttpRequest.setRequestHeader("Accept", "application/json");
    },
    success: function (data, textStatus, XMLHttpRequest) {
        account = data.d;
    },
    error: function (XMLHttpRequest, textStatus, errorThrown) {
        errorHandler(XMLHttpRequest, textStatus, errorThrown);
    }
});
```

Module 8 – User Interface

This module explains how to add custom button, menu items and navigation areas so that custom solutions are included in Microsoft Dynamics CRM.

This module also explains how to modify the Application Navigation area in Microsoft Dynamics CRM. The navigation can be reorganized, and areas can be added so that users can navigate in the application.

Objectives

The key objectives of this module are to:

- Understand how to customise the SiteMap
- Understand how to customise the Command Bar

Lesson 8-1 User Interface

Changes for CRM 2013

Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online has an updated user experience and one of the major changes is in the way that commands are presented. In most places in the web application you will see a command bar instead of a ribbon. CRM for tablets also uses data defined as ribbons to control what commands are available using a command bar that is optimized for touch.

The command bar provides a new way of displaying commands and provides better performance.

Lesson 8-2 SiteMap

Application navigation

The SiteMap provides the structure for navigation in Microsoft Dynamics CRM. It is evaluated together with your security privileges to display navigation options in the application.

If your security privileges do not provide read access to an entity specified in the SiteMap, that navigation option will not be displayed to you.

Web application navigation bar

The following image shows the web application navigation bar, showing the available areas. The default areas are SFA (SALES), CS (SERVICE), MA (MARKETING), Settings (SETTINGS), and HLP (HELP).

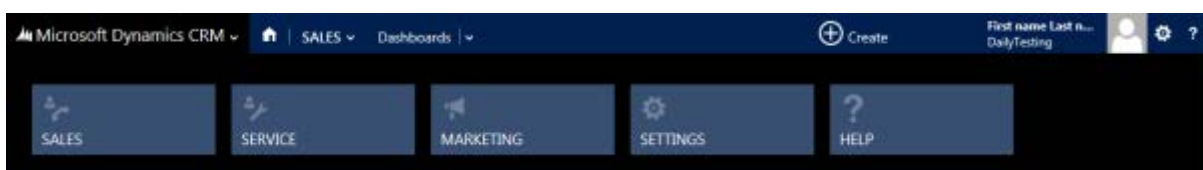


Figure 18 - Web application navigation bar

Selectable areas appear at the bottom of the navigation bar. When an area is selected the area displays as the selected area. The most recently used subarea for that area is automatically shown. When the selected area is clicked or tapped, the available groups and subareas are displayed as shown in the following image.

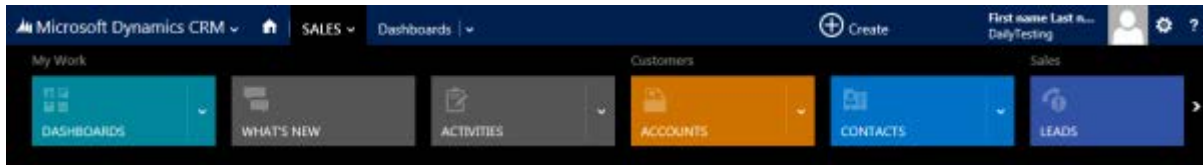


Figure 19 - Groups and Sub-Areas

In this image the SFA area is selected and the groups MyWork (My Work), Customers (Customers), and SFA (Sales) are visible. To view the remaining groups and subareas users must scroll to the right.

Each group contains a number of subareas. When a subarea is selected, the content defined in the sitemap for that subarea is displayed below the navigation

Microsoft Dynamics CRM for Outlook navigation

The following diagram shows the CRM for Outlook navigation. Microsoft Outlook presents each navigation area in alphabetical order in a tree view instead of using the order specified in the SiteMap. Each person can choose to position the reading pane at the bottom as shown or on the side. They can also choose not to show it. Each entity can be configured to determine whether it will display the reading pane.

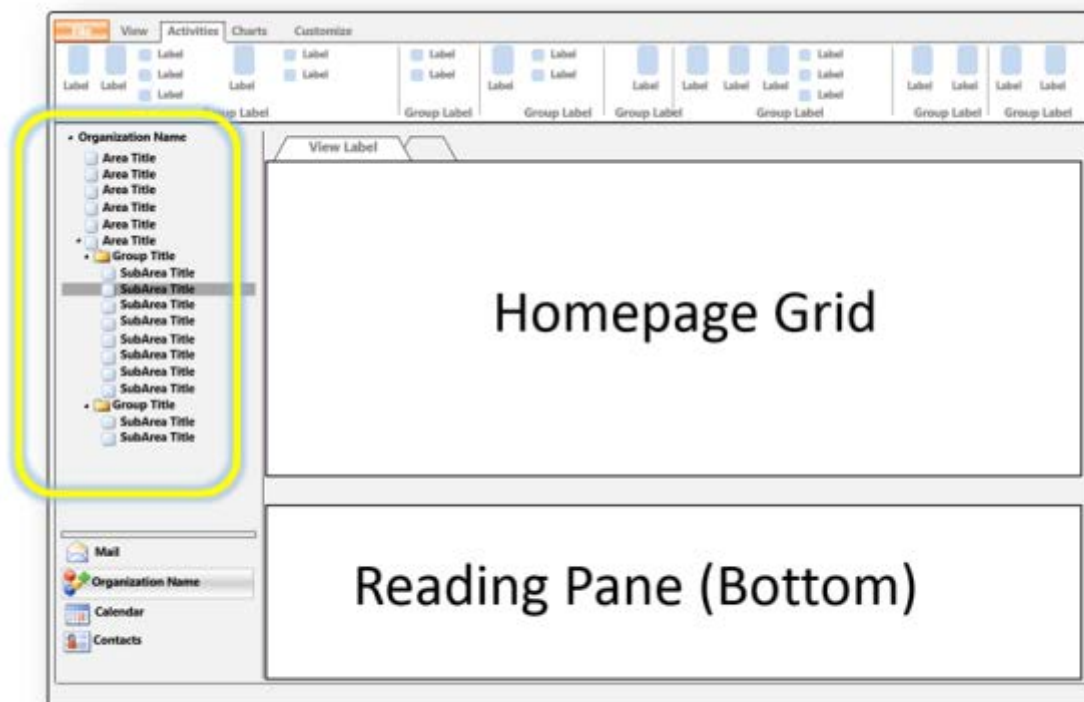


Figure 20 - CRM for Outlook Navigation

Configuration options available with the SiteMap

Commonly performed customisations that you can perform by using the SiteMap are:

- Edit labels
- Add or change icons
- Add or remove elements
- Group links within areas
- Add new pages to an area

SiteMap Schema

The following XML outline describes the high level structure of the Site Map.

```

<SiteMap>
  <Area>
    <Group>
      <SubArea>
        <Titles>
          <Title/>
        </Titles>
        <Descriptions>
          <Description/>
        </Descriptions>
        <Privilege/>
      </SubArea>
    <Titles>
      <Title/>
    </Titles>
    <Descriptions>
      <Description/>
    </Descriptions>
  </Group>
  <Titles>
    <Title/>
  </Titles>
  <Descriptions>
    <Description/>
  </Descriptions>
</Area>
</SiteMap>

```

Each element in the Site Map has attributes that control the behaviour and appearance of the elements.

Site Map Editors

Using a third-party sitemap editor can facilitate editing the sitemap.

A popular one is included in the Toolbox for Dynamics CRM 2011 released by Microsoft CRM MVP Tanguy Touzard on Codeplex.

Tools that worked with CRM 2011 will work with CRM 2013.

Lesson 8-3 Command Bar

Command Bar and RibbonDiffXML

The ribbon is still displayed in the web application for certain entity forms and it is still used for list views in Microsoft Dynamics CRM 2013 for Microsoft Office Outlook.

Both the command bar and the ribbon use the same underlying RibbonDiffXML data to define what commands to display, when the commands are enabled, and what the commands do.

For developers, the Ribbon commands provide flexibility and additional functionality including:

- Ability to Add, Remove, Override, or Disable controls

Ribbon Editors

The SDK describes the process of editing the ribbon by editing the customization.xml file directly. Several people have created ribbon editors that provide a user interface to make editing the ribbon easier.

The Ribbon Workbench for Dynamics CRM 2013 by Develop1 is recommended.

RibbonDiffXML Schema

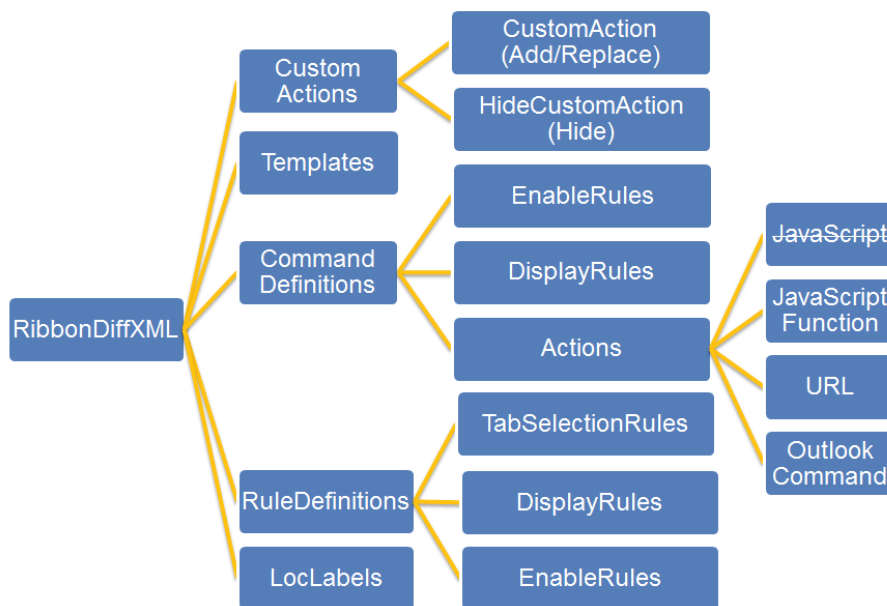


Figure 21 - RibbonDiffXML Schema

The change definitions that you specify are applied at runtime when the ribbon is displayed in the application. All of your changes will be in the <CustomAction> or <HideCustomAction> elements. These elements are applied over the default ribbon definitions provided by Microsoft Dynamics CRM 2013 and Microsoft Dynamics CRM Online.

Ribbons Available

Entity ribbons

All entities use a common ribbon definition called the Entity Ribbon Template. When you create a custom entity, the ribbon you see is the default ribbon defined by the entity ribbon template. Each system entity has a separate <RibbonDiffXml> definition that builds upon the entity ribbon template definition.

Grid ribbons

The entity grid ribbon is a collection of tabs that have an Id attribute value beginning with Mscrm.HomepageGrid.<entity logical name>.

For example, the tab with the text "Accounts" on an account entity grid is Mscrm.HomepageGrid.account.MainTab.

Subgrid ribbons

The entity subgrid ribbon is a contextual group with a collection of tabs that have an Id attribute value beginning with Mscrm.SubGrid.<entity logical name>.

For example, the tab with the text "Accounts" on account entity sub grid is Mscrm.SubGrid.account.MainTab.

Form ribbons

Each entity can have multiple forms. You can define changes to the form ribbon for all forms for that entity by adding your definition at the entity level.

Basic home tab

The basic home tab is displayed on the main application ribbon whenever an alternative tab is not defined because of entity context or a display rule that suppresses it for specific pages. For example, this tab is displayed when you view the Microsoft Dynamics CRM Help. The Id of the basic home tab is Mscrm.BasicHomeTab.

Microsoft Dynamics CRM for Microsoft Office Outlook ribbons

Microsoft Office Outlook 2007 do not display a ribbon. Microsoft Office Outlook 2010 uses the ribbon. You can use Microsoft Dynamics CRM ribbon definitions to add controls to all of them.

Microsoft Office Outlook 2007

The Microsoft Dynamics CRM for Microsoft Office Outlook controls to support older versions of Microsoft Office Outlook toolbars and menus are defined as tabs with the Id values of Mscrm.LegacyOfficeToolbar and Mscrm.LegacyOfficeMenubar, respectively.

Microsoft Office Outlook 2010

The Microsoft Dynamics CRM for Microsoft Office Outlook controls to support Microsoft Office Outlook 2010 toolbars and menus are defined as tabs with the Id values of Mscrm.Outlook14GlobalToolbar and Mscrm.Outlook14GlobalMenuBar, respectively.

Other ribbons

Several other special purpose ribbon tabs and a contextual group are defined by Microsoft Dynamics CRM. Each tab is associated with a specific <TabDisplayRule> that controls when they will display. The following table lists these tabs.

Tab	Root Id	Description
Web Resource Edit page tab.	Mscrm.WebResourceEditTab	Displays when editing Web resources within a solution.
Form Editor tab	Mscrm.FormEditorTab	Provides Save, Edit, Select, and View groups of actions for entity forms.
Form Editor Insert tab	Mscrm.FormEditorInsertTab	Provides buttons to insert Sections, Tabs, and Controls in entity forms.
Dashboard Homepage tab	Mscrm.DashboardTab	Displays in the Workplace area.
Visualization Tools Contextual Group	Mscrm.VisualizationTools	Displays when the New Chart button is clicked on the Charts tab displayed in the entity grid ribbon.
AptbookTab Homepage tab	Mscrm.AptbookTab	Displays when viewing the Service Calendar in the Service area.
Advanced Find tab	Mscrm.AdvancedFind	Displays in the Advanced Find window.
Dashboard Editor tab	Mscrm.DashboardEditorTab	Displays when editing a dashboard.
Documents tab	Mscrm.DocumentsTab	Displays if SharePoint integration has been enabled for the organization.
Chart Editor tab	Mscrm.VisualizationDesignerTab	Displays when editing a chart from the solutions window.
Search Tools Contextual Group	Mscrm.ArticleSearch	Displays when viewing the KBarticle entity.

Custom Actions

The default, an application command bar or ribbon is defined by Microsoft Dynamics CRM metadata. This default data cannot be changed, but you can include definitions of specific actions that will override the default ribbon

Types of Custom Actions

There are two types of custom actions for ribbons:

- <CustomAction> Defines an action to add or replace items in the ribbon.
- <HideCustomAction> Removes an existing ribbon element from being processed for the ribbon

Custom Actions

A custom action is a statement of how you want to change the default ribbon definition. It is evaluated and applied to the ribbon at runtime. To set the context for a custom action, you must include information about the location of the items that you want to change. Use the Location attribute to specify where your change applies.

When you add a new ribbon element, you refer to the containing element, for example, an existing tab or group.

A custom action must include a Command Definition.

Hide Custom Actions

A <HideCustomAction> is a statement that you use when you want to remove an existing ribbon element so that it is not rendered.

This does not hide the ribbon element, it actually removes the ribbon element at runtime so that it does not exist in the ribbon

Command Definition

A ribbon command defines three things:

- Enable Rules: Specifies when a specific ribbon control is enabled.
- Display Rules: Specifies when a specific ribbon element is visible.
- Actions: Specifies what code executes when a ribbon control is used.

Rule Definitions: Enable Rules

When configuring Ribbon elements you can define specific rules to control when the ribbon elements are enabled. The <EnableRule> element is used as follows:

- Use the `/RuleDefinitions/EnableRules/EnableRule` element to define rules controlling when the ribbon element should be enabled.
- Use the `/CommandDefinitions/CommandDefinition/EnableRules/EnableRule` element to associate specific enable rules to a command definition.

What does enabled mean?

With the command bar, commands that are disabled are hidden. With the ribbon, commands that are disabled are visible but do not respond to events

Control when ribbon elements are enabled

Enable rules are intended to be re-used. By defining them with rule definitions, you can use the same enable rule for many command definitions. When more than one enable rule is defined for a command definition, all of the enable rules must evaluate as true for the ribbon element to be enabled.

All Enable rules provide an optional attribute to specify whether the default value of the rule is true or false and an optional `InvertResult` attribute to allow for returning a negative result when the item being tested returns true.

Rule Definitions: Display Rules

When configuring ribbon elements, you can define specific rules to control when the ribbon elements will display.

- Use the `/RuleDefinitions/DisplayRules/<DisplayRule>` element to define rules controlling when the ribbon element should be displayed.
- Use the `/CommandDefinitions/CommandDefinition/DisplayRules/<DisplayRule>` element to associate specific display rules to a command definition.

Control when ribbon elements are displayed

By defining display rules in rule definitions, you can use the same display rule for many command definitions. When more than one display rule is defined for a command definition, all of the display rules must evaluate as true for the ribbon element to be displayed.

All display rules provide an optional attribute to specify whether the default value of the rule is true or false and an optional `InvertResult` attribute to enable returning a negative result when the item being tested returns true.

Rule Definitions: Actions

Define the actions to be performed by a command bar or ribbon control in a `<CommandDefinition>` element together with rules that control whether the control is enabled or visible in the ribbon.

A Ribbon control can perform two types of action and may include multiple actions:

- JavaScript Functions: A <JavaScriptFunction> element references a function defined in a JScript Web resource.
- Open a URL: The ribbon opens a URL using the value from an Address attribute in the <Url> element. Additional parameters can pass information about how what querystring parameters are passed and the mode in which the window opens.