Microsoft Dynamics™ CRM 4.0
Development Guide
for Service Providers

Get the Latest Version of This Document

Before starting with any deployment or development process, you should make sure you have the latest version of this document. Search for this document in the Business Solutions section of the Microsoft Download Center at http://www.microsoft.com/downloads/Browse.aspx?displaylang=en&productID=67D35328-CB42-4879-A35A-D59E94D9DACA.

Table of Contents

Introduction

Welcome to the Microsoft Dynamics™ CRM 4.0 Development Guide for Service Providers. This guide elaborates on coding practices, processes, and methods found in the Microsoft Dynamics CRM 4.0 Software Development Kit (SDK) that are of interest to service providers. In particular, service providers can use this document to help develop an environment where they can provision their hosted Microsoft Dynamics CRM customers and users programmatically.

Assumptions

Before reading this document and working with the Microsoft Dynamics CRM 4.0 SDK, please spend a few moments reviewing the following assumptions concerning reader and system preparation.

Familiarity with the Microsoft Dynamics CRM 4.0 SDK

This document expands on the information found in the Microsoft Dynamics CRM 4.0 SDK. Before reading this document to learn about multi-tenant–specific development, we strongly recommend you first become familiar with the SDK, as it provides a wide range of instructive and practical information. In addition to a comprehensive class library reference, the SDK offers conceptual overviews, integration details, database schemas, and sample code.

Functioning Microsoft Dynamics CRM 4.0 Environment

Before you can use this guide effectively, it is very important that you have properly installed and configured your hosted Microsoft Dynamics CRM environment. To ensure a successful installation, follow the detailed procedures in the Readme and the *Microsoft Dynamics 4.0 Deployment Walkthrough for Service Providers*. These documents, available on TechNet, describe the essential preparation steps for setting up the environment prior to installing Microsoft Dynamics CRM and working with the SDK.

Running the Latest Edition of the Solution

This document does not address earlier editions of Microsoft Dynamics CRM; if you are running an earlier version of Microsoft Dynamics CRM, you will need to upgrade to version 4.0 before working with the SDK.

Development Environment and Tools

You should be familiar with general software development practices in general and C# coding practices in particular.

In addition to working with Microsoft® Visual C#, you are also expected to be familiar with using Web services, and using the Microsoft Windows PowerShell™ 1.0 SDK.

You will also need to have access to a suitable development environment.

For more information about accessing appropriate development environments and tools, see Additional Resources at the end of this document.

Provisioning Organizations Programmatically

The focus of this documentation is to discuss how to use the Microsoft Dynamics CRM SDK to develop an automated end-to-end provisioning system. The sample code is written in C# to provision hosted Microsoft Dynamics CRM organizations and users, and to control a variety of other elements, including:

- Active Directory multi-tenancy configuration
- Exchange Server 2007 configuration for multi-tenancy
- E-mail router configuration for the organization

Special Considerations for MPS Users

Microsoft Dynamics CRM 4.0 can be deployed in a Microsoft Solution for Hosted Messaging and Collaboration (HMC) 4.0 environment in order to offer hosted CRM services. Those service providers who are using HMC 4.0 and who want to integrate the Microsoft Provisioning System (MPS) with Microsoft Dynamics CRM 4.0 to carry out automated provisioning tasks can review the following list of components required for the integration.

Required Components

The components needed to integrate Microsoft Dynamics CRM 4.0 with HMC 4.0 and Microsoft Provisioning System (MPS) include:

- CRM Web Service
- .NET integration layer
- Hosted CRM Namespace
- Business Logic
- Resource Management calls
- Plan Manager calls
- CRM Resource Manager
- Allocation management for CRM resources
- Managed CRM Namespace
- Security and Delegated Administration
- Service Provisioning (User and Customer creation)
- Support for Multi-tenancy
- CRM Provider
- Wrapper for the CRM API

Additionally, the control panel for administration in place at the hosting provider would need to be integrated into the CRM Web Service to support Delegated Administration, Order Handling and Billing.

For more detailed documentation about, and tools for, developing the various MPS components for integration into HMC 4.0, see the Microsoft Provisioning System SDK July 2007.

End-to-End Provisioning with the Microsoft Dynamics CRM 4.0 SDK

This section of the document provides a complete sequence of tasks you will need to perform in the proper order to programmatically provision Microsoft Dynamics CRM 4.0 entities. Each task provides sample code to illustrate proper use of the methods.

Creating and Configuring a Customer Organization

The first task to complete is to create a customer organization.

► **To create the AlpineSkiHouse customer organization and groups**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx)

Arguments

Remarks

Sample

```
namespace Microsoft.CRM40.Samples
{
    class CreateOrganization
    {
        static void Main()
        {
            string orgName = "AlpineSkiHouse";          //
   Organizational unit.
            string HostingPath =
   "LDAP://localhost/OU=Hosting,DC=litwareinc,DC=com"; //
   Binding path.

            DirectoryEntry objAD = new
   DirectoryEntry(HostingPath);

            // Create Organizational Unit.
            try
            {
                // Create Organizational Unit
                DirectoryEntry objOU =
   objAD.Children.Add("OU=" + OrgName,
   "OrganizationalUnit");
                objOU.Properties["description"].Add(OrgName
   + " Business Org");
                objOU.CommitChanges();
```

```
                // Create Admins Group
                DirectoryEntry AdminGroup =
   objOU.Children.Add("CN=Admins@" + OrgName, "Group");

   AdminGroup.Properties["samAccountName"].Value = "Admins "
   + OrgName;
                AdminGroup.CommitChanges();


                // Create AllUsers Group
                DirectoryEntry AllUsersGroup =
   objOU.Children.Add("CN=AllUsers@" + OrgName, "Group");

   AllUsersGroup.Properties["samAccountName"].Value =
   "AllUsers_" + OrgName;
                AllUsersGroup.CommitChanges();
   }
            catch (Exception e)
            {
                Console.WriteLine("Error:   Create failed
   {0}", e.Message);
            }
        }
    }
}
```

► **To make the AlpineSkiHouse groups members of the Hosting groups**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx)

Arguments

Remarks

Sample

```
namespace Microsoft.CRM40.Samples
{
    class GroupMembers
    {
        static void Main()
        {
            string OrgName = "AlpineSkiHouse";
   // CompanyName.
            string OrgAdminsGroup = "CN=Admins@" + OrgName +
   ",OU=" + OrgName + ",OU=Hosting,DC=LitwareInc,DC=com";
```

```
            string OrgAllUsersGroup = "CN=AllUsers@" +
    OrgName + ",OU=" + OrgName +
    ",OU=Hosting,DC=LitwareInc,DC=com";
            DirectoryEntry HostingAdminsGroup = new
    DirectoryEntry("LDAP://localhost/CN=AllCustomerAdminsGrou
    ps@Hosting,OU=Hosting,DC=litwareinc,DC=com");
            DirectoryEntry HostingAllUsersGroup = new
    DirectoryEntry("LDAP://localhost/CN=AllCustomers@Hosting,
    OU=Hosting,DC=litwareinc,DC=com");


            try
            {
                // Add Customer Admin Groups to Hosting
    Admins

    HostingAdminsGroup.Properties["member"].Add(OrgAdminsGrou
    p);
                HostingAdminsGroup.CommitChanges();
                // Add Customer User Groups to Hosting Users

    HostingAllUsersGroup.Properties["member"].Add(OrgAllUsers
    Group);
                HostingAllUsersGroup.CommitChanges();
            }
            catch (Exception e)
            {
                Console.WriteLine("Error:   Group Add failed
    {0}", e.StackTrace);
            }
        }
     }
}
```

► **To secure the customer organization**

API – System.Security.Principal (http://msdn2.microsoft.com/en-us/library/system.security.principal.aspx)

Arguments

Remarks

Sample

```
namespace Microsoft.CRM40.Samples
{
    class RemoveACLs
    {
```

```
        static void Main(string[] args)
        {
            string OrgName = "AlpineSkiHouse";
   // CompanyName.

            DirectoryEntry OrgPath = new
   DirectoryEntry("LDAP://localhost/OU=" + OrgName +
   ",OU=Hosting,DC=litwareinc,DC=com");
            ActiveDirectorySecurity OrgSecurity =
   OrgPath.ObjectSecurity;

            IdentityReference AuthUsersGroup = new
   NTAccount("Authenticated Users");
            IdentityReference EveryoneGroup = new
   NTAccount("Everyone");
            OrgSecurity.PurgeAccessRules(AuthUsersGroup);
            OrgSecurity.PurgeAccessRules(EveryoneGroup);

        }
    }
}
```

► **To configure the Domain User group for List Object access to the customer OU**

API – System.DirectoryServices ActiveDirectoryAccessRule (http://msdn2.microsoft.com/en-us/library/system.directoryservices.activedirectoryaccessrule.aspx)

Arguments

Remarks

Sample

```
namespace Microsoft.CRM40.Samples
{
    class AddACLs
    {
        static void Main(string[] args)
        {
            string OrgName = "AlpineSkiHouse";
   // CompanyName.

            DirectoryEntry OrgPath = new
   DirectoryEntry("LDAP://localhost/OU=" + OrgName +
   ",OU=Hosting,DC=litwareinc,DC=com");
```

```
            IdentityReference DomainUsersGroup = new
    NTAccount("Domain Users");


            ActiveDirectoryAccessRule accessRule = new
    ActiveDirectoryAccessRule(

    DomainUsersGroup,

    ActiveDirectoryRights.ListObject,

    AccessControlType.Allow,

    ActiveDirectorySecurityInheritance.None);


            ActiveDirectorySecurity CustomerOrgSecurity =
    OrgPath.ObjectSecurity;
            CustomerOrgSecurity.AddAccessRule(accessRule);
        }
     }
}
```

Adding a UPN Suffix

► **To configure UPN suffixes for the customer organization**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx)

Arguments

Remarks

Sample

```
namespace Microsoft.CRM40.Samples
{
    class AddUPNSuffix
    {
        static void Main(string[] args)
        {
            string OrgSuffix = "AlpineSkiHouse.com";
            string HostingPath = "LDAP://localhost/OU=" +
    OrgName + ",OU=Hosting,DC=litwareinc,DC=com";


            DirectoryEntry objAD = new
    DirectoryEntry(HostingPath);
            objAD.Properties["uPNSuffixes"].Add(OrgSuffix);
            objAD.CommitChanges();
```

```
        }
    }
}
```

Enabling Hosted Exchange for Customer Organizations

Creating Accepted Domains

► **To add customer organization accepted domains**

API – Windows PowerShell via C#

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Host;
using System.Management.Automation.Runspaces;

namespace Microsoft.CRM40.Samples
{
    class CreateAcceptedDomain
    {
        static void Main(string[] args)
        {
            string OrgSMTPName = "AlpineSkiHouse.com";
            string PsText = "New-AcceptedDomain”;

            RunspaceConfiguration rsConfig =
  RunspaceConfiguration.Create();
            PSSnapInException snapInException = null;
            PSSnapInInfo info =
  rsConfig.AddPSSnapIn("Microsoft.Exchange.Management.Power
  Shell.Admin", out snapInException);

            if (snapInException != null)
            {

  System.Console.Write(snapInException.Message);
                return;
            }
```

```
            Runspace myRunSpace =
   RunspaceFactory.CreateRunspace(rsConfig);
            myRunSpace.Open();
            Pipeline pipeLine = myRunSpace.CreatePipeline();

            Command myCommand = new Command(PsText);
            myCommand.Parameters.Add("Name", OrgSMTPName);
            myCommand.Parameters.Add("DomainName",
   OrgSMTPName);
            pipeLine.Commands.Add(myCommand);

            Collection<PSObject> commandResults =
   pipeLine.Invoke();
        }
     }
}
```

Creating the DNS Zone for the Customer Organization

► **To create the Alpine Ski House external DNS zones**

API – Requires reference to Systems.Management

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Management;


namespace CreateDNSZone
{
    class Program
    {
        static void Main(string[] args)
        {
            string ZoneName = "AlpineSkiHouse.com";
            int ZoneType = 0; //AD Integrated
            string DNSServer = "DC1";
```

```
            ManagementScope scope = new
    ManagementScope("\\\\" + DNSServer +
    "\\root\\microsoftdns");


            ManagementClass zone = new
    ManagementClass(scope, new
    ManagementPath("MicrosoftDNS_Zone"), null);
            ManagementBaseObject inputs =
    zone.GetMethodParameters("CreateZone");
            ManagementBaseObject outputs = null;

            inputs["ZoneName"] = ZoneName;
            inputs["ZoneType"] = ZoneType;

            try
            {
                outputs = zone.InvokeMethod("CreateZone",
    inputs, null);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Source + " " +
    e.Message);
            }
        }
     }
}
```

► **To add the alpineskihouse.com Mail Exchanger (MX) entry**

API – System.Management

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Management;

namespace CreateMXRecord
{
    class Program
    {
```

```
        static void Main(string[] args)
        {
            string DomainName = "AlpineSkiHouse.com";
            string DNSServer = "DC1";

            ManagementScope scope = new
   ManagementScope("\\\\" + DNSServer +
   "\\root\\microsoftdns");
            ManagementClass MXType = new
   ManagementClass(scope, new
   ManagementPath("MicrosoftDNS_MXTYPE"), null);
            object[] MXRecord = { DNSServer, DomainName,
   DomainName, 1, 86400, 10, "smtp." + DomainName };

   MXType.InvokeMethod("CreateInstanceFromPropertyData",
   MXRecord);
        }
    }
}
```

Creating and Configuring a Customer GAL

► **To create and configure a global address list (GAL) for alpineskihouse.com**

API – Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Host;
using System.Management.Automation.Runspaces;

namespace Microsoft.CRM40.Samples
{
    class CreateGlobalAddressList
    {
        static void Main(string[] args)
        {
            string OrgName = "AlpineSkiHouse";
            string PsText = "New-GlobalAddressList";
```

```
            RunspaceConfiguration rsConfig =
    RunspaceConfiguration.Create();
            PSSnapInException snapInException = null;
            PSSnapInInfo info =
    rsConfig.AddPSSnapIn("Microsoft.Exchange.Management.Power
    Shell.Admin", out snapInException);

            if (snapInException != null)
            {

    System.Console.Write(snapInException.Message);
                return;
            }

            Runspace myRunSpace =
    RunspaceFactory.CreateRunspace(rsConfig);
            myRunSpace.Open();
            Pipeline pipeLine = myRunSpace.CreatePipeline();

            Command myCommand = new Command(PsText);
            myCommand.Parameters.Add("Name", OrgName + "
    GAL");
            myCommand.Parameters.Add("IncludedRecipients",
    "AllRecipients");
            myCommand.Parameters.Add("ConditionalCompany",
    OrgName);

            pipeLine.Commands.Add(myCommand);

            Collection<PSObject> commandResults =
    pipeLine.Invoke();
        }
     }
}
```

Creating and Configuring a Customer Address List

► **To create and configure an address list for Alpine Ski House**

API – Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Host;
using System.Management.Automation.Runspaces;

namespace Microsoft.CRM40.Samples
{
    class CreateAddressList
    {
        static void Main(string[] args)
        {
            string OrgName = "AlpineSkiHouse";
            string PsText = "New-AddressList";

            RunspaceConfiguration rsConfig =
   RunspaceConfiguration.Create();
            PSSnapInException snapInException = null;
            PSSnapInInfo info =
   rsConfig.AddPSSnapIn("Microsoft.Exchange.Management.Power
   Shell.Admin", out snapInException);

            if (snapInException != null)
            {

   System.Console.Write(snapInException.Message);
                return;
            }

            Runspace myRunSpace =
   RunspaceFactory.CreateRunspace(rsConfig);
            myRunSpace.Open();
            Pipeline pipeLine = myRunSpace.CreatePipeline();

            Command myCommand = new Command(PsText);
            myCommand.Parameters.Add("Name", OrgName + "
   AL");
            myCommand.Parameters.Add("IncludedRecipients",
   "AllRecipients");
            myCommand.Parameters.Add("Container", "\\");
            myCommand.Parameters.Add("ConditionalCompany",
   OrgName);
```

```
            pipeLine.Commands.Add(myCommand);

            Collection<PSObject> commandResults =
   pipeLine.Invoke();
        }
     }
}
```

Setting Security Policies for Customer Address Lists

► **To set the security policies for customer global address list**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx) or Windows PowerShell

Arguments

Remarks –Option: Only needed if configuring hosted Exchange

Sample

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Security.AccessControl;
using System.Security.Principal;
using System.DirectoryServices;

namespace Microsoft.CRM40.Samples
{
    class ConfigureGALSecurity
    {
        static void Main(string[] args)
        {
            string OrgName = "AlpineSkiHouse";
            string GalDN = "CN=" + OrgName + " GAL,CN=All
   Global Address Lists,CN=Address Lists
   Container,CN=Litware Inc,CN=Microsoft
   Exchange,CN=Services,CN=Configuration,DC=litwareinc,DC=co
   m";

            DirectoryEntry OrgGal = new
   DirectoryEntry("LDAP://" + GalDN);
```

```
            List<ActiveDirectoryAccessRule> newAccessRules =
new List<ActiveDirectoryAccessRule>();

            string[] OrgGroups = new string[] {"Admins",
"AllUsers"};

            // Add Read and Open Address List Rights to
Admin@ and Allusers@ groups
            foreach (string group in OrgGroups)
            {
                DirectoryEntry OrgGroup = new
DirectoryEntry("LDAP://CN=" + group + "@" + OrgName +
",OU=" + OrgName + ",OU=Hosting,DC=LitwareInc,DC=com");
                IdentityReference OrgGroupTrustee = new
NTAccount(OrgGroup.Properties["sAMAccountName"].Value.ToS
tring());

                // Add GenericRead for each Group
                ActiveDirectoryAccessRule accessRule = new
ActiveDirectoryAccessRule(
                                    OrgGroupTrustee,

ActiveDirectoryRights.GenericRead,
                                    AccessControlType.Allow,

ActiveDirectorySecurityInheritance.SelfAndChildren);

                newAccessRules.Add(accessRule);

                // Add Open Address List for each Group
                accessRule = new ActiveDirectoryAccessRule(
                                    OrgGroupTrustee,

ActiveDirectoryRights.ExtendedRight,
                                    AccessControlType.Allow,
                                    //GUID represents "Open
Address List" extended right
(http://msdn2.microsoft.com/en-us/library/ms684393.aspx)
                                    new Guid("a1990816-4298-
11d1-ade2-00c04fd8d5cd"),

ActiveDirectorySecurityInheritance.SelfAndChildren);

                newAccessRules.Add(accessRule);
```

```
            }

            ActiveDirectorySecurity addressListSecurity =
OrgGal.ObjectSecurity;

            foreach (ActiveDirectoryAccessRule rule in
newAccessRules)
            {
                addressListSecurity.AddAccessRule(rule);
            }

            //Breaking inheritance incorrectly orders ACE's

addressListSecurity.SetAccessRuleProtection(true, true);

            //Resort all ACE's on object - currently puts
inheritance back
            OrgGal = SortDACLs(OrgGal);

        }

        public static DirectoryEntry
SortDACLs(DirectoryEntry GalEntry)
        {
            AuthorizationRuleCollection DaclRules = null;
            AccessRule accessRule = null;
            List<AccessRule> inheritedRules = new
List<AccessRule>();
            List<AccessRule> deniedRules = new
List<AccessRule>();
            List<AccessRule> allowedRules = new
List<AccessRule>();

            // Retreive the DACL ACE's for re-ordering
            GalEntry.Options.SecurityMasks =
SecurityMasks.Dacl;
            DaclRules =
GalEntry.ObjectSecurity.GetAccessRules(true, true,
typeof(NTAccount));

            // Make separate collections of the
inheritedRule, Allow and Deny DaclRules.
            foreach (AuthorizationRule rule in DaclRules)
```

```
            {
                accessRule = (AccessRule)rule;

                if (accessRule.IsInherited)
                {
                    inheritedRules.Add(accessRule);
                }
                else
                {
                    switch (accessRule.AccessControlType)
                    {
                        case AccessControlType.Allow:
                            allowedRules.Add(accessRule);
                            break;
                        case AccessControlType.Deny:
                            deniedRules.Add(accessRule);
                            break;
                    }
                }
            }

            // Create a new container for the ACE rules
            ActiveDirectorySecurity objectSecurity = new
ActiveDirectorySecurity();

            // Add the rules to the new container in the
required order
            foreach (ActiveDirectoryAccessRule rule in
deniedRules)
            {
                objectSecurity.AddAccessRule(rule);
            }
            foreach (ActiveDirectoryAccessRule rule in
allowedRules)
            {
                objectSecurity.AddAccessRule(rule);
            }
            foreach (ActiveDirectoryAccessRule rule in
inheritedRules)
            {
                objectSecurity.AddAccessRule(rule);
            }
```

```
            GalEntry.ObjectSecurity = objectSecurity;
            GalEntry.CommitChanges();

            return GalEntry;
        }
    }
}
```

► **To set the security policies for customer address list**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx) or Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Security.AccessControl;
using System.Security.Principal;
using System.DirectoryServices;

namespace Microsoft.CRM40.Samples
{
    class ConfigureALSecurity
    {
        static void Main(string[] args)
        {
            string OrgName = "AlpineSkiHouse";
            string GalDN = "CN=" + OrgName + " AL,CN=All
   Address Lists,CN=Address Lists Container,CN=Litware
   Inc,CN=Microsoft
   Exchange,CN=Services,CN=Configuration,DC=litwareinc,DC=co
   m";

            DirectoryEntry OrgAl = new
   DirectoryEntry("LDAP://" + GalDN);

            List<ActiveDirectoryAccessRule> newAccessRules =
   new List<ActiveDirectoryAccessRule>();
```

```
            string[] OrgGroups = new string[] {"Admins",
"AllUsers"};

            // Add Read and Open Address List Rights to
Admin@ and Allusers@ groups
            foreach (string group in OrgGroups)
            {
                DirectoryEntry OrgGroup = new
DirectoryEntry("LDAP://CN=" + group + "@" + OrgName +
",OU=" + OrgName + ",OU=Hosting,DC=LitwareInc,DC=com");
                IdentityReference OrgGroupTrustee = new
NTAccount(OrgGroup.Properties["sAMAccountName"].Value.ToS
tring());

                // Add GenericRead for each Group
                ActiveDirectoryAccessRule accessRule = new
ActiveDirectoryAccessRule(
                                    OrgGroupTrustee,

ActiveDirectoryRights.GenericRead,
                                    AccessControlType.Allow,

ActiveDirectorySecurityInheritance.SelfAndChildren);

                newAccessRules.Add(accessRule);

                // Add Open Address List for each Group
                accessRule = new ActiveDirectoryAccessRule(
                                    OrgGroupTrustee,

ActiveDirectoryRights.ExtendedRight,
                                    AccessControlType.Allow,
                                    //GUID represents "Open
Address List" extended right
(http://msdn2.microsoft.com/en-us/library/ms684393.aspx)
                                    new Guid("a1990816-4298-
11d1-ade2-00c04fd8d5cd"),

ActiveDirectorySecurityInheritance.SelfAndChildren);

                newAccessRules.Add(accessRule);
            }
```

```
            ActiveDirectorySecurity addressListSecurity =
OrgAl.ObjectSecurity;

            foreach (ActiveDirectoryAccessRule rule in
newAccessRules)
            {
                addressListSecurity.AddAccessRule(rule);
            }

            //Breaking inheritance incorrectly orders ACE's

addressListSecurity.SetAccessRuleProtection(true, true);

            //Resort all ACE's on object - currently puts
inheritance back
            OrgAl = SortDACLs(OrgAl);

        }

        public static DirectoryEntry
SortDACLs(DirectoryEntry AlEntry)
        {
            AuthorizationRuleCollection DaclRules = null;
            AccessRule accessRule = null;
            List<AccessRule> inheritedRules = new
List<AccessRule>();
            List<AccessRule> deniedRules = new
List<AccessRule>();
            List<AccessRule> allowedRules = new
List<AccessRule>();

            // Retreive the DACL ACE's for re-ordering
            AlEntry.Options.SecurityMasks =
SecurityMasks.Dacl;
            DaclRules =
GalEntry.ObjectSecurity.GetAccessRules(true, true,
typeof(NTAccount));

            // Make separate collections of the
inheritedRule, Allow and Deny DaclRules.
            foreach (AuthorizationRule rule in DaclRules)
            {
                accessRule = (AccessRule)rule;
```

```
            if (accessRule.IsInherited)
            {
                inheritedRules.Add(accessRule);
            }
            else
            {
                switch (accessRule.AccessControlType)
                {
                    case AccessControlType.Allow:
                        allowedRules.Add(accessRule);
                        break;
                    case AccessControlType.Deny:
                        deniedRules.Add(accessRule);
                        break;
                }
            }
        }

        // Create a new container for the ACE rules
        ActiveDirectorySecurity objectSecurity = new
ActiveDirectorySecurity();

        // Add the rules to the new container in the
required order
        foreach (ActiveDirectoryAccessRule rule in
deniedRules)
        {
            objectSecurity.AddAccessRule(rule);
        }
        foreach (ActiveDirectoryAccessRule rule in
allowedRules)
        {
            objectSecurity.AddAccessRule(rule);
        }
        foreach (ActiveDirectoryAccessRule rule in
inheritedRules)
        {
            objectSecurity.AddAccessRule(rule);
        }

        AlEntry.ObjectSecurity = objectSecurity;
```

```
            AlEntry.CommitChanges();

            return AlEntry;
        }
    }
}
```

Creating and Configuring an Offline Address List

► **To create a custom Offline Address List for Alpine Ski House**

API – Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Host;
using System.Management.Automation.Runspaces;

namespace Microsoft.CRM40.Samples
{
    class CreateOfflineAddresList
    {
        static void Main(string[] args)
        {
            string OrgName = "AlpineSkiHouse";
            string OABServer = "DC1";
            string OABVdir = "DC1\\OAB (Default Web Site)";
            string PsText = "New-OfficeAddressBook";

            RunspaceConfiguration rsConfig =
   RunspaceConfiguration.Create();
            PSSnapInException snapInException = null;
            PSSnapInInfo info =
   rsConfig.AddPSSnapIn("Microsoft.Exchange.Management.Power
   Shell.Admin", out snapInException);

            if (snapInException != null)
            {
```

```
   System.Console.Write(snapInException.Message);
                return;
            }

            Runspace myRunSpace =
   RunspaceFactory.CreateRunspace(rsConfig);
            myRunSpace.Open();
            Pipeline pipeLine = myRunSpace.CreatePipeline();

            Command myCommand = new Command(PsText);
            myCommand.Parameters.Add("Name", OrgName + "
   OAL");
            myCommand.Parameters.Add("Server", OABServer);
            myCommand.Parameters.Add("AddressLists",
   OrgName + " AL");

               myCommand.Parameters.Add("PublicFolderDistributionEna
   bled", false);
            myCommand.Parameters.Add("DiffRetentionPeriod",
   "Unlimited");
            myCommand.Parameters.Add("VirtualDirectories",
   OABVdir);
            myCommand.Parameters.Add("Schedule",
   Microsoft.Exchange.Data.Schedule.Never);

            pipeLine.Commands.Add(myCommand);

            Collection<PSObject> commandResults =
   pipeLine.Invoke();
         }
     }
}
```

Using ADSI Edit to Secure Offline Address Lists

► **To configure permissions on Alpine Ski House offline address list**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx) or Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System;
```

```
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Security.AccessControl;
using System.Security.Principal;
using System.DirectoryServices;

namespace Microsoft.CRM40.Samples
{
    class ConfigureOALSecurity
    {
        static void Main(string[] args)
        {
            string OrgName = "AlpineSkiHouse";
            string OalDN = "CN=" + OrgName + "
   OAL,CN=Offline Address Lists,CN=Address Lists
   Container,CN=Litware Inc,CN=Microsoft
   Exchange,CN=Services,CN=Configuration,DC=litwareinc,DC=co
   m";

            DirectoryEntry OrgOal = new
   DirectoryEntry("LDAP://" + OalDN);

            List<ActiveDirectoryAccessRule> newAccessRules =
   new List<ActiveDirectoryAccessRule>();

            string[] OrgGroups = new string[] {"Admins",
   "AllUsers"};

            // Add Read and Open Address List Rights to
   Admin@ and Allusers@ groups
            foreach (string group in OrgGroups)
            {
                DirectoryEntry OrgGroup = new
   DirectoryEntry("LDAP://CN=" + group + "@" + OrgName +
   ",OU=" + OrgName + ",OU=Hosting,DC=LitwareInc,DC=com");
                IdentityReference OrgGroupTrustee = new
   NTAccount(OrgGroup.Properties["sAMAccountName"].Value.ToS
   tring());

                // Add GenericRead for each Group
                ActiveDirectoryAccessRule accessRule = new
   ActiveDirectoryAccessRule(
                                    OrgGroupTrustee,
```

```
ActiveDirectoryRights.GenericRead,
                                    AccessControlType.Allow,

ActiveDirectorySecurityInheritance.SelfAndChildren);

                newAccessRules.Add(accessRule);

                // Add Open Address List for each Group
                accessRule = new ActiveDirectoryAccessRule(
                                    OrgGroupTrustee,

ActiveDirectoryRights.ExtendedRight,
                                    AccessControlType.Allow,
                                    //GUID represents "Open
Address List" extended right
(http://msdn2.microsoft.com/en-us/library/ms684393.aspx)
                                    new Guid("a1990816-4298-
11d1-ade2-00c04fd8d5cd"),

ActiveDirectorySecurityInheritance.SelfAndChildren);

                newAccessRules.Add(accessRule);
            }

            ActiveDirectorySecurity addressListSecurity =
OrgOal.ObjectSecurity;

            foreach (ActiveDirectoryAccessRule rule in
newAccessRules)
            {
                addressListSecurity.AddAccessRule(rule);
            }

            //Breaking inheritance incorrectly orders ACE's

addressListSecurity.SetAccessRuleProtection(true, true);

            //Resort all ACE's on object - currently puts
inheritance back
            OrgOal = SortDACLs(OrgOal);

        }
```

```
     public static DirectoryEntry
SortDACLs(DirectoryEntry OalEntry)
     {
          AuthorizationRuleCollection DaclRules = null;
          AccessRule accessRule = null;
          List<AccessRule> inheritedRules = new
List<AccessRule>();
          List<AccessRule> deniedRules = new
List<AccessRule>();
          List<AccessRule> allowedRules = new
List<AccessRule>();

          // Retreive the DACL ACE's for re-ordering
          OalEntry.Options.SecurityMasks =
SecurityMasks.Dacl;
          DaclRules =
OalEntry.ObjectSecurity.GetAccessRules(true, true,
typeof(NTAccount));

          // Make separate collections of the
inheritedRule, Allow and Deny DaclRules.
          foreach (AuthorizationRule rule in DaclRules)
          {
              accessRule = (AccessRule)rule;

              if (accessRule.IsInherited)
              {
                  inheritedRules.Add(accessRule);
              }
              else
              {
                  switch (accessRule.AccessControlType)
                  {
                      case AccessControlType.Allow:
                          allowedRules.Add(accessRule);
                          break;
                      case AccessControlType.Deny:
                          deniedRules.Add(accessRule);
                          break;
                  }
              }
```

```
            }

            // Create a new container for the ACE rules
            ActiveDirectorySecurity objectSecurity = new
    ActiveDirectorySecurity();

            // Add the rules to the new container in the
    required order
            foreach (ActiveDirectoryAccessRule rule in
    deniedRules)
            {
                objectSecurity.AddAccessRule(rule);
            }
            foreach (ActiveDirectoryAccessRule rule in
    allowedRules)
            {
                objectSecurity.AddAccessRule(rule);
            }
            foreach (ActiveDirectoryAccessRule rule in
    inheritedRules)
            {
                objectSecurity.AddAccessRule(rule);
            }

            OalEntry.ObjectSecurity = objectSecurity;
            OalEntry.CommitChanges();

            return OalEntry;
        }
    }
}
```

Creating the Customer Organization Administrator Account

► **To create the customer administrator account**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx) or Windows PowerShell

Arguments

Remarks

Sample

```
using System;
using System.DirectoryServices;
```

```
namespace Microsoft.CRM40.Samples
{
    class CreateAdminUser
    {
        static void Main()
        {

            string OrgName = "AlpineSkiHouse";          //
   Organizational unit.
            string FirstName = "John";
            string LastName = "Chen";
            string userPassword = "Pass1word";
            string AdminUser = FirstName + LastName + "@" +
   OrgName;
            string UPN = FirstName + LastName.Substring(0,
   1) + "@" + OrgName;
            string SamAccountName = UPN.Replace("@", "_");
            string OrgPath = "LDAP://localhost/OU="+ OrgName
   +",OU=Hosting,DC=litwareinc,DC=com"; // Binding path.

            DirectoryEntry objOrg = new
   DirectoryEntry(OrgPath);

            // Create Admin User.
            try
            {
                DirectoryEntry objAdminUser =
   objOrg.Children.Add("CN=" + AdminUser, "user");

   objAdminUser.Properties["description"].Add(OrgName + "
   Admin Account");

   objAdminUser.Properties["givenName"].Add(FirstName);
                objAdminUser.Properties["sn"].Add(LastName);

   objAdminUser.Properties["userPrincipalName"].Add(UPN);

   objAdminUser.Properties["samAccountName"].Add(SamAccountN
   ame);

                objAdminUser.CommitChanges();
```

```
                objAdminUser.Invoke("SetPassword", new
   object[] {userPassword});

   objAdminUser.Properties["userAccountControl"].Value =
   0x200;
                objAdminUser.CommitChanges();
            }
            catch (Exception e)
            {
                Console.WriteLine("Error:   Create failed
   {0}", e.Message);
            }
        }
    }
}
```

► **To modify the customer administrator group memberships**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx) or Windows PowerShell

Arguments

Remarks

Sample

```
using System;
using System.DirectoryServices;

namespace Microsoft.CRM40.Samples
{
    class AddUserToGroup
    {
        static void Main()
        {
            string OrgName = "AlpineSkiHouse";
   // CompanyName.
            string user = "JohnChen@alpineskihouse";
   // Admin User name

            string AdminUserDN = "CN=" + user + ",OU=" +
   OrgName + ",OU=Hosting,DC=LitwareInc,DC=com";
            string AdminsGroupDN = "CN=Admins@" + OrgName +
   ",OU=" + OrgName + ",OU=Hosting,DC=LitwareInc,DC=com";
```

```
            Console.WriteLine(AdminUserDN + "\n" +
    AdminsGroupDN);
            try
            {
                Console.WriteLine("Adding user " + user + "
    to " + OrgName + " Admin Group");
                DirectoryEntry AdminsGroup = new
    DirectoryEntry("LDAP://" + AdminsGroupDN);

    AdminsGroup.Properties["member"].Add(AdminUserDN);
                AdminsGroup.CommitChanges();
            }
            catch
    (System.DirectoryServices.DirectoryServicesCOMException
    e)
            {
                Console.WriteLine("Error: " +
    e.Message.ToString());
            }

        }
     }
}
```

► **To create customer administrator mailboxes using the Exchange Server 2007 Management Shell**

API – Windows PowerShell

Arguments

Remarks –Option: Only needed if configuring hosted Exchange

Sample

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Host;
using System.Management.Automation.Runspaces;

namespace PSCreateMailbox
{
    class CreateMailbox
    {
        static void Main(string[] args)
        {
```

```
            string user = "JohnChen@AlpineSkiHouse";
            string PsText = "Enable-Mailbox";
            string Mailstore = "Mailbox Database";

            RunspaceConfiguration rsConfig =
   RunspaceConfiguration.Create();
            PSSnapInException snapInException = null;
            PSSnapInInfo info =
   rsConfig.AddPSSnapIn("Microsoft.Exchange.Management.Power
   Shell.Admin", out snapInException);

            if (snapInException != null)
            {

   System.Console.Write(snapInException.Message);
                return;
            }

            Runspace myRunSpace =
   RunspaceFactory.CreateRunspace(rsConfig);
            myRunSpace.Open();
            Pipeline pipeLine = myRunSpace.CreatePipeline();

            Command myCommand = new Command(PsText);
            myCommand.Parameters.Add("Identity", user);
            myCommand.Parameters.Add("Database", Mailstore);

            pipeLine.Commands.Add(myCommand);

            Collection<PSObject> commandResults =
   pipeLine.Invoke();
        }
     }
}
```

► **To limit the OWA Address Book search results to Alpine Ski House**

API – ADSI/Windows PowerShell

Arguments

Remarks –Option: Only needed if configuring hosted Exchange

Sample

```
using System;
using System.DirectoryServices;
```

```
namespace Microsoft.CRM40.Samples
{
    class SetUserOWALimit
    {
        static void Main()
        {
            string userDN =
   "CN=JohnChen@AlpineSkiHouse,OU=AlpineSkiHouse,OU=Hosting,
   DC=litwareinc,DC=com";          // Admin User name

            try
            {
                DirectoryEntry user = new
   DirectoryEntry("LDAP://" + userDN);
                string ParentPath =
   user.Parent.Path.ToString();
                string OrgDN = ParentPath.Substring(7,
   ParentPath.Length - 7);
                Console.WriteLine("Parent: " + OrgDN);

   user.Properties["msExchQueryBaseDN"].Add(OrgDN);
                user.CommitChanges();
            }
            catch
   (System.DirectoryServices.DirectoryServicesCOMException
   e)
            {
                Console.WriteLine("Error: " +
   e.Message.ToString());
            }

        }
    }
}
```

► **To configure the user Offline Address Book**

API – ADSI/Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System;
```

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Host;
using System.Management.Automation.Runspaces;

namespace PSModifyMailbox
{
    class ModifyMailbox
    {
        static void Main(string[] args)
        {
            string user = "KimAkers@AlpineSkiHouse";
            string OfflineAddressBook = "AlpineSkiHouse
   OAL";
            string PsText = "Set-Mailbox";

            RunspaceConfiguration rsConfig =
   RunspaceConfiguration.Create();
            PSSnapInException snapInException = null;
            PSSnapInInfo info =
   rsConfig.AddPSSnapIn("Microsoft.Exchange.Management.Power
   Shell.Admin", out snapInException);

            if (snapInException != null)
            {

   System.Console.Write(snapInException.Message);
                return;
            }

            Runspace myRunSpace =
   RunspaceFactory.CreateRunspace(rsConfig);
            myRunSpace.Open();
            Pipeline pipeLine = myRunSpace.CreatePipeline();

            Command myCommand = new Command(PsText);
            myCommand.Parameters.Add("Identity", user);
            myCommand.Parameters.Add("OfflineAddressBook",
   OfflineAddressBook);

   myCommand.Parameters.Add("EmailAddressPolicyEnabled",
   false);
```

```
            pipeLine.Commands.Add(myCommand);

            Collection<PSObject> commandResults =
   pipeLine.Invoke();
        }
     }
}
```

Creating Customer Organization User Accounts

► **To create the customer user account**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx) or Windows PowerShell

Arguments

Remarks

Sample

```
using System;
using System.DirectoryServices;

namespace Microsoft.CRM40.Samples
{
    class CreateAdminUser
    {
        static void Main()
        {

            string OrgName = "AlpineSkiHouse";          //
   Organizational unit.
            string FirstName = "Kim";
            string LastName = "Akers";
            string userPassword = "Pass1word";
            string User = FirstName + LastName + "@" +
   OrgName;
            string UPN = FirstName + LastName.Substring(0,
   1) + "@" + OrgName;
            string SamAccountName = UPN.Replace("@", "_");
            string OrgPath = "LDAP://localhost/OU="+ OrgName
   +",OU=Hosting,DC=litwareinc,DC=com"; // Binding path.

            DirectoryEntry objOrg = new
   DirectoryEntry(OrgPath);
```

```
            // Create Admin User.
            try
            {
                DirectoryEntry objUser =
   objOrg.Children.Add("CN=" + User, "user");

   objUser.Properties["description"].Add(OrgName + "
   Account");

   objUser.Properties["givenName"].Add(FirstName);
                objUser.Properties["sn"].Add(LastName);

   objUser.Properties["userPrincipalName"].Add(UPN);

   objUser.Properties["samAccountName"].Add(SamAccountName);


                objUser.CommitChanges();


                objUser.Invoke("SetPassword", new object[]
   {userPassword});

   objUser.Properties["userAccountControl"].Value = 0x200;
                objUser.CommitChanges();
            }
            catch (Exception e)
            {
                Console.WriteLine("Error:   Create failed
   {0}", e.Message);
            }
        }
    }
}
```

► **To modify the customer user group memberships**

API – System.DirectoryServices (http://msdn2.microsoft.com/en-us/library/system.directoryservices.aspx) or Windows PowerShell

Arguments

Remarks

Sample

```
using System;
using System.DirectoryServices;
```

```
namespace Microsoft.CRM40.Samples
{
    class AddUserToGroup
    {
        static void Main()
        {
            string OrgName = "AlpineSkiHouse";    //
   CompanyName.
            string user = "JohnChen@alpineskihouse";      //
   User name

            string UserDN = "CN=" + user + ",OU=" + OrgName
   + ",OU=Hosting,DC=LitwareInc,DC=com";
            string AllUsersGroupDN = "CN=AllUsers@" +
   OrgName + ",OU=" + OrgName +
   ",OU=Hosting,DC=LitwareInc,DC=com";

            Console.WriteLine(UserDN + "\n" +
   AllusersGroupDN);
            try
            {
                Console.WriteLine("Adding user " + user + "
   to " + OrgName + " All Users Group");
                DirectoryEntry AdminsGroup = new
   DirectoryEntry("LDAP://" + AdminsGroupDN);

   AllUsersGroupDN.Properties["member"].Add(UserDN);
                AllUsersGroupDN.CommitChanges();
            }
            catch
   (System.DirectoryServices.DirectoryServicesCOMException
   e)
            {
                Console.WriteLine("Error: " +
   e.Message.ToString());
            }

        }
    }
}
```

► **To create customer user mailboxes using the Exchange Server 2007 Management Shell**

API – Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Host;
using System.Management.Automation.Runspaces;

namespace PSCreateMailbox
{
    class CreateMailbox
    {
        static void Main(string[] args)
        {
            string user = "KimAkers@AlpineSkiHouse";
            string PsText = "Enable-Mailbox";
            string Mailstore = "Mailbox Database";

            RunspaceConfiguration rsConfig =
  RunspaceConfiguration.Create();
            PSSnapInException snapInException = null;
            PSSnapInInfo info =
  rsConfig.AddPSSnapIn("Microsoft.Exchange.Management.Power
  Shell.Admin", out snapInException);

            if (snapInException != null)
            {

  System.Console.Write(snapInException.Message);
                return;
            }

            Runspace myRunSpace =
  RunspaceFactory.CreateRunspace(rsConfig);
            myRunSpace.Open();
            Pipeline pipeLine = myRunSpace.CreatePipeline();
```

```
            Command myCommand = new Command(PsText);
            myCommand.Parameters.Add("Identity", user);
            myCommand.Parameters.Add("Database", Mailstore);

            pipeLine.Commands.Add(myCommand);

            Collection<PSObject> commandResults =
   pipeLine.Invoke();
        }
     }
}
```

► **To limit the OWA Address Book search results to Alpine Ski House**

API – ADSI/Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System;
using System.DirectoryServices;

namespace Microsoft.CRM40.Samples
{
    class SetUserOWALimit
    {
        static void Main()
        {
            string userDN =
   "CN=KimAkers@AlpineSkiHouse,OU=AlpineSkiHouse,OU=Hosting,
   DC=litwareinc,DC=com";          // Admin User name

            try
            {
                DirectoryEntry user = new
   DirectoryEntry("LDAP://" + userDN);
                string ParentPath =
   user.Parent.Path.ToString();
                string OrgDN = ParentPath.Substring(7,
   ParentPath.Length - 7);
                Console.WriteLine("Parent: " + OrgDN);

   user.Properties["msExchQueryBaseDN"].Add(OrgDN);
```

```
                user.CommitChanges();
            }
            catch
    (System.DirectoryServices.DirectoryServicesCOMException
    e)
            {
                Console.WriteLine("Error: " +
    e.Message.ToString());
            }

        }
     }
}
```

► **To configure the user Offline Address Book**

API – ADSI/Windows PowerShell

Arguments

Remarks – Option: Only needed if configuring hosted Exchange

Sample

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Host;
using System.Management.Automation.Runspaces;

namespace PSModifyMailbox
{
    class ModifyMailbox
    {
        static void Main(string[] args)
        {
            string user = "KimAkers@AlpineSkiHouse";
            string OfflineAddressBook = "AlpineSkiHouse
    OAL";
            string PsText = "Set-Mailbox";

            RunspaceConfiguration rsConfig =
    RunspaceConfiguration.Create();
            PSSnapInException snapInException = null;
```

```
            PSSnapInInfo info =
    rsConfig.AddPSSnapIn("Microsoft.Exchange.Management.Power
    Shell.Admin", out snapInException);

            if (snapInException != null)
            {

    System.Console.Write(snapInException.Message);
                return;
            }

            Runspace myRunSpace =
    RunspaceFactory.CreateRunspace(rsConfig);
            myRunSpace.Open();
            Pipeline pipeLine = myRunSpace.CreatePipeline();

            Command myCommand = new Command(PsText);
            myCommand.Parameters.Add("Identity", user);
            myCommand.Parameters.Add("OfflineAddressBook",
    OfflineAddressBook);

    myCommand.Parameters.Add("EmailAddressPolicyEnabled",
    false);

            pipeLine.Commands.Add(myCommand);

            Collection<PSObject> commandResults =
    pipeLine.Invoke();
        }
     }
}
```

Enabling Customers for CRM

Enabling a Customer Organization for CRM

► **To enable a customer organization for CRM**

API – CRM Web Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
```

```
using System.Text;
using System.Net;
using CreateCRMOrg.CRMDeploymentService;

namespace Microsoft.Crm.Samples
{
    class CreateCRMOrg
    {
        static void Main(string[] args)
        {
            string orgName = "TestOrg10";
            string CRMServer = "CRM01";
            string CRMSQLServer = "CRMSQL";

            CrmDeploymentService service = new
   CrmDeploymentService();
            service.Credentials =
   System.Net.CredentialCache.DefaultCredentials;
            service.Url = "http://" + CRMServer +
   "/MSCRMServices/2007/CrmDeploymentService.asmx";
            service.PreAuthenticate = true;
            service.Timeout = 150000 * 1000;

            Organization newOrg = new Organization();
            newOrg.SqlServerName = CRMSQLServer;
            newOrg.FriendlyName = orgName;
            newOrg.UniqueName = orgName;
            newOrg.BaseCurrencyName = "US Dollar";
            newOrg.BaseCurrencyCode = "USD";
            newOrg.BaseCurrencySymbol = "$";
            newOrg.State = OrganizationState.Enabled;
            newOrg.SrsUrl = "http://"+ CRMSQLServer +
   "/ReportServer";

            CreateRequest create = new CreateRequest();
            create.Entity = newOrg;

            try
            {
                CreateResponse created =
   (CreateResponse)service.Execute(create);
            }
```

```
            catch
    (System.Web.Services.Protocols.SoapException ex)
            {
                Console.WriteLine(ex.Message + "." +
    ex.Detail.InnerText);
            }

        }
     }
}
```

► **To disable a customer organization for CRM**

API – CRM Discovery and Deployment Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using DisableCRMOrg;
using DisableCRMOrg.CrmSdk.Deployment;
using DisableCRMOrg.CrmSdk.Discovery;

namespace Microsoft.Crm.Samples
{
    class DisableCRMOrg
    {
        static void Main(string[] args)
        {
            string orgName = "ManualTest";
            string CRMServer = "CRM01";

            // Instantiate and configure the
    CrmDiscoveryService Web service.
            CrmDiscoveryService discoveryService = new
    CrmDiscoveryService();
            discoveryService.Credentials =
    System.Net.CredentialCache.DefaultCredentials;
            discoveryService.Url = "http://" + CRMServer +
    "/mscrmservices/2007/ad/crmdiscoveryservice.asmx";
            discoveryService.PreAuthenticate = true;
```

```
        // Retrieve the organization name and endpoint
Url from the CrmDiscoveryService Web service.
        RetrieveOrganizationsRequest orgRequest = new
RetrieveOrganizationsRequest();
        RetrieveOrganizationsResponse orgResponse =
(RetrieveOrganizationsResponse)discoveryService.Execute(o
rgRequest);

        OrganizationDetail orgInfo = null;

        foreach (OrganizationDetail orgDetail in
orgResponse.OrganizationDetails)
        {
            if
(orgDetail.OrganizationName.Equals(orgName))
            {
                orgInfo = orgDetail;
                break;
            }
        }

        if (orgInfo == null)throw new Exception("Invalid
organization.");

        // Instantiate and configure the
CrmDeploymentService Web service.
        CrmDeploymentService deploymentService = new
CrmDeploymentService();
        deploymentService.Credentials =
System.Net.CredentialCache.DefaultCredentials;
        deploymentService.Url = "http://" + CRMServer +
"/MSCRMServices/2007/CrmDeploymentService.asmx";
        deploymentService.PreAuthenticate = true;

        // Configure the status of the retrieved
organization to disabled
        SetStateOrganizationRequest setState = new
SetStateOrganizationRequest();
        setState.Id = orgInfo.OrganizationId;
        setState.State = OrganizationState.Disabled;

        try
        {
```

```
                SetStateOrganizationResponse setStateResponse
    =
    (SetStateOrganizationResponse)deploymentService.Execute(s
    etState);
                }
                catch
    (System.Web.Services.Protocols.SoapException ex)
                {
                    Console.WriteLine(ex.Message + "." +
    ex.Detail.InnerText);
                }

            }
        }
}
```

Enabling a Customer Administrator Account for CRM

► **To provision and enable business users for Microsoft Dynamics CRM**

API – CRM Web Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Security;
using CreateCRMUser.CrmSdk;
using CreateCRMUser.CrmSdk.Discovery;

namespace CreateUser
{
    class Program
    {
        static void Main(string[] args)
        {
            string orgName = "AdventureWorks";

            CrmDiscoveryService discoveryService = new
    CrmDiscoveryService();
```

```
            discoveryService.Url =
"http://CRM01/MSCRMServices/2007/AD/CrmDiscoveryService.a
smx";
            discoveryService.Credentials =
System.Net.CredentialCache.DefaultCredentials;

            CrmAuthenticationToken token = new
CrmAuthenticationToken();
            token.OrganizationName = orgName;
            token.AuthenticationType = 0; //AD
authentication

            CrmService crmService = new CrmService();
            crmService.UseDefaultCredentials = true;
            crmService.Url =
"http://CRM01/MSCRMServices/2007/CrmService.asmx";
            crmService.CrmAuthenticationTokenValue = token;

            WhoAmIRequest userRequest = new WhoAmIRequest();
            WhoAmIResponse user =
(WhoAmIResponse)crmService.Execute(userRequest);

            systemuser newUser = new systemuser();
            newUser.domainname = @"FABRIKAM\JohnC";
            newUser.firstname = "John";
            newUser.lastname = "Chen";
            newUser.businessunitid = new Lookup();
            newUser.businessunitid.type =
EntityName.businessunit.ToString();
            newUser.businessunitid.Value =
user.BusinessUnitId;

            TargetCreateSystemUser target = new
TargetCreateSystemUser();
            target.SystemUser = newUser;

            CreateRequest create = new CreateRequest();
            create.Target = target;

            CreateResponse response =
(CreateResponse)crmService.Execute(create);
```

```
            // Assign the user to the System Administrator
Role by first retrieving ID of Role

            QueryByAttribute query = new QueryByAttribute();
            query.ColumnSet = new AllColumns();
            query.EntityName = EntityName.role.ToString();

            // Query will retrieve all account's whose
address1_city is Sammamish
            query.Attributes = new string[] { "name" };
            query.Values = new string[] { "System
Administrator" };

            // Execute the Retrieval
            BusinessEntityCollection retrieved = null;

            try
            {
                retrieved =
crmService.RetrieveMultiple(query);
            }
            catch
(System.Web.Services.Protocols.SoapException ex)
            {
                Console.WriteLine(ex.Message + "." +
ex.Detail.InnerText);
            }
            bool success = false;

            if (retrieved.EntityName.ToLower() ==
EntityName.role.ToString().ToLower())
            {
                success = true;

            }

Console.WriteLine(retrieved.BusinessEntities.Length);

            foreach (role CrmRole in
retrieved.BusinessEntities)
            {
```

```
                AssignUserRolesRoleRequest assign = new
   AssignUserRolesRoleRequest();

                // Set the properties of the request object.
                assign.UserId = response.id;
                assign.RoleIds = new Guid[] {
   CrmRole.roleid.Value };

                try
                {
                    // Execute the request.
                    AssignUserRolesRoleResponse assigned =
   (AssignUserRolesRoleResponse)crmService.Execute(assign);
                }
                catch
   (System.Web.Services.Protocols.SoapException ex)
                {
                    Console.WriteLine(ex.Message + "." +
   ex.Detail.InnerText);
                }
            }

        }
    }
}
```

► **To enable an Alpine Ski House customer administrator account for Microsoft Dynamics CRM**

API – CRM Web Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using EnableUser.CrmSdk;
using EnableUser.CrmSdk.Discovery;

namespace EnableUser
{
    class Program
    {
```

```
        static void Main(string[] args)
        {
            string UserToEnable = @"Fabrikam\JohnC";
            string CRMServer = "CRM01";

            string orgName = "AdventureWorks";

            CrmAuthenticationToken token = new
CrmAuthenticationToken();
            token.OrganizationName = orgName;
            token.AuthenticationType = 0; //AD
authentication

            CrmService crmService = new CrmService();
            crmService.UseDefaultCredentials = true;
            crmService.Url = "http://" + CRMServer +
"/MSCRMServices/2007/CrmService.asmx";
            crmService.CrmAuthenticationTokenValue = token;

            // Query for the user to retrieve its ID
            QueryByAttribute query = new QueryByAttribute();
            query.ColumnSet = new AllColumns();
            query.EntityName =
EntityName.systemuser.ToString();

            // Query will retrieve all account's whose
address1_city is Sammamish
            query.Attributes = new string[] { "domainname"
};
            query.Values = new string[] { UserToEnable };

            // Execute the Retrieval
            BusinessEntityCollection retrieved = null;
            try
            {
                retrieved =
crmService.RetrieveMultiple(query);
            }
            catch
(System.Web.Services.Protocols.SoapException ex)
            {
                Console.WriteLine(ex.Message + "." +
ex.Detail.InnerText);
```

```
            }

            if (retrieved.EntityName.ToLower() ==
    EntityName.systemuser.ToString().ToLower())
            {
                // User Found
            }

            foreach (systemuser user in
    retrieved.BusinessEntities)
            {

                SetStateSystemUserRequest state = new
    SetStateSystemUserRequest();

                state.EntityId = user.systemuserid.Value;
                state.SystemUserState =
    SystemUserState.Active;
                state.SystemUserStatus = 0;

                try
                {
                    SetStateSystemUserResponse stateSet =
    (SetStateSystemUserResponse)crmService.Execute(state);
                }
                catch
    (System.Web.Services.Protocols.SoapException ex)
                {
                    Console.WriteLine(ex.Message + "." +
    ex.Detail.InnerText);
                }

            }
        }
     }
}
```

► **To disable an Alpine Ski House customer administrator account for Microsoft Dynamics CRM**

API – CRM Web Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using DisableUser.CrmSdk;
using DisableUser.CrmSdk.Discovery;

namespace DisableUser
{
    class Program
    {
        static void Main(string[] args)
        {
            string UserToDisable = @"Fabrikam\JohnC";
            string CRMServer = "CRM01";

            string orgName = "AdventureWorks";

            CrmAuthenticationToken token = new
   CrmAuthenticationToken();
            token.OrganizationName = orgName;
            token.AuthenticationType = 0; //AD
   authentication

            CrmService crmService = new CrmService();
            crmService.UseDefaultCredentials = true;
            crmService.Url = "http://" + CRMServer +
   "/MSCRMServices/2007/CrmService.asmx";
            crmService.CrmAuthenticationTokenValue = token;

            // Query for the user to retrieve its ID
            QueryByAttribute query = new QueryByAttribute();
            query.ColumnSet = new AllColumns();
            query.EntityName =
   EntityName.systemuser.ToString();

            // Query will retrieve all account's whose
   address1_city is Sammamish
            query.Attributes = new string[] { "domainname"
   };
            query.Values = new string[] { UserToDisable };
```

```
            // Execute the Retrieval
            BusinessEntityCollection retrieved = null;
            try
            {
                retrieved =
crmService.RetrieveMultiple(query);
            }
            catch
(System.Web.Services.Protocols.SoapException ex)
            {
                Console.WriteLine(ex.Message + "." +
ex.Detail.InnerText);
            }

            if (retrieved.EntityName.ToLower() ==
EntityName.systemuser.ToString().ToLower())
            {
                // User Found
            }

            foreach (systemuser user in
retrieved.BusinessEntities)
            {

                SetStateSystemUserRequest state = new
SetStateSystemUserRequest();

                state.EntityId = user.systemuserid.Value;
                state.SystemUserState =
SystemUserState.Inactive;
                state.SystemUserStatus = -1;

                try
                {
                    SetStateSystemUserResponse stateSet =
(SetStateSystemUserResponse)crmService.Execute(state);
                }
                catch
(System.Web.Services.Protocols.SoapException ex)
                {
                    Console.WriteLine(ex.Message + "." +
ex.Detail.InnerText);
                }
```

```
            }
        }
    }
}
```

Enabling a Customer User Account for CRM

► **To enable an Alpine Ski House customer user account for Microsoft Dynamics CRM**

API – CRM Web Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using EnableUser.CrmSdk;
using EnableUser.CrmSdk.Discovery;

namespace EnableUser
{
    class Program
    {
        static void Main(string[] args)
        {
            string UserToEnable = @"Fabrikam\JohnC";
            string CRMServer = "CRM01";

            string orgName = "AdventureWorks";

            CrmAuthenticationToken token = new
   CrmAuthenticationToken();
            token.OrganizationName = orgName;
            token.AuthenticationType = 0; //AD
   authentication

            CrmService crmService = new CrmService();
            crmService.UseDefaultCredentials = true;
            crmService.Url = "http://" + CRMServer +
   "/MSCRMServices/2007/CrmService.asmx";
```

```
            crmService.CrmAuthenticationTokenValue = token;

            // Query for the user to retrieve its ID
            QueryByAttribute query = new QueryByAttribute();
            query.ColumnSet = new AllColumns();
            query.EntityName =
EntityName.systemuser.ToString();

            // Query will retrieve all account's whose
address1_city is Sammamish
            query.Attributes = new string[] { "domainname"
};
            query.Values = new string[] { UserToEnable };

            // Execute the Retrieval
            BusinessEntityCollection retrieved = null;
            try
            {
                retrieved =
crmService.RetrieveMultiple(query);
            }
            catch
(System.Web.Services.Protocols.SoapException ex)
            {
                Console.WriteLine(ex.Message + "." +
ex.Detail.InnerText);
            }

            if (retrieved.EntityName.ToLower() ==
EntityName.systemuser.ToString().ToLower())
            {
                // User Found
            }

            foreach (systemuser user in
retrieved.BusinessEntities)
            {

                SetStateSystemUserRequest state = new
SetStateSystemUserRequest();

                state.EntityId = user.systemuserid.Value;
```

```
                state.SystemUserState =
    SystemUserState.Active;
                state.SystemUserStatus = 0;


                try
                {
                    SetStateSystemUserResponse stateSet =
    (SetStateSystemUserResponse)crmService.Execute(state);
                }
                catch
    (System.Web.Services.Protocols.SoapException ex)
                {
                    Console.WriteLine(ex.Message + "." +
    ex.Detail.InnerText);
                }


            }
        }
     }
}
```

► **To disable an Alpine Ski House customer user account for Microsoft Dynamics CRM**

API – CRM Web Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using DisableUser.CrmSdk;
using DisableUser.CrmSdk.Discovery;

namespace DisableUser
{
    class Program
    {
        static void Main(string[] args)
        {
            string UserToDisable = @"Fabrikam\JohnC";
```

```
            string CRMServer = "CRM01";

            string orgName = "AdventureWorks";

            CrmAuthenticationToken token = new
CrmAuthenticationToken();
            token.OrganizationName = orgName;
            token.AuthenticationType = 0; //AD
authentication

            CrmService crmService = new CrmService();
            crmService.UseDefaultCredentials = true;
            crmService.Url = "http://" + CRMServer +
"/MSCRMServices/2007/CrmService.asmx";
            crmService.CrmAuthenticationTokenValue = token;

            // Query for the user to retrieve its ID
            QueryByAttribute query = new QueryByAttribute();
            query.ColumnSet = new AllColumns();
            query.EntityName =
EntityName.systemuser.ToString();

            // Query will retrieve all account's whose
address1_city is Sammamish
            query.Attributes = new string[] { "domainname"
};
            query.Values = new string[] { UserToDisable };

            // Execute the Retrieval
            BusinessEntityCollection retrieved = null;
            try
            {
                retrieved =
crmService.RetrieveMultiple(query);
            }
            catch
(System.Web.Services.Protocols.SoapException ex)
            {
                Console.WriteLine(ex.Message + "." +
ex.Detail.InnerText);
            }
```

```
            if (retrieved.EntityName.ToLower() ==
    EntityName.systemuser.ToString().ToLower())
            {
                // User Found
            }

            foreach (systemuser user in
    retrieved.BusinessEntities)
            {

                SetStateSystemUserRequest state = new
    SetStateSystemUserRequest();

                state.EntityId = user.systemuserid.Value;
                state.SystemUserState =
    SystemUserState.Inactive;
                state.SystemUserStatus = -1;

                try
                {
                    SetStateSystemUserResponse stateSet =
    (SetStateSystemUserResponse)crmService.Execute(state);
                }
                catch
    (System.Web.Services.Protocols.SoapException ex)
                {
                    Console.WriteLine(ex.Message + "." +
    ex.Detail.InnerText);
                }

            }
        }
     }
}
```

► **To modify a CRM-enabled user account**

API – CRM Web Service

Arguments

Remarks

Sample

```
using System;
```

```
using System.Collections.Generic;
using System.Text;
using ModifyUser.CrmSdk;

namespace ModifyUser
{
    class Program
    {
        static void Main(string[] args)
        {

            string UserToModify = @"Fabrikam\JohnC";
            string CRMServer = "CRM01";
            string orgName = "AdventureWorks";
            string RoleToAdd = "Marketing Manager";

            CrmAuthenticationToken token = new
   CrmAuthenticationToken();
            token.OrganizationName = orgName;
            token.AuthenticationType = 0; //AD
   authentication

            CrmService crmService = new CrmService();
            crmService.UseDefaultCredentials = true;
            crmService.Url = "http://" + CRMServer +
   "/MSCRMServices/2007/CrmService.asmx";
            crmService.CrmAuthenticationTokenValue = token;

            // Query for the user to retrieve its ID
            QueryByAttribute query = new QueryByAttribute();
            query.ColumnSet = new AllColumns();
            query.EntityName =
   EntityName.systemuser.ToString();

            query.Attributes = new string[] { "domainname"
   };
            query.Values = new string[] { UserToModify };

            // Execute the Retrieval
            BusinessEntityCollection retrieved = null;
            try
            {
```

```
            retrieved =
crmService.RetrieveMultiple(query);
        }
        catch
(System.Web.Services.Protocols.SoapException ex)
        {
            Console.WriteLine(ex.Message + "." +
ex.Detail.InnerText);
        }

        if (retrieved.EntityName.ToLower() ==
EntityName.systemuser.ToString().ToLower())
        {
            // User Found
        }

        systemuser RetrievedUser = null;
        foreach (systemuser user in
retrieved.BusinessEntities)
        {

            RetrievedUser = user;

        }

        // Create the user object.
        systemuser user = new account();

        // Set the properties of the user object to be
updated.
        user.address1_line = "34 Market St.";
        user.jobtitle = "Finance Controller";

        // systemuserid is a key that references the ID
of the user to be updated.
        user.systemuserid = new Key();

        // systemuserid.Value is the GUID of the user to
be changed.
        user.systemuserid.Value =
RetrievedUser.systemuserid.Value;

        // Create the target object for the request.
```

```
            TargetUpdateSystemUser target = new
   TargetUpdateSystemUser ();

            // Set the properties of the target object.
            target.SystemUser = user;

            // Create the request object.
            UpdateRequest update = new UpdateRequest();

            // Set the properties of the request object.
            update.Target = target;

            // Execute the request.
            UpdateResponse updated =
   (UpdateResponse)service.Execute(update);



            }
        }
    }
}
```

Configuring the CRM E-mail Router

► **To configure mail forwarding rules**

API – Rules.exe executable

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;

namespace AddMailboxRules
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            /*parameters[0] = command;
<- /add, /delete, or /verify the rule
            parameters[1] = exchangeServerName;
<- Exchange server name.
            parameters[2] = exchangeStoreDn;
<- Exchange store legacy DN.
            parameters[3] = exchangeSystemDn;
<- Exchange system attendant DN.
            parameters[4] = exchangeMailboxDn;
<- Exchange mailbox legacy DN.
            parameters[5] = crmEmailAddress;
<- target email address
            parameters[6] = disabledMode ? "1" : "0";
<- create the rule as disabled*/

            StringBuilder sb = new StringBuilder();
            // Add required parameters to Stringbuilder


            // Start the process
            Process process = new Process();
            using (process)
            {
                process.StartInfo.UseShellExecute = false;
                process.StartInfo.RedirectStandardOutput =
true;
                process.StartInfo.FileName = "Rules.exe";
                process.StartInfo.Arguments = sb.ToString();
                process.Start();
                string output =
process.StandardOutput.ReadToEnd();
                process.WaitForExit();

                // Collect results
                int exitCode = process.ExitCode;
                if (0 != exitCode)     //<- failure
                {
                    LAST_ACTION lastAction =
LAST_ACTION.UNKNOWN_ACTION;
                    if (!String.IsNullOrEmpty(output))
                    {
                        lastAction =
(LAST_ACTION)Enum.Parse(typeof(LAST_ACTION), output);
```

```
                    }     //<- see how far we could go
                    throw new Exception("Error from
Rules.exe - Exit code = " + exitCode);
                }
                //ruleStatus =
(RULE_STATUS)Enum.Parse(typeof(RULE_STATUS), output);
//<- success
            }
        }

        /// <summary>
        /// Rule status.
        /// </summary>
        private enum RULE_STATUS : int
        {
            RULE_NOT_FOUND = 0,
            RULE_FOUND_DISABLED = 1,
            RULE_FOUND_ENABLED = 2,
            RULE_ADDED = 3,
        };

        /// <summary>
        /// Last action.
        /// </summary>
        private enum LAST_ACTION : int
        {
            UNKNOWN_ACTION = 0,
            MAPI_INIT = 1,
            PROFILE_CREATE = 2,
            PROFILE_CONFIGURE = 3,
            PROFILE_LOG = 4,
            STORES_ACCESS = 5,
            STORE_ACCESS = 6,
            STORE_PRIV_OBTAIN = 7,
            STORE_PRIV_ACCESS = 8,
            INBOX_ACCESS = 9,
            RULES_ACCESS = 10,
            RULES_LOAD = 11,
            RULES_CREATE = 12,
            RULES_SAVE = 13,
            RULES_REMOVE = 14,
```

```
            RULES_EXAMINE = 15,
        };

    }
}
```

► **To remove mail forwarding rules**

API – Rules.exe executable

Arguments

Remarks – Server side rule

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;

namespace AddMailboxRules
{
    class Program
    {
        static void Main(string[] args)
        {
            /*parameters[0] = command;
   <- /add, /delete, or /verify the rule
            parameters[1] = exchangeServerName;
   <- Exchange server name.
            parameters[2] = exchangeStoreDn;
   <- Exchange store legacy DN.
            parameters[3] = exchangeSystemDn;
   <- Exchange system attendant DN.
            parameters[4] = exchangeMailboxDn;
   <- Exchange mailbox legacy DN.
            parameters[5] = crmEmailAddress;
   <- target email address
            parameters[6] = disabledMode ? "1" : "0";
   <- create the rule as disabled*/

            StringBuilder sb = new StringBuilder();
            // Add required parameters to Stringbuilder


            // Start the process
```

```
            Process process = new Process();
            using (process)
            {
                process.StartInfo.UseShellExecute = false;
                process.StartInfo.RedirectStandardOutput =
true;
                process.StartInfo.FileName = "Rules.exe";
                process.StartInfo.Arguments = sb.ToString();
                process.Start();
                string output =
process.StandardOutput.ReadToEnd();
                process.WaitForExit();

                // Collect results
                int exitCode = process.ExitCode;
                if (0 != exitCode)     //<- failure
                {
                    LAST_ACTION lastAction =
LAST_ACTION.UNKNOWN_ACTION;
                    if (!String.IsNullOrEmpty(output))
                    {
                        lastAction =
(LAST_ACTION)Enum.Parse(typeof(LAST_ACTION), output);
                    }     //<- see how far we could go
                    throw new Exception("Error from
Rules.exe - Exit code = " + exitCode);
                }
                //ruleStatus =
(RULE_STATUS)Enum.Parse(typeof(RULE_STATUS), output);
//<- success
            }
        }

        /// <summary>
        /// Rule status.
        /// </summary>
        private enum RULE_STATUS : int
        {
            RULE_NOT_FOUND = 0,
            RULE_FOUND_DISABLED = 1,
            RULE_FOUND_ENABLED = 2,
            RULE_ADDED = 3,
```

```
        };

        /// <summary>
        /// Last action.
        /// </summary>
        private enum LAST_ACTION : int
        {
            UNKNOWN_ACTION = 0,
            MAPI_INIT = 1,
            PROFILE_CREATE = 2,
            PROFILE_CONFIGURE = 3,
            PROFILE_LOG = 4,
            STORES_ACCESS = 5,
            STORE_ACCESS = 6,
            STORE_PRIV_OBTAIN = 7,
            STORE_PRIV_ACCESS = 8,
            INBOX_ACCESS = 9,
            RULES_ACCESS = 10,
            RULES_LOAD = 11,
            RULES_CREATE = 12,
            RULES_SAVE = 13,
            RULES_REMOVE = 14,
            RULES_EXAMINE = 15,
        };

    }
}
```

► **To query CRM customer organizations**

API – CRM Discovery Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using QueryAllOrgs.CrmSdk.Discovery;

namespace QueryAllOrgs
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            string CRMServer = "CRM01";

            CrmDiscoveryService discoveryService = new
   CrmDiscoveryService();
            discoveryService.Credentials =
   System.Net.CredentialCache.DefaultCredentials;
            discoveryService.Url = "http://" + CRMServer +
   "/mscrmservices/2007/ad/crmdiscoveryservice.asmx";
            discoveryService.PreAuthenticate = true;



             // STEP 2: Retrieve the organization name and
   endpoint Url from the
            // CrmDiscoveryService Web service.

            RetrieveOrganizationsRequest orgRequest = new
   RetrieveOrganizationsRequest();
            RetrieveOrganizationsResponse orgResponse =
   (RetrieveOrganizationsResponse)discoveryService.Execute(o
   rgRequest);

            foreach (OrganizationDetail orgDetail in
   orgResponse.OrganizationDetails)
            {

   Console.WriteLine(orgDetail.OrganizationName);
            }
        }
    }
}
```

► **To add and remove users from roles**

API – CRM Web Service

Arguments

Remarks

Sample

```
using System;
using System.Collections.Generic;
using System.Text;
using ModifyUserRoles.CrmSdk;

namespace ModifyUserRoles
{
    class Program
    {
        static void Main(string[] args)
        {

            string UserToModify = @"Fabrikam\Daveb2";
            string CRMServer = "CRM01";
            string orgName = "AdventureWorks";
            string RoleToAdd = "Marketing Manager";

            CrmAuthenticationToken token = new
   CrmAuthenticationToken();
            token.OrganizationName = orgName;
            token.AuthenticationType = 0; //AD
   authentication

            CrmService crmService = new CrmService();
            crmService.UseDefaultCredentials = true;
            crmService.Url = "http://" + CRMServer +
   "/MSCRMServices/2007/CrmService.asmx";
            crmService.CrmAuthenticationTokenValue = token;

            // Query for the user to retrieve its ID
            QueryByAttribute query = new QueryByAttribute();
            query.ColumnSet = new AllColumns();
            query.EntityName =
   EntityName.systemuser.ToString();

            query.Attributes = new string[] { "domainname"
   };
            query.Values = new string[] { UserToModify };

            // Execute the Retrieval
            BusinessEntityCollection retrieved = null;
            try
```

```
            {
                retrieved =
crmService.RetrieveMultiple(query);
            }
            catch
(System.Web.Services.Protocols.SoapException ex)
            {
                Console.WriteLine(ex.Message + "." +
ex.Detail.InnerText);
            }

            if (retrieved.EntityName.ToLower() ==
EntityName.systemuser.ToString().ToLower())
            {
                // User Found
            }

            systemuser RetrievedUser = null;
            foreach (systemuser user in
retrieved.BusinessEntities)
            {

                RetrievedUser = user;

            }

            // Create the user object.
            systemuser user = new account();

            // Set the properties of the user object to be
updated.
            user.address1_line = "34 Market St.";
            user.jobtitle = "Finance Controller";

            // systemuserid is a key that references the ID
of the user to be updated.
            user.systemuserid = new Key();

            // systemuserid.Value is the GUID of the user to
be changed.
            user.systemuserid.Value =
RetrievedUser.systemuserid.Value;
```

```
            // Create the target object for the request.
            TargetUpdateAccount target = new
TargetUpdateAccount();

            // Set the properties of the target object.
            target.Account = user;

            // Create the request object.
            UpdateRequest update = new UpdateRequest();

            // Set the properties of the request object.
            update.Target = target;

            // Execute the request.
            UpdateResponse updated =
(UpdateResponse)service.Execute(update);



            // Assign the user to the System Administrator
Role by first retrieving ID of Role
            QueryByAttribute roleQuery = new
QueryByAttribute();
            roleQuery.ColumnSet = new AllColumns();
            roleQuery.EntityName =
EntityName.role.ToString();

            // Query will retrieve all account's whose
address1_city is Sammamish
            roleQuery.Attributes = new string[] { "name" };
            roleQuery.Values = new string[] { RoleToAdd };

            // Execute the Retrieval
            BusinessEntityCollection roleRetrieved = null;

            try
            {
                roleRetrieved =
crmService.RetrieveMultiple(query);
            }
            catch
(System.Web.Services.Protocols.SoapException ex)
            {
```

```
                Console.WriteLine(ex.Message + "." +
    ex.Detail.InnerText);
            }
            bool success = false;


            if (roleRetrieved.EntityName.ToLower() ==
    EntityName.role.ToString().ToLower())
            {
                success = true;


            }


            foreach (role CrmRole in
    roleRetrieved.BusinessEntities)
            {
                AssignUserRolesRoleRequest assign = new
    AssignUserRolesRoleRequest();


                // Set the properties of the request object.
                assign.UserId =
    RetrievedUser.systemuserid.Value;
                assign.RoleIds = new Guid[] {
    CrmRole.roleid.Value };


                try
                {
                    // Execute the request.
                    AssignUserRolesRoleResponse assigned =
    (AssignUserRolesRoleResponse)crmService.Execute(assign);
                }
                catch
    (System.Web.Services.Protocols.SoapException ex)
                {
                    Console.WriteLine(ex.Message + "." +
    ex.Detail.InnerText);
                }
            }
        }
    }
}
```

Additional Resources

Web Sites

- Microsoft Visual C#® Express Edition - http://msdn2.microsoft.com/en-us/express/aa700756.aspx
- Web Services and Other Distributed Technologies - http://msdn2.microsoft.com/webservices/default.aspx
- Windows PowerShell SDK - http://msdn2.microsoft.com/en-us/library/ms714469.aspx

Developing with Visual C#

Microsoft Visual C# 2005 is the modern, innovative programming language and tool for building .NET-connected software for Microsoft Windows®, the Web, and a wide range of devices. With syntax that resembles C++, a flexible integrated development environment (IDE), and the capability to build solutions across a variety of platforms and devices, Visual C# 2005 significantly eases the development of .NET-connected software.

For more information about Visual C#, including case studies, news and reviews, and system requirements, visit the Visual C# Developer Center.

XML Document Object Model

The XML Document Object Model (DOM) is used to work with in-memory XML documents. This information is needed for using many of the methods found in the Microsoft Dynamics CRM application programming interface (API). This Web site is located at http://msdn2.microsoft.com/en-us/library/hf9hbf87.aspx.

Active Directory

Active Directory® is the Microsoft Windows operating system directory service that is designed for distributed computing environments. For more information, visit the Active Directory Web site at Microsoft Windows Server 2003 Active Directory.