Deploying Microsoft Dynamics CRM 2011 and Microsoft Dynamics CRM Online solutions from development through test and production environments

Microsoft Corporation

Published October 2011

Updated September 2013

Abstract

This paper focuses on approaches to managing the post-development testing and deployment of business solutions built on Microsoft Dynamics CRM 2011 or Microsoft Dynamics CRM Online through test, staging, and production environments.

Important

While different aspects of the information in this paper apply to hosted deployments of Microsoft Dynamics CRM 2011, for purposes of this paper, references to Microsoft Dynamics CRM 2011 herein refer exclusively to on-premises implementations of the product.

Contents

Deploying Microsoft Dynamics CRM 2011 and Microsoft Dynamics CRM Online solutions from development through test and production environments

Contributors: Darren Hubert, Darren Liu, and Joey Su (MCS, US); Roger Gilchrist and Phil Hand, (MCS, UK); Ryan Casey (Zero2Ten CRM); and Riaan Van Der Merwe (Neudesic)

Reviewers: Microsoft Dynamics CRM Product Development

Published: October 2011 Updated: September 2013

Applies To

- Microsoft Dynamics CRM 2011
- Microsoft Dynamics CRM Online

In this White Paper

Introduction

This section introduces the purpose, scope, and applicability of the information provided in this paper.

Scope and Purpose

This white paper explains considerations and best practices for managing post-development testing and deployment of business solutions built on Microsoft Dynamics CRM 2011 or Microsoft Dynamics CRM Online through test, staging, and production environments following guidelines specified in the Software Development Lifecycle.

Important

While aspects of the information in this paper apply to hosted deployments of Microsoft Dynamics CRM Online, for purposes of this paper, references to Microsoft Dynamics CRM 2011 herein refer exclusively to on-premises implementations of the product.

Note
This paper complements rather than replaces existing support resources that are available for Microsoft Dynamics CRM 2011 and Microsoft Dynamics CRM Online.

Applicability

When considering the applicability of the information in this white paper to any specific solution based on Microsoft Dynamics CRM 2011 or Microsoft Dynamics CRM Online, note that the techniques and guidance provided in this paper can yield varying results depending on a wide range of potential environmental factors, for example the level and complexity of the customizations applied to a particular solution.

Be sure to verify the functionality and performance impact associated with any specific deployment technique in a test environment prior to making any change to the production environment.

Important
In addition, for clients that are connected to a Microsoft Dynamics CRM solution that is integrated with other systems, be sure to verify deployment techniques in a test environment that approximates the complexity and integration that is present in the production environment.

Download

This paper can be downloaded from the Microsoft Download Center: Deploying Microsoft Dynamics CRM solutions from development through test and production environments.

Value proposition: Extending the application

A core value proposition of Microsoft Dynamics CRM is the ability to extend the application to meet the functional needs of a specific organization.

Solutions

Microsoft Dynamics CRM 2011 and the latest version of Microsoft Dynamics CRM Online introduce the concept of “solutions” to provide a simple and reliable way to deploy packages of related extensions to Microsoft Dynamics CRM implementations.

For more information, in the Microsoft Dynamics CRM SDK, see the topic Package and Distribute Extensions with Microsoft Dynamics CRM Solutions.

In professionally managed development organizations, these solutions will be deployed into various test and staging environments for analysis before they are released or deployed into a production environment.

This white paper explores deploying real-world Microsoft Dynamics CRM solutions across test and production environments in reliable and repeatable ways by using automation, together with some automation examples. The paper also highlights the specific constraints that exist when deploying and testing solutions in the Microsoft Dynamics CRM Online environment.

Solution Design and Development

A development team typically uses one or more Microsoft Dynamics CRM environments in which to develop and test its solution. In addition, assembling the various solution components (which are exported unmanaged solutions) that are delivered by supporting feature teams typically requires a separate Microsoft Dynamics CRM organization. In the end, a single managed solution is “built” (exported) from this integration environment.

This white paper focuses on deploying a managed solution and its accompanying dependencies into test, staging, and production environments. The paper also covers the impact of specific solution design choices on the ability to deploy the solution in a production environment.

Important

While outside the scope of this paper, solution design and the design of dependencies between solutions can also have a major impact on a how a solution is deployed and tested. For more information, in the Microsoft Dynamics CRM SDK, see the topic Compartmentalize Your Solutions.

Maintaining test environments

Approaches to maintaining test environments differ based on whether the testing is performed in an on-premises environment using Microsoft Dynamics CRM 2011 or in an online environment using Microsoft Dynamics CRM Online. Specifically, the approaches that are available for “resetting” an environment to a known state differ because an administrator of an on-premises deployment has the ability to access the database or the underlying server directly, while administrators of Microsoft Dynamics CRM Online implementations have only limited access to the database and server.

Testing Microsoft Dynamics CRM 2011 on-premises deployments

A primary method for delivering reproducible, “known state” environments is to use virtualization and “snapshot” technology to capture a baseline environment and then to revert to that baseline at appropriate times during the testing process. The known state environment might include a default installation (without solution components) and functionally tested instances of various versions of the solution (including test data) to support upgrade testing.

Because Microsoft Dynamics CRM is a database application, taking database backups of various environments can also allow the restoration of various data configurations. In addition, backups of production organization databases can be imported into test Microsoft Dynamics CRM deployments to allow for production upgrade scenario testing.

Because testing on-premises solutions can offer more flexibility, easier debugging, and so on over testing online solutions, organizations may opt to perform some on-premises testing even if the solution ultimately will be deployed to a Microsoft Dynamics CRM Online production environment. In these cases, multiple test environments are often created to allow for testing on-premises as well as online.

Testing Microsoft Dynamics CRM Online deployments

When testing Microsoft Dynamics CRM Online environments, some important constraints affect the choice of an approach. First, and perhaps most importantly, Microsoft Dynamics CRM Online administrators cannot initiate explicit point-in-time back-ups or to allow users to "reset" their systems to a previous or initial state. As a result, managing the state of a particular instance is accomplished by using the application itself (either through the user interface or through various APIs), primarily by deleting managed solutions and by using the Bulk Record Deletion feature.

Managing configuration

Managed solutions play an especially important role in Microsoft Dynamics CRM Online because they provide a clean and comprehensive "uninstall" capability that removes all customizations and data specifically associated with a solution. When deleting unmanaged solutions, on the other hand, all components of the unmanaged solution remain in the system and become part of the underlying "default" solution, and they cannot be removed from the system. As a result, after an unmanaged solution has been deleted, administrators must remove solution elements manually, one at a time, without the benefit of tools to help distinguish the custom elements from built-in elements.

For more information, in the Microsoft Dynamics CRM SDK, see the topic Uninstall or Delete a Solution.

Managing data

The Bulk Record Deletion feature provides the ability to specify query criteria for a set of records and to submit an asynchronous job to delete those records, a process that can be initiated by using the application UI or through APIs. As a result, you can use Bulk Delete functionality to develop environment cleaning tools that will delete test data from the system.

Note
For more information, in the Microsoft Dynamics CRM SDK, see the topic Delete Data in Bulk in Microsoft Dynamics CRM.

You can also use the Microsoft Dynamics CRM Data Import Wizard to delete imported data sets quickly and efficiently.

Using Microsoft Dynamics CRM Online trials

Given the challenge of resetting Microsoft Dynamics CRM Online instances, many organizations have adopted the approach of using new Microsoft Dynamics CRM Online trial organizations as test environments. However, this approach has constraints, as listed in the following table:

Dimension	Constraint
Trial Length	30 days
Maximum # of users	25 interactive users, 5 API users
Maximum Storage	5 GB
Customizations	300 custom entities, 200 workflows

Important

Setting up each Microsoft Dynamics CRM Online trial organization requires establishing a new subscription by using a unique identity.

Managing solution component dependencies

The Microsoft Dynamics CRM solution model contains a variety of components, including:

- Entity customizations/custom entities (including schema, views and forms)
- Global option sets
- SiteMap and Ribbon customizations
- Web resources (HTML & CSS, Images & Icons, Jscript, XML & XSL, Silverlight packages)
- Custom plug-in/workflow assemblies and associated registration data
- Windows Azure ServiceBus event registrations
- Dashboards and Charts
- Report definitions
- Connection role definitions
- Templates for articles, contracts, email and Mail Merge
- Security role definitions
- Field Level Security profiles

However, organizations that leverage these components during solution development often build business processes that depend on elements external to the solution model, such as:

- Users and teams
- Queues
- Configuration data
- Solution reference data (including Lists of Values (LOVs))

Dependencies on elements external to a solution model require special consideration for successful deployment, often while building a custom automated solution deployment tool.

Note

Solutions can be deployed by using the Solution Management UI or through Organization Service API calls. For additional information, in the Microsoft Dynamics CRM SDK, see the topic Work with Solutions.

Addressing execution identity issues

Plug-ins and workflows each have explicit identity models that control access to system data.

Plug-ins

By default, plug-ins execute in the security context of the calling CRM user, meaning that any additional calls made by these plug-ins to the Microsoft Dynamics CRM Web services will be constrained by the access rights of the original CRM user. As a result, the plug-in can make changes only to Microsoft Dynamics CRM data that the original CRM user would have been able to make directly.

Note
"CRM user" here refers to a user that is defined in Microsoft Dynamics CRM (regardless of the user's means of authentication) and not necessarily to the Windows domain identity of the user or the processes on the Microsoft Dynamics CRM server.

Plug-ins can also be registered to execute as a specific CRM user in the system, which allows a plug-in to take actions that the calling user might not have permission to take. When plug-ins that are registered to run as a specific CRM user are packaged as part of a solution, the Full Name attribute of the specified user is captured in the solution package. When that solution is imported, the plug-in is registered to execute as the user with a matching Full Name.

If a user with a matching name is not located, the plug-in is registered to execute in the context of the calling CRM user. For this reason, take extra care to ensure that the appropriate users have been created before you attempt to import a solution.

Workflows

Workflows have an execution identity model similar to that of plug-ins. Workflows that are executed on-demand run in the context of the calling CRM user and the actions they can take are limited to the system rights of that user. However, workflows configured to execute automatically in response to system events (such as record creations) execute in the context of the user that published the workflow.

Workflows and plug-ins behave differently during solution import. When importing a solution that contains automatic workflows, the workflows will be "owned" by the user performing the import. If the importing user chooses to activate workflows as part of the import, those automatic workflows will be configured to execute in the context of the importing user regardless of what user they were configured to run as on the exporting system.

To enable specific workflow identity, the administrator must assign the workflows to the appropriate users, and then the new workflow owners must "publish" the workflows. This can be automated by using the Organization API and impersonation.

Note that if workflows in a managed solution are reassigned and published by other users, administrators will see errors when trying to delete that managed solution. All workflows not owned by the user deleting the solution must be un-published to allow deletion of the managed solution.

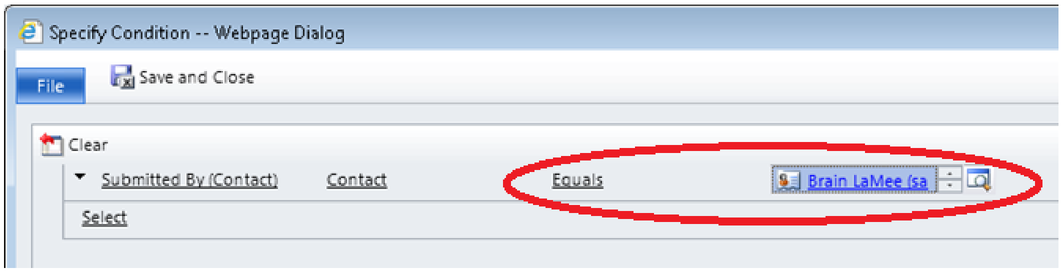
For more information, in the Microsoft Dynamics CRM SDK, see the topic Impersonate Another User.

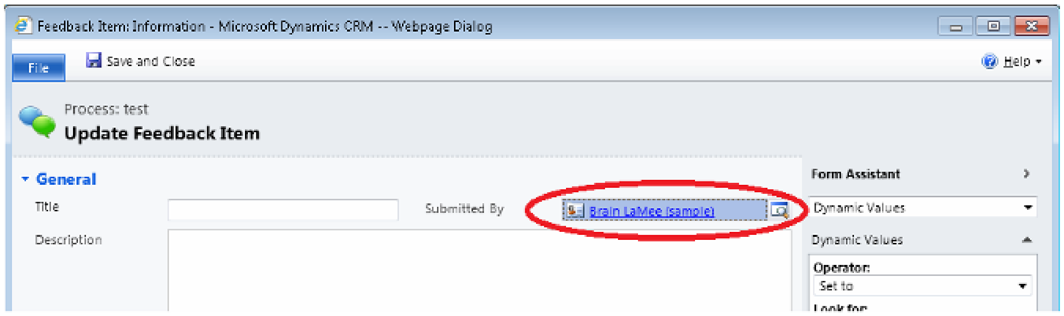
The following code example demonstrates reassigning a workflow and publishing it as a specific user identity.

```
public static void AssignAndPublishWorkflow(ServerConnection.Configuration serverConfig,
      Guid workflowid, Guid userid)
{
    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
serverConfig.OrganizationUri,
                                                            serverConfig.HomeRealmUri,
                                                            serverConfig.Credentials,

serverConfig.DeviceCredentials))
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        //unpublish the workflow
        SetStateRequest unpubReq = new SetStateRequest();
        unpubReq.EntityMoniker = new EntityReference("workflow", workflowid);
        unpubReq.State = new OptionSetValue(0); //draft state
        unpubReq.Status = new OptionSetValue(1); //draft status
        SetStateResponse unpubResp = (SetStateResponse)service.Execute(unpubReq);

        //assign the workflow to the new userid
        AssignRequest assignReq = new AssignRequest();
        assignReq.Target = new EntityReference("workflow", workflowid);
        assignReq.Assignee = new EntityReference("systemuser", userid);
        AssignResponse assignResp = (AssignResponse)service.Execute(assignReq);

        //impersonate the new userid
        serviceProxy.CallerId = userid;

        //publish the workflow
        SetStateRequest pubReq = new SetStateRequest();
        pubReq.EntityMoniker = new EntityReference("workflow", workflowid);
        pubReq.State = new OptionSetValue(1); //published state
        pubReq.Status = new OptionSetValue(2); //published status
        SetStateResponse pubResp = (SetStateResponse)service.Execute(pubReq);

    }
}
```

Addressing workflow reference issues

The Microsoft Dynamics CRM workflow designer allows for building conditional expressions and assignments that result in references to specific records, as shown in the following examples.





Because these records contain IDs that are unique to the system on which the workflows were developed, in general it is best to avoid using these types of workflows in solutions intended for deployment across multiple organizations, each of which would have unique record IDs. When specifying workflow conditions, for instance, an alternative approach is to use a custom record identifier that is defined as part of the solution.

For situations in which this approach is required (for instance to represent industry reference data or taxonomy or for solution specific queue arrangements), be sure to import this solution data with the record ID values intact so that references in workflows will continue to function as designed. Note that while importing data through bulk import does not allow assigning record IDs, it does provide for assigning record IDs programmatically by using the Organization Service's Create method.

For more information, in the Microsoft Dynamics CRM SDK, see the topic IOrganizationService.Create Method.

Important

While the workflow designer can also be used to create references to specific business units, teams, and users, because these aspects of a system are typically beyond the scope of any particular solution, avoid such references in solutions that are designed for use across different organizations.

Automating the creation of users, teams, and business units

Testing Microsoft Dynamics CRM solutions often requires multiple user and team definitions to validate scenarios across functional and security roles, organizational boundaries, and so on. In test environments, this usually means automating the provisioning of users, business units, and teams to support various test cases.

Creating business units and teams

Automating the creation of business units and teams is a relatively straight forward process using the Create method on the Organization web service. The following examples demonstrate the creation of test business units and teams.

Creating business units

When considering population of business units, remember that while Microsoft Dynamics CRM allows for creation of security roles associated with subordinate business units, any security roles that are defined and imported as part of a solution will always be associated with the root business unit. The primary challenge is creating business unit records in the correct order to allow for dependencies between records (such as parent business units, and so on). In addition, to delete these records, all dependent records must be deleted first. In addition, to delete a business unit, you must first disable it.

```
public static Guid CreateBU(ServerConnection.Configuration serverConfig,
  Guid id, string name, Guid parentBU)
{
    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
 serverConfig.OrganizationUri,
                                                           serverConfig.HomeRealmUri,
                                                           serverConfig.Credentials,

serverConfig.DeviceCredentials))
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        EntityReference buref = new EntityReference("businessunit", parentBU);

        Entity newBU = new Entity("businessunit");
        newBU.Id = id;
        newBU["name"] = name;
        newBU["parentbusinessunitid"] = buref;
        Guid newbuid = service.Create(newBU);
        return newbuid;
    }
}
```

Creating teams

```
public static Guid CreateTeam(ServerConnection.Configuration serverConfig,
    Guid id, string name, Guid parentBU, Guid adminUser)
{
    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
 serverConfig.OrganizationUri,
                                                              serverConfig.HomeRealmUri,
                                                              serverConfig.Credentials,

serverConfig.DeviceCredentials))
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        EntityReference buref = new EntityReference("businessunit", parentBU);
        EntityReference adminref = new EntityReference("systemuser", adminUser);

        Entity newTeam = new Entity("team");
        newTeam.Id = id;
        newTeam["name"] = name;
        newTeam["businessunitid"] = buref;
        newTeam["administratorid"] = adminref;

        Guid newteamid = service.Create(newTeam);

        return newteamid;
    }
}
```

Creating users

Creating users presents different challenges with different considerations than does creating business units and teams. For on-premises environments, creating users requires providing the user names and domain names of users that exist in the target environment. Creating users in Microsoft Dynamics CRM Online, however, requires providing existing Microsoft accounts.

Note
Microsoft Dynamics CRM Online does not currently support sending invitations to new users through the web service APIs. As a result, this must be performed manually by using the application UI.

Important
Administrators can only disable CRM users, not delete them. As a result, subsequent attempts to create existing users, even if those user records are disabled, will fail.

The following examples demonstrate the creation and assignment of a user to a security role.

Create user – on-premises

```
public static Guid CreateUserOnPremise(ServerConnection.Configuration serverConfig,
    Guid id, Guid buid, string username,
    string lastname, string firstname)
{
    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
 serverConfig.OrganizationUri,
                                                           serverConfig.HomeRealmUri,
                                                           serverConfig.Credentials,

serverConfig.DeviceCredentials))
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        EntityReference buref = new EntityReference("businessunit", buid);

        Entity newUser = new Entity("systemuser");
        newUser.Id = id;
        newUser["domainname"] = username;
        newUser["lastname"] = lastname;
        newUser["firstname"] = firstname;
        newUser["businessunitid"] = buref;

        Guid userid = service.Create(newUser);

        return userid;
    }
}
```

Create user - online

```
public static Guid CreateUserOnline(ServerConnection.Configuration serverConfig,
    Guid id, Guid buid, string liveid,
    string lastname, string firstname)
{
    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
 serverConfig.OrganizationUri,
                                                            serverConfig.HomeRealmUri,
                                                            serverConfig.Credentials,

serverConfig.DeviceCredentials))
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        EntityReference buref = new EntityReference("businessunit", buid);

        Entity newUser = new Entity("systemuser");
        newUser.Id = id;
        newUser["internalemailaddress"] = liveid;
        newUser["windowsliveid"] = liveid;
        newUser["lastname"] = lastname;
        newUser["firstname"] = firstname;
        newUser["businessunitid"] = buref;
        Guid userid = service.Create(newUser);

        return userid;
    }
}
```

Assign user

```
public static void AssignUserToRole(ServerConnection.Configuration serverConfig,
   Guid userid, Guid roleid)
{
    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
 serverConfig.OrganizationUri,
                                                           serverConfig.HomeRealmUri,
                                                           serverConfig.Credentials,

serverConfig.DeviceCredentials))
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        EntityReferenceCollection relatedEntities = new EntityReferenceCollection();
        relatedEntities.Add(new EntityReference("role", roleid));

        Relationship relationship = new Relationship("systemuserroles_association");

        AssociateRequest req = new AssociateRequest();
        req.Target = new EntityReference("systemuser", userid);
        req.RelatedEntities = relatedEntities;
        req.Relationship = relationship;

        AssociateResponse response = (AssociateResponse)service.Execute(req);
    }
}
```

Automating the creation of queues

Designated queues can be an important part of a solution, especially for solutions that contain workflows with references to these specific queues. As a result, it is often necessary to create queues as part of an automated solution deployment process.

The following example demonstrates the creation of a queue.

```
public static Guid CreateQueue(ServerConnection.Configuration serverConfig,
     Guid id, string name)
{
    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
 serverConfig.OrganizationUri,
                                                           serverConfig.HomeRealmUri,
                                                           serverConfig.Credentials,

serverConfig.DeviceCredentials))
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        Entity newQ = new Entity("queue");
        newQ.Id = id;
        newQ["name"] = name;

        Guid newqueueid = service.Create(newQ);

        return newqueueid;
    }
}
```

Preparing configuration data

Most solutions require some kind of environment-specific data that is adjusted to fit the business process or infrastructure of a particular deployment. Approaches to enabling solution configuration in Microsoft Dynamics CRM can vary, but two common approaches involve using custom entities or using XML web resources. The advantages and disadvantages associated with each approach are shown in the following table.

Approach	Advantages	Disadvantages
Using web resources	• Automatically deployed as part of a solution	• Less detailed security model (all users can read web resources) • Additional coding required to parse a custom XML file • Configuration can be overwritten by solution upgrade
Using custom entities	• Robust security model • Can use built-in forms environment for editing configuration values	• Must import initial configuration values into the system after the initial solution import

Depending on the approach used in the solution, looking up configuration records may or may not be dependent on specific record ID values. If the approach is to locate configuration values based on record IDs, that data must by imported using the Organization service's Create method, which allows setting each record's unique ID.

Configuring a workflow process can be a challenge since the execution context of a workflow process only provides access to records directly related to the target record. Typical approaches that leverage custom workflow activities to read configuration data are not available today in the Microsoft Dynamics CRM Online environment.

Preparing solution reference data

For scenarios in which data is part of the solution, it is important for that data to be imported into the target system in conjunction with importing the solution. There are many approaches to importing data into Microsoft Dynamics CRM, but to support scenarios where workflows will contain references to specific records in these data sets, the data must be imported using the Organization Service's Create method to preserve the unique IDs of records.

In situations like this, care must be taken to build appropriate ownership and security models around reference data so that users don't accidentally modify solution behavior by changing relationships in the data or by adding records. When upgrading a solution, a strategy must be identified to deal with updates to this reference data. How will deployment-specific changes to reference data be handled? How will records that were added to the system be handled?

The following example demonstrates a generic approach to exporting reference data and importing it into a Microsoft Dynamics CRM organization after the solution has been imported.

```
public static void SaveData(ServerConnection.Configuration serverConfig,

  string fetchXml, string filename)

{

    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
 serverConfig.OrganizationUri,
                                                            serverConfig.HomeRealmUri,
                                                            serverConfig.Credentials,

serverConfig.DeviceCredentials))
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        FetchExpression query = new FetchExpression(fetchXml); //hint: use advanced find
to
     //build fetchXml

        EntityCollection queryresult = service.RetrieveMultiple(query);

        List<Type> knownTypes = new List<Type>();
        knownTypes.Add(typeof(Entity));

        FileStream writer = new FileStream(filename, FileMode.Create);
        DataContractSerializer ser = new DataContractSerializer(
 typeof(EntityCollection),knownTypes);
        ser.WriteObject(writer,queryresult);
        writer.Close();
    }
}


public static void LoadData(ServerConnection.Configuration serverConfig, string filename)
{
    using (OrganizationServiceProxy serviceProxy = new OrganizationServiceProxy(
 serverConfig.OrganizationUri,
                                                            serverConfig.HomeRealmUri,
                                                            serverConfig.Credentials,

serverConfig.DeviceCredentials))
```

```
    {
        IOrganizationService service = (IOrganizationService)serviceProxy;

        List<Type> knownTypes = new List<Type>();
        knownTypes.Add(typeof(Entity));

        FileStream fs = new FileStream(filename, FileMode.Open);
        DataContractSerializer ser = new DataContractSerializer(
 typeof(EntityCollection), knownTypes);
        XmlDictionaryReader reader = XmlDictionaryReader.CreateTextReader(fs,
  new XmlDictionaryReaderQuotas());

        EntityCollection fromDisk = (EntityCollection)ser.ReadObject(reader, true);
        reader.Close();
        fs.Close();

        foreach (Entity entity in fromDisk.Entities)
        {
            service.Create(entity);
        }
    }
}
```

Upgrading solutions

One key challenge for any software development effort is how to upgrade solutions that have been deployed into production environments. Upgrades of Microsoft Dynamics CRM solutions have some specific behaviors that should be highlighted, such as:

- Deleting a managed solution will delete any components and associated data that are not included in some other imported solution.
- Installing a version of a solution that is missing components over a solution that contains those components does not remove those components (or their data) from the system. This works fine in most scenarios, but beware of plug-in step registrations that might refer to plug-in classes that no longer exist in the solution.

For a complete discussion of the upgrade models supported by Microsoft Dynamics CRM solutions, in the Microsoft Dynamics CRM SDK, see the topic Maintain Managed Solutions.

Managing solution conflicts

During solution development, it is possible to make changes to shared components in such a way that there are conflicts with changes in other solutions. For example, one solution leverages the built in “Accounts” entity but renames it “Customers,” while a separate solution uses the Accounts entity but renames as “Organizations”. It is also possible that the Microsoft Dynamics CRM organization into which a solution is being imported has been customized in a way to cause conflicts, for example, by renaming “Accounts” as mentioned previously.

In each of these preceding cases, the end result of deploying these conflicting customizations is determined by the order in which the solutions are imported, with the last imported managed solution taking precedence over unmanaged changes.

For more information, in the Microsoft Dynamics CRM SDK, see the following topics:

- Introduction to Solutions: Conflict Resolution.
- Understand How Managed Solutions are Merged.

Deploying solutions

The following sections outline the steps that must be considered when planning for Microsoft Dynamics CRM solution deployments. Many of these steps can be automated using code similar to the examples provided previously in this paper.

Step 1: Environment provisioning

Before you can deploy a solution, you need an environment into which to import it. For production environments, provisioning the environment is essentially a one-time event. During testing however, the environment might need to be provisioned with each test cycle. Be sure to keep the following points in mind:

- For “green field” testing, reset the test environment to the initial state or provision a new organization
- In addition, for upgrade testing, restore the environment and then import a back-up of the production organization, or simply load production solutions and data into a newly provisioned organization

For examples of automating to process of environment provisioning, in the Microsoft Dynamics CRM SDK, see the topic Deployment Web Service in Microsoft Dynamics CRM.

Step 2: Test organization setup

Test deployments also require preparation to configure the environment to represent a specific test scenario

1. Import test business unit structure
2. Import test CRM users (for Microsoft Dynamics CRM Online, send invitations by using the Application UI, if necessary)
3. Assign test users to standard security roles

Step 3: Solution dependencies

To successfully import solutions with dependencies on user identities or with workflow references to specific queues or teams, it is necessary to create some system records before the solution is imported.

1. Import any CRM users that serve as execution identities for workflows or plug-ins
2. Import any teams upon which the solution depends
3. Import any queues upon which the solution depends and assign those queues to appropriate teams or users

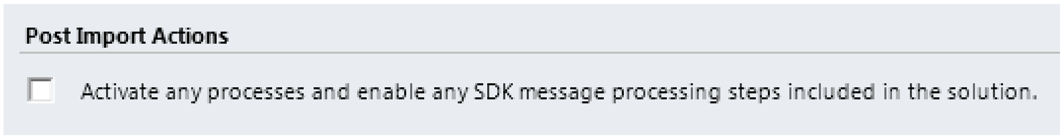
Step 4: Import solution

After environments are prepared and dependencies are in place, it is time to import the solution. This can be accomplished in the Solutions UI or by using the Microsoft Dynamics CRM SDK to call the solution management APIs.

For more details, in the Microsoft Dynamics CRM SDK, see the topic Work with Solutions.

Also under Post Import Actions, be sure to clear the Activate any processes check box when importing a solution that contains either:

- Plug-ins or workflows that depend on specific data that is part of the solution
- Automatic workflows that must be published as a user other than the user who is importing the solution


Post Import Actions

☐ Activate any processes and enable any SDK message processing steps included in the solution.

Note

These items are configured and activated in steps 7 and 8.

Step 5: Configure security for execution identities

Now that the solution and its custom security roles are imported, make sure that any of the Microsoft Dynamics CRM users that were created to execute workflows or plug-ins are configured with appropriate security roles.

- Assign any teams created for the solution to appropriate security roles
- Assign any CRM users created for the solution as workflow and plug-in identities to appropriate teams or security roles

Step 6: Load solution data

After importing the solution and its custom schema, load any data required by the solution.

- Import configuration data and adjust settings to be appropriate for the environment
- Import solution reference data

Step 7: Activate plug-ins

With security for the solution configured appropriately, you can activate any plug-in processing steps. To accomplish this, in Microsoft Dynamics CRM, in Settings, under Customizations, select the steps in the default solution and activate them. This process can also be achieved by using the application APIs as demonstrated in the following example.

```
public static void UpdateStepStatus(CrmOrganization org, Guid stepId, bool isEnable)

{

SetStateRequest request = new SetStateRequest();

request.EntityMoniker = new EntityReference(SdkMessageProcessingStep.EntityLogicalName,

 stepId);

if (isEnable)

{

request.State = new OptionSetValue((int)SdkMessageProcessingStepState.Enabled);

}

else

{

request.State = new OptionSetValue((int)SdkMessageProcessingStepState.Disabled);

}

request.Status = new OptionSetValue(-1);

org.OrganizationService.Execute(request);


return;

}
```

Step 8: Configure and activate solution automatic workflows

In addition, with security configured appropriately you can assign (as discussed previously) and publish workflows.

1. Assign automatic workflows to the appropriate CRM users that were created in step 3 to serve as execution identities.
2. Impersonate the relevant CRM user required for each automatic workflow and publish the workflows.
3. Publish all remaining workflows and dialogs as appropriate (remember that these may not have been published during solution import in step 4).

Step 9: Configure users

Now that the solution has been imported, configured, and activated, the final step is to assign users to appropriate security roles and teams so the users can work with (or test) the new solution.

- Assign users to custom security roles that were created as part of the solution

- Assign users to teams that were created for the solution

The solution should now be fully deployed and ready for testing or production.

Conclusion

Deploying real-world business solutions of even mild complexity into test and production environments in reliable and repeatable ways can involve many elements that often are not obvious to new users developing solutions for Microsoft Dynamics CRM 2011 or Microsoft Dynamics CRM Online. This white paper has attempted to expose the reader to potential challenges and to provide them with tools and approaches to accelerate the development of a successful deployment approach.

Feedback

We appreciate hearing from you. To send your feedback, click the following link and type your comments in the message body.

Note
The subject-line information is used to route your feedback. If you remove or modify the subject line, we may be unable to process your feedback.

Send feedback (http://go.microsoft.com/fwlink/?LinkID=227061)