InRule® for Microsoft® Dynamics® 365 Deployment Guide

Document Updated against InRule v5.4.3

Document Updated against InRule for Microsoft Dynamics 365 Integration Framework v2.3.13

Document Updated against Microsoft Dynamics 365 v9.0 online, v9.0 on-prem

InRule does not upgrade this document after each Integration Framework release, please see release notes for individual versions if the version that you are using does not match the versions listed above.

If you are working with earlier versions of any of the above products, the information in this document may not apply to you. Please check to see if earlier documentation is available to cover your needs.

CONFIDENTIAL Any use, copying or disclosure by or to any other person than has downloaded a trial version of InRule or signed an DNA is strictly prohibited. If you have received this document by any other means than a download or an email from an InRule employee, please destroy it retaining no electronic or printed copies.

© Copyright 2019 InRule Technology, Inc.

Microsoft®, Microsoft Dynamics® and the Microsoft Dynamics Logo are registered trademarks of Microsoft Corporation.

All rights reserved. No parts of this work may be reproduced in any form or by any means – graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems – without the written permission from InRule Technologies, Inc.

InRule, InRule Technology, irAuthor, irVerify, irServer, irCatalog, irSDK and irX are registered trademarks of InRule Technology, Inc. All other trademarks and trade names mentioned herein may be the trademarks of their respective owners and are hereby acknowledged.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document. The publisher and author reserve the right to make corrections, updates, revisions, or changes to the information contained herein. InRule Technology, Inc. does not warrant the material described herein to be free from patent infringement.

Table of Contents

Table of Contents 2
1 Introducing InRule® for Microsoft® Dynamics® 365 4
2 Understanding your options 5
2.1 Microsoft Dynamics 365 (Online) – PaaS with Microsoft Azure 5
2.2 Microsoft Dynamics 365 (On Premise) – Local IIS or Azure IaaS 6
2.3 Other Integration Options 6
3 Performing the Installation: In Azure 7
3.1 An overview of the Components needed 7
3.2 Gathering prerequisites 9
3.3 Deploying and Configuring Components 14
3.3.1 Catalog App Service 14
3.3.2 Registering an Azure Active Directory Application (Optional) 17
3.3.3 Rule Execution App Service for Dynamics 365 19
3.3.4 Upload License file 26
3.3.5 Rule Services Solution for Dynamics 365 27
4 Performing the Installation: On-Premise 36
4.1 An overview of the Components needed 36
4.2 Gathering prerequisites 37
4.3 Deploying and Configuring Components 39
4.3.1 Setting Up an Application Pool in IIS 39
4.3.2 Setting Up the IIS Site for the Rule Execution Web Service 42
4.3.3 Deploy the Rule Execution Web Service 45
4.3.4 Rule Services Solution for Dynamics On-Prem Deployment 47
Appendix A: Additional Resources 51
InRule's Support Website 51
InRule's Support Team 56
InRule's ROAD Team 56
InRule Training Services 56
Appendix B: Anatomy of a Request for Execution of Rules Diagram 57
Appendix C: irX General Integration Concepts 58
Appendix D: Accessing Dynamics 365 Directly from Rule Helper 59
Appendix E: Methods for Executing Rules from Dynamics 365 68
1 Plugin Events 70

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

2 Run Rules Button 72
3 Workflow Activity 73
4 Form Events 74
5 Custom JavaScript 78
6 Custom Action 79
Appendix F: Rules Configuration and Settings 80
Appendix G: Azure App Service Plan Configuration 89
Appendix H: Dynamics 365 Tracing and InRule Event Logging 91
Appendix I: Activating Your License Keys 100
Appendix J: Upgrading Versions 103
Appendix K: Known Issues, Limitations and Troubleshooting 105

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

1 Introducing InRule® for Microsoft® Dynamics® 365

InRule provides the *InRule for Microsoft Dynamics 365 Integration Framework* to enable rule execution integration with Microsoft Dynamics 365. This integration framework serves as a runtime companion to the *irX® for Microsoft Dynamics 365* Product.

This guide focuses on the deployment of the Rule Execution Services for Dynamics and corresponding Dynamics Rule Services Solution to your environment. The InRule Dynamics Solution may be deployed directly from the Microsoft AppSource or from the Integration Framework zip file downloaded from the InRule Support Site. This guide details the primary deployment paths for cloud-based integration in Microsoft Azure® as well as on-premise Dynamics.

Before beginning this guide, you may first want to familiarize yourself with the *irX for Microsoft Dynamics 365* product by reading the *irX for Microsoft Dynamics 365 Help Documentation*. This irAuthor extension will allow you to author rules against Dynamics entities and become familiar with the types of rules-driven processes that can be implemented. After testing locally from your desktop using irVerify, the rules will be ready for execution from Dynamics. At this point, this guide will become highly relevant for deploying the InRule solution and services and establishing the selected integration patterns.

There are a number of options available when it comes to choosing how to integrate InRule with Dynamics 365. This document also provides an addendum, Appendix E: *Methods for Executing Rules from Dynamics 365* that discusses the different options available to you for running rules beyond what the primary deployment steps covers. It is a good next step to review for implementers who are looking for advanced options for running rules.

Additional material is available on the Downloads section of our support website. Please see the Additional Resources section of this document for support website detail.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

2 Understanding your options

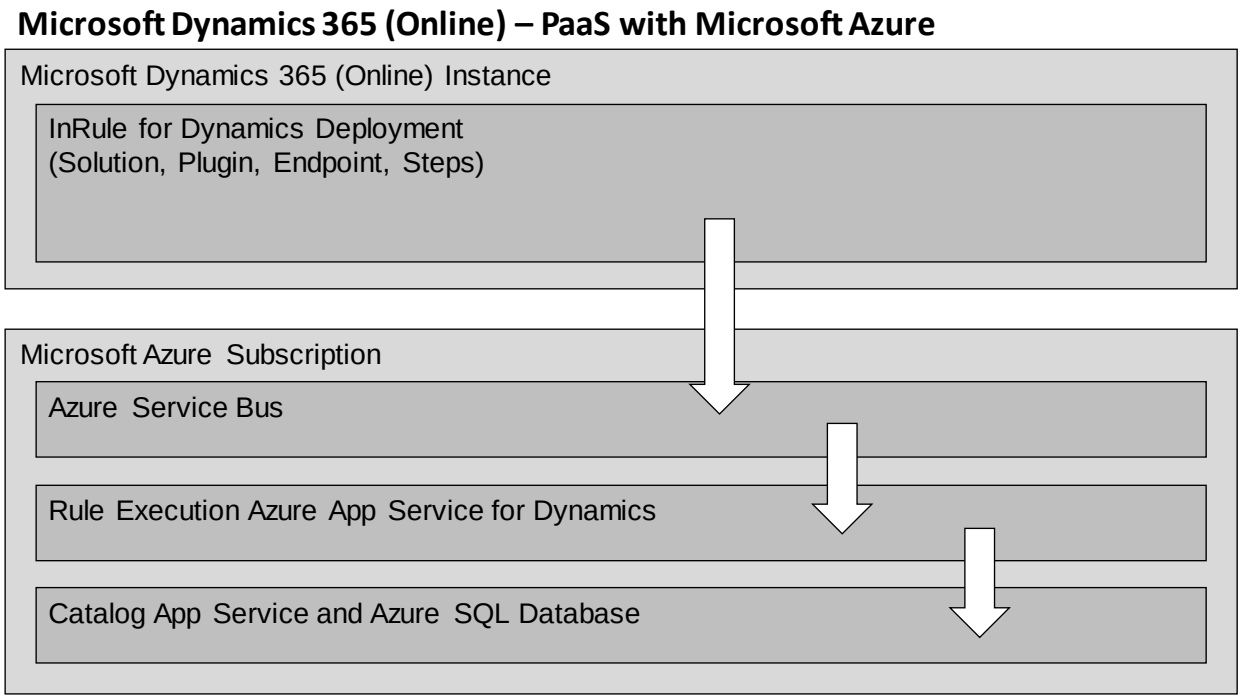
Microsoft Dynamics 365 is available through Microsoft as both an Online SaaS option, and as a downloadable software product that can be installed on either an on-premise environment or through a third party cloud or hosting provider in an IaaS manner with Virtual Machines.

2.1 Microsoft Dynamics 365 (Online) – PaaS with Microsoft Azure

If you are using Microsoft Dynamics 365 (Online), you will be hosting InRule using a Platform-As-A-Service (PaaS) model on Microsoft Azure. When setting up the InRule App Services, ensure that all Azure components and the Dynamics installation are in the same geographical location to reduce timeout exceptions due to network latency.

This document discusses the following four components:

1. Catalog App Service and Azure SQL Database
2. Azure Service Bus Relay
3. Rule Execution App Service for Dynamics 365
4. Rule Services Solution for Dynamics 365



Section 3 of this document, Performing the Installation: In Azure, provides a complete walkthrough.

Appendix B: Anatomy of a Request for Execution of Rules Diagram contains a more complete diagram depicting how a standard request navigates through components.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

2.2 Microsoft Dynamics 365 (On Premise) – Local IIS or Azure IaaS

If you are using Microsoft Dynamics 365 (On premise) or plan to deploy to the cloud using an Infrastructure-As-A-Service (IaaS) model, you will be hosting our integration framework via Internet Information Services (IIS) on a Window Server OS. This document discusses the following 3 components:

1. Catalog Web Service and Database
2. Rule Execution Web Service for Dynamics 365 On-Premise
3. Rule Services Solution for Dynamics 365 On-Premise

Section 4 of this document, Performing the Installation: On-Premise, provides a complete walkthrough.

2.3 Other Integration Options

You are welcome to look at alternate integration options. Here are some possible integration options that you may pursue:

- Integration with Microsoft Dynamics 365 (Online) without the use of an Azure Service Bus.
- Integration with Microsoft Dynamics 365 (Online) utilizing IaaS options instead of Azure's PaaS options.
- Integration with Microsoft Dynamics 365 (On premise) against Azure PaaS hosted InRule components. If this is your preferred approach, an on-premise Dynamics installation will work off the shelf with the inRule installation instructions in the "Performing the Installation: In Azure" section of this document.

While all of these are possible routes, they will not work with the Integration Framework we provide off the shelf. As such, we strongly encourage that you follow the model outlined in this guide to start with, until you've acquired a more advanced understanding of how all of the pieces of the *InRule® for Microsoft Dynamics 365 Integration Framework* work together to provide a solution.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

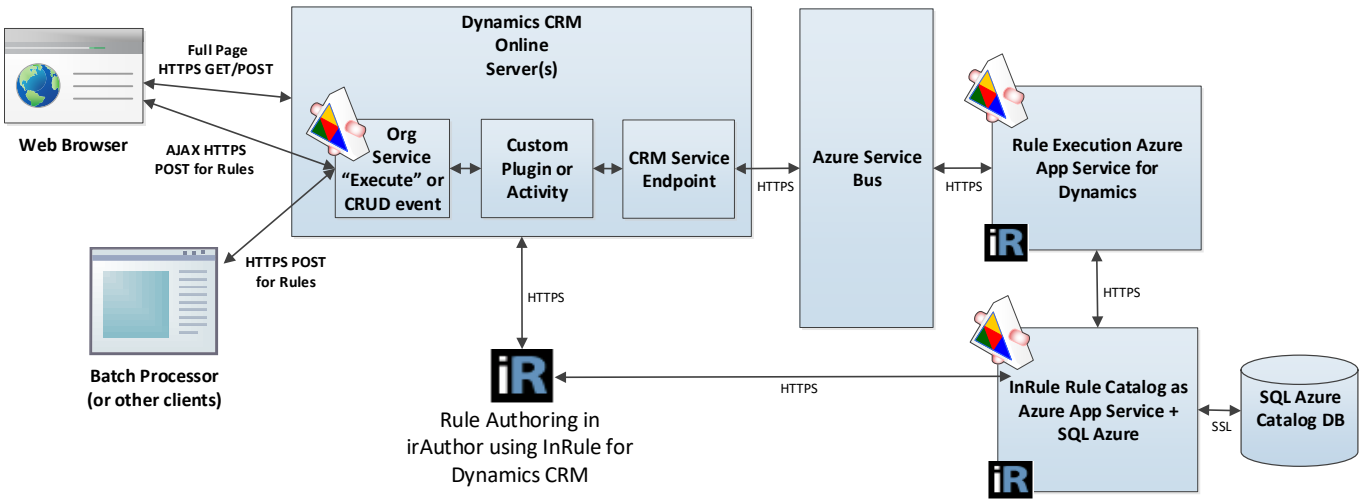
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3 Performing the Installation: In Azure

This section discusses the steps needed to integrate InRule with Microsoft's online version of Microsoft Dynamics 365. Using this scenario InRule components are hosted on Microsoft Azure.

3.1 An overview of the Components needed

This guide will provide the instructions for setting up all of the components below:



Catalog App Service and Azure SQL Database

A Catalog service will be used to store Rule Apps that will be consumed by the Rule Execution App Service. This Catalog Service will be hosted as an Azure App Service. The back end of the Catalog Service utilizes an Azure SQL Database for retrieval and persistence of Rule Applications.

Azure Service Bus Relay

Dynamics 365 is designed to communicate to third party services through an Azure Service Bus. Utilizing an Azure Service Bus is the preferred communication mechanism allowing for security and quick horizontal scaling of services. The *Rule Execution App Service* connects to the Service Bus and registers itself as a Relay listener. When Dynamics 365 makes a request to the Service Bus, the Service Bus relays that message to a listener and allows for two-way communications for as long as the connection is open.

The *InRule Rule Execution App Service* leverages a specific kind of Service Bus called Azure Relay. Within the context of the InRule architecture, the Relay performs the same function as a "traditional" Service Bus, but includes some extra functionality specific to WCF relays. For the sake of consistency with Microsoft's Dynamics documentation, we will refer to the Relay as a Service Bus throughout this document. However, be aware that they are different resource types within Azure itself.

Rule Execution App Service for Dynamics 365

The Rule Execution App Service is responsible for loading Dynamics entity data, executing rules against loaded data, and responding to Dynamics with rule execution results.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Rule Services Solution for Dynamics 365

The Rule Services Solution contains a custom plugin, an endpoint, client resources, a configuration form, and a security role called InRule Integration Administrator. It must be configured to communicate with the Azure Service Bus.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3.2 Gathering prerequisites

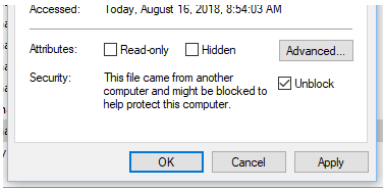
This section reviews what you will want to have prepared before you begin with the integration steps in the next section.

Required Files

The following file should be downloaded from our support website's downloads section before you begin:

- `InRule for Microsoft Dynamics 365 Integration Framework.zip`

After you have downloaded this file, but before extracting, make sure that you go to the file properties for the zip and select **Unblock**. If the zip file is not unblocked before extracting, the deployment scripts will not be able to execute successfully.



After unblocking the zip file, extract the contents to a working folder. When you are finished, you should have a directory structure that looks like this:

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

```
│   InRule For Microsoft Dynamics CRM Integration Framework.zip
│
└──InRule For Microsoft Dynamics CRM Integration Framework
    │   readme.txt
    │
    ├──DynamicsDeployment
    │   │   Deploy-CrmPackage.ps1
    │   │   ...(many other supporting files)
    │   │
    │   ├──PkgFolder
    │       └  ...(many other supporting files)
    │
    ├──RuleApplications
    │       DynamicsRules.ruleapp
    │
    ├──RuleExecutionAzureService
    │       Register-AzureApp.ps1
    │       azuredeploy.json
    │       azuredeploy.parameters.json
    │       InRule.Crm.WebJob.zip
    │
    ├──RuleExecutionOnPremService
    │       DeployScript.ps1
    │       InRule.Crm.WebService.deploy-readme.txt
    │       InRule.Crm.WebService.SetParameters.xml
    │       ...(many other supporting files)
    │
    └──RuleHelperDeployment
            │  InRule.Crm.RuleHelper.dll
            └ ...(many other supporting files)
```

Files in **bold** will be used directly in the walkthrough steps below.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Rule Authoring Environment

A Rule Authoring Environment is used to upload a Rule Application to your catalog app service. A Rule Authoring Environment is a machine or virtual machine where irAuthor has been installed with the irX for Microsoft Dynamics 365 extension. If you followed the instructions outlined in *irX for Microsoft Dynamics 365 Help Documentation* then you should already have a rule-authoring environment available to you.

We have made it a point to call out the rule authoring environment separately because it is important to be aware of the licensing implications of this step. You will need to utilize an irAuthor license and an irX for Microsoft Dynamics 365 license for the duration of this process. If you're a system administrator who does not intend to perform rule authoring duties after the deployment is up and running, you can either chose to borrow an environment from someone who will use a rule authoring environment, or you will want to be sure to deactivate your license when you're finished with your deployment responsibilities.

Administrative Accounts

Dynamics 365 Organization Service URI: You will want to have the root URL of the organization web service exposed by your Dynamics 365 instance. The server URL is usually in the format of "https://organization-name.crm.dynamics.com"

Dynamics Service Account Login and Password: You will want to have a username and account created specifically for use by the InRule for Microsoft Dynamics 365 Framework App Service. This account should have the 'System Administrator' role as it is continually updated with permission to new entities when they are created. Alternatively, this user can have another security role such as the 'InRule Integration Administrator' which is included in the InRule solution. If an alternate role is used, it is important to note that its permissions are not dynamically updated and will have to be done manually. This user account will not be needed if you decide to create an S2S user account outlines in Section 3.3.2: Registering an Azure Active Directory Application (Optional) of this document.

Administrative Password to use for SQL Server: You should decide what username and password you want to use for administrative privileges on the SQL Server. You will use this password when following the referenced catalog setup guide.

*** This walkthrough will utilize the above administrative login and password for the Catalog Service to connect to the SQL Server Database. In a more secure environment, a separate SQL User should be created that only has access to the single database needed by the catalog. It is up to the reader of this document to go this more secure route.*

Administrative Password to use for Catalog Service: You should decide what username and password you want to use for administrative privileges within irCatalog. You will use this password when following the referenced catalog setup guide and will need to provide it when deploying the Execution Service.

*** This walkthrough utilizes the default login of '`admin`' and password of '`password`'. It will be up to the reader to go through the process of utilizing the Catalog Manager to change these credentials to be more secure.*

Administrative Login and Password for Microsoft Azure: You must have a username and password that will be used to perform administrative tasks within Microsoft Azure.

InRule Azure License File

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

You will need a special .xml file used for licensing InRule in an Azure cloud environment. This may have been provided with your InRule Welcome package. You can contact support@InRule.com if you have questions about where to get your license file.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Deciding resource names

The following worksheet can be used to decide what to name Azure resources as you go through this Guide.

Many of these resources must have names that are unique in the world; they are hosted on Microsoft Azure and are given domain names that match. We recommend creating a “Base” name that does not exceed 14 characters. We recommend encoding an organization name, an application name, and an environment name into this ‘Base’ name. For Example:

```
{ApplicationAbbreviation}{OrganizationAbreviation}{EnvironmentAbreviation}
```

MyAppInRuleDev
12345678901234

You can choose to follow this convention or invent your own.

Resource and Description	Example Name
Base Name	MyAppInRuleDev
Azure Resource Group Name	MyAppInRuleDevResourceGroup
Azure SQL Server Name *must be lower case*	myappinruledevsqlserver
Catalog Database Name	MyAppInRuleDevCatalogDb
Catalog App Service Name	MyAppInRuleDevCatalogService
Service Bus Namespace	MyAppInRuleDevServiceBus
Rule Execution App Service Plan Name	MyAppInRuleDevRuleExecutionAppServicePlan
Rule Execution App Service Name	MyAppInRuleDevRuleExecutionAppService

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3.3 Deploying and Configuring Components

3.3.1 Catalog App Service

Installing the Catalog App Service

The first major objective for a Dynamics 365 implementation is the deployment of a Catalog service to Microsoft Azure.

During this process, you will be creating:

- An Azure SQL Server
- An Azure SQL Server Database
- An Azure App Service to host the InRule Catalog in the Azure cloud

When you are finished, you should be able to connect to this app service from a locally installed copy of irAuthor, and successfully save a RuleApp to the catalog.

The full process of installing the Catalog in Microsoft Azure is outlined in the documentation found on the InRule AzureAppServices GitHub, which can be found here:

https://github.com/InRule/AzureAppServices#ircatalog

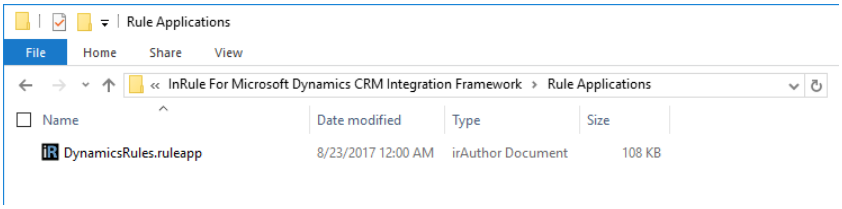
Pleasure ensure you are installing the irCatalog service, not the irServer Rule Execution Service, which is found on the same page and is not compatible with Dynamics 365 integration.

Testing the Catalog App Service by uploading the starter Rule App

At the conclusion of the installation process outlined above you should have a Catalog URI, Username, and Password to use to connect to the catalog service.

Next we will utilize your rule authoring environment to upload the Rule Application that you extracted into your working directory at:

```
\InRule for Microsoft Dynamics 365 Integration Framework\Rule Applications\DynamicsRules.ruleapp
```

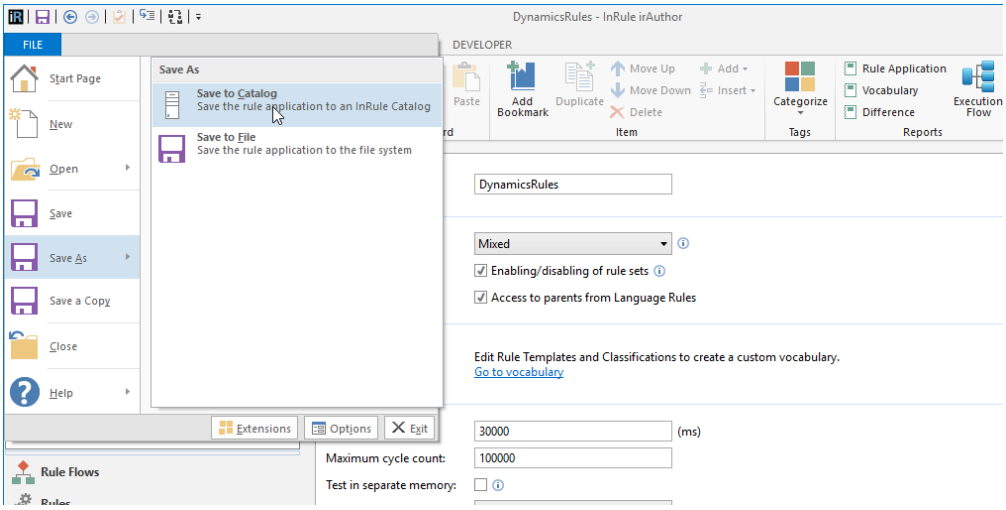


1: Navigate to this file in your rule authoring environment and double click on it, this will open the file with irAuthor.

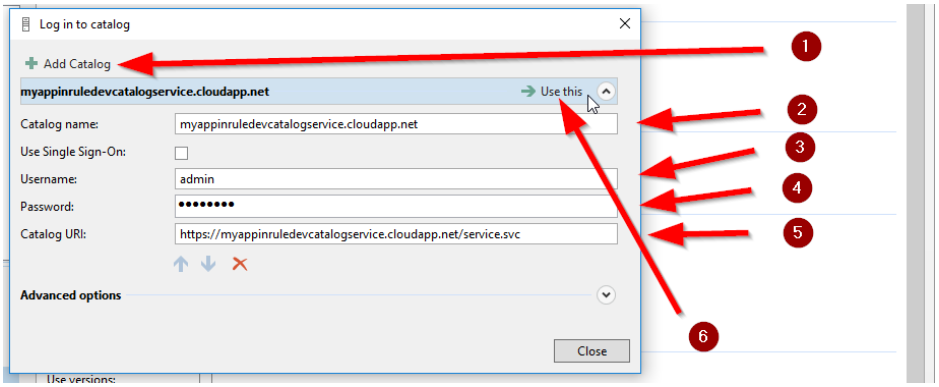
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

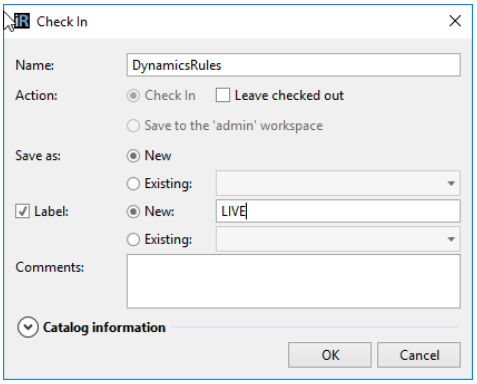
2: Save the Rule Application by chosing `File` → `Save As` →`Save to Catalog`



3: Choose `Add Catalog`, enter connection information for the Catalog Server that you deployed, and then select `Use This`.



4: Save with the name `DynamicsRules` and the Label `LIVE`



Save the Rule Application to the Catalog using the name of *DynamicsRules*. We must also be sure to label this Rule Application with the text '*LIVE*', as configured in the app service configuration. Be sure not to forget this label, it is a small but important step!

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

At this point, if you can click OK without an error, we have successfully saved the DynamicsRules Rule App to the new Catalog that you have created. We can now continue by creating the Service Bus Namespace.

If you have any trouble getting to this point, it is advised that you resolve any issues with the Catalog before attempting to continue.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3.3.2 Registering an Azure Active Directory Application (Optional)

This section is only necessary if you intend to leverage a server-to-server (S2S) connection between the rule execution service and your Dynamics environment. Server-to-server authentication uses a Dynamics Application User associated to an Azure AD Application for authentication instead of a named user account. This approach can be beneficial as it allows Dynamics integration with InRule without having to purchase an additional full user. Additionally, this will save you from needing to store Dynamics user passwords in Azure. For more information about server-to-server authentication, please refer to the following link: https://docs.microsoft.com/en-us/dynamics365/customer-engagement/developer/build-web-applications-server-server-s2s-authentication

If you do not want to use server-to-server authentication, you can skip this section and connect with a named user account and Dynamics connection string instead.

Be sure to reference Appendix J: Known Issues, Limitations and Troubleshooting of this document for considerations regarding the user account and permissions created with this approach.

Registering an Azure AD application is fully automated in the Register-AzureApp.ps1 PowerShell script included in the *RuleExecutionAzureService* folder within the *InRule for Microsoft Dynamics 365 Integration Framework.zip* file downloaded in Section 3.2: Required Files.

This script requires the Azure AD PowerShell module to be installed on the computer before running the script. If this module is not yet installed, you can install it by opening an admin PowerShell window and executing 'Install-Module AzureAD`.

1: Navigate to the `\RuleExecutionAzureService` directory:

```
PS C:\> cd .\RuleExecutionAzureService\
PS C:\RuleExecutionAzureService>
```

2: Execute Register-AzureApp.ps1

The script accepts the parameters "Username" and "Password," which need to be credentials for an Azure Active Directory Global Administrator. Alternatively, you can run the script without passing in any credentials, and the script with prompt you with an interactive login.

By default, the application is registered in the primary directory the user account belongs to. If the app needs to be registered in a different Azure Active Directory tenant, you can pass in the tenant ID with the optional "TenantId" parameter.

Without a TenantId passed:

```
PS C:\RuleExecutionAzureService> .\Regiser-AzureApp.ps1 -Username admin -Password password
```

With an example TenantId passed:

```
PS C:\RuleExecutionAzureService> .\Regiser-AzureApp.ps1 -Username admin -Password password `
>>  -TenantId 29d4c658-27b9-4a52-829d-f2dfe1cc7dfg
```

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Observe that no errors occur while the application registration script is running.

When finished, the script will output three values: The application name, application ID, and the secret key. Save these 3 values, particularly the secret key, as there will be no way to retrieve this later. The application ID will be needed when deploying the Rule Execution package, and both the Application ID and Key will be needed when deploying the Rule Execution App Service.

```
Application Details for the InRuleDynamicsCrmIntegration application:
        ==========================================================
        Application Name:       InRuleDynamicsCrmIntegration
        Application Id:         [redacted]
        Secret Key:             [redacted]
```

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3.3.3 Rule Execution App Service for Dynamics 365

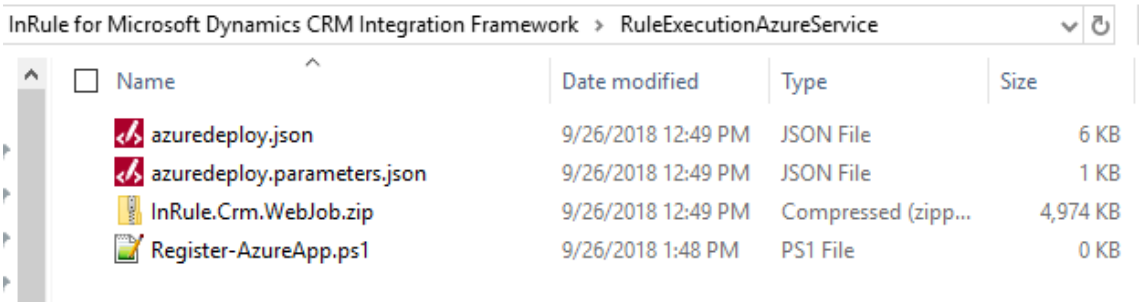
Next, we'll deploy the InRule Rule Execution service as an Azure WebJob running on an App Service, along with all its Azure resource dependencies. To make this process easier, we'll be using an Azure Resource Manager (ARM) template, which allows us to deploy and configure all the Azure resources the Rule Execution Service relies on.

There are a number of methods for deploying an ARM template; this documentation will detail two: via Azure CLI and via PowerShell.

1: Locate azuredeploy.parameters.json

Before deploying the ARM template, we need to define certain parameters.

Locate the *azuredeploy.parameters.json* file in the *RuleExecutionAzureService* folder within the *InRule for Microsoft Dynamics 365 Integration Framework.zip* file downloaded in Section 3.2: Required Files.



2: Update parameters

Open the file with your text editor of choice and edit the parameters listed below

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

```
1   {
2     "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
3     "contentVersion": "1.0.0.0",
4     "parameters": {
5       "appServiceName": {
6         "value": ""
7       },
8       "relayName": {
9         "value": ""
10      },
11      "catalogUri": {
12        "value": "https://{Context}/Service.svc"
13      },
14      "catalogUser": {
15        "value": "admin"
16      },
17      "catalogPassword": {
18        "value": "password"
19      },
20      "orgUrl": {
21        "value": ""
22      },
23      "appId": {
24        "value": ""
25      },
26      "appSecret": {
27        "value": ""
28      },
29      "crmConnectionString": {
30        "value":  ""
31      },
32      "appServicePlanName": {
33        "value": ""
34      },
35      "inRuleVersion": {
36        "value": "5.4.2"
37      },
38      "appInsightsInstrumentationKey": {
39        "value": ""
40      }
41    }
42  }
```

1 appServiceName	Provide a name for the Azure App Service that the Rule Execution Service will run on.
2 relayName	Provide a name for the Azure WCF Relay (aka Azure Service Bus). This is NOT the desired namespace URL for the Relay, it should be the desired name of the actual Azure resource.
3 catalogUri	The URI for the Catalog Service that will be used Example: https://myappinruledevcatalogservice.cloudapp.net/service.svc
4 catalogUser	Username for Catalog Service, default value is 'admin'.
5 catalogPassword	Password for Catalog Service, default is 'password', please change this using the catalog manager!
6 orgUrl (If using named user account authentication instead of S2S authentication, leave blank)	The root URL for the Dynamics 365 instance. Follows this format: https://organization-name.crm.dynamics.com If using named user account authentication instead of S2S authentication, leave this blank. The Dynamics Org URL will be defined as a part of the crmConnectionString parameter instead.
7 appId (If using named user account authentication instead of S2S authentication, leave blank)	ID for the Azure Active Directory application registered in Registering an Azure Active Directory Application. If using connection string instead of S2S authentication, provide a value for the crmConnectionString parameter instead

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

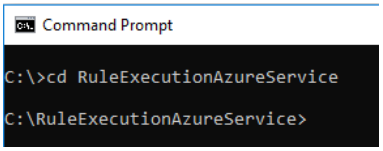
8 appSecret (If using named user account authentication instead of S2S authentication, leave blank)	Key secret for the Azure AD app registered in Registering an Azure Active Directory Application. If using named user account authentication instead of S2S authentication, provide a value for the crmConnectionString parameter instead
9 crmConnectionString (If using S2S authentication, leave blank)	Provide a Dynamics connection string if you are not using S2S authentication, otherwise leave blank. Information on connection string formatting can be found here: https://docs.microsoft.com/en-us/dynamics365/customer-engagement/developer/xrm-tooling/use-connection-strings-xrm-tooling-connect The account used here should have the 'System Administrator' role or the 'InRule Integration Administrator' role. It is important to note that the 'InRule Integration Administrator' role is not updated when new entities are created. Reference Appendix J: Known Issues, Limitations and Troubleshooting for more information. ***Important:*** Some customers have reported needing to provide a username in the format "domain\username" in order to successfully connect using IFD.
10. **inRuleVersion**	InRule version to use for deploying the execution service. The default is the most recent version.
11. **appServicePlanName** (If you wish to override the default value)	Provide a name for the Azure App Service Plan. If you leave this value blank it will be derived as the App Service name you provide above, with "Plan" appended to the end (ex. appServiceNamePlan) Note, by default, an App Service Plan will be created by the ARM template. If you wish to deploy your app service to an already existing App Service Plan rather than create a new one, or for more information on the necessary configurations for the App Service Plan, reference Appendix G: Azure App Service Plan Configuration
12. **appInsightsInstrumentationKey**	If you want to use Application Insights as a log sink in addition to the app service logging already enabled, provide an instrumentation key from an existing Application Insights resource here. If you leave this parameter blank, the execution service will still log to the app service log as described in Rule Execution Service Event Log.

3: Option 1: Deploy ARM Template with Azure CLI

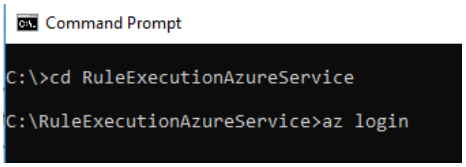
Now that the ARM template is configured, we'll deploy it to get the resources up and running. The following will detail how to use the Azure CLI to deploy the ARM template (Note, this section assumes Azure CLI has already been installed):

3.1 Run Command Prompt or Powershell

3.2 Navigate to the RuleExecutionAzureService folder

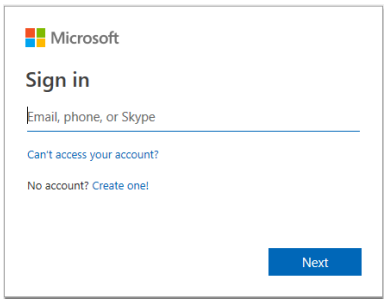


3.3 Enter "az login" to login into Azure



3.4 Enter your Azure admin credentials to login when prompted in the new browser window opened

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



3.5 Select the appropriate subscription

If your Azure account has access to multiple subscriptions, you will need to set your active subscription to where you create your Azure resources:

```
C:\RuleExecutionAzureService>az account set --subscription SUBSCRIPTION_NAME
```

3.6 Create Resource group

If you have not created a resource group yet, you will need to create one. You will need to define a name and a geographic location for where to host the resource. This example uses Central US:

```
C:\RuleExecutionAzureService>az group create --name ResourceGroupName --location centralus
```

3.7 Execute the following command to deploy the ARM template

Replace “ResourceGroupName” with the name of the Azure Resource Group you want to deploy to

```
C:\RuleExecutionAzureService>az group deployment create -g ResourceGroupName
--template-file .\azuredeploy.json --parameters .\azuredeploy.parameters.json
 --query properties.outputs.relayKey.value -o table
```

Observe that the template deploys with no errors

If the template deploys successfully, you will see a result that looks similar to this:

```
Result
-----------------------------------------
kwSQxK+[redacted]
```

The value that is output is your Azure Service Bus key, which is a value you will need to configure your Dynamics instance in a later step.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

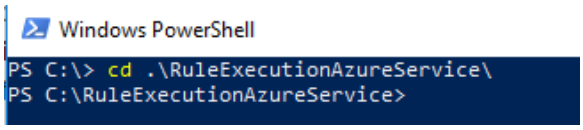
4: Option 2: Deploy ARM Template with Powershell

(If you have already deployed the ARM template via Azure CLI in the section above, this section is not necessary)

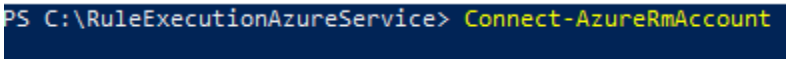
Now that the ARM template is configured, we'll deploy it to get the resources up and running. The following will detail how to use Powershell to deploy the ARM template (Note, this section assumes Azure PowerShell has already been installed):

1.1 Run Powershell

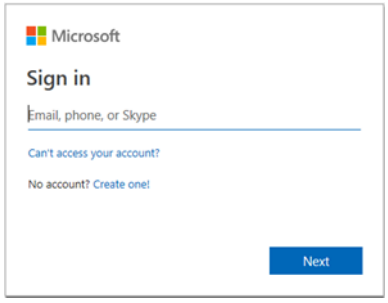
1.2 Navigate to the RuleExecutionAzureService folder



1.3 Enter "Connect-AzureRmAccount" to login into Azure

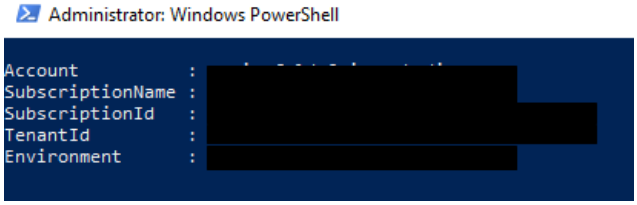


1.4 Enter your Azure admin credentials to login when prompted in the new browser window opened



1.5 Select the appropriate subscription

Upon logging in, your default subscription information will be displayed:



If this is not the subscription you want to deploy to, you can use the "Select-AzureRmSubscription" cmdlet to change the targeted subscription. Just replace "SubscriptionNameHere" with the name of the desired subscription:

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Windows PowerShell

```
PS C:\RuleExecutionAzureService> Select-AzureRmSubscripton -SubscriptionName SubscriptionNameHere
```

4.6 Create Resource Group

If you have not created a resource group yet, you will need to create one. You will need to define a name and a geographic location for where to host the resource. This example uses Central US:

```
PS C:\RuleExecutionAzureService> New-AzureRmResourceGroup -Name ResourceGroupName -Location centralus
```

4.7 Execute the following command to deploy the ARM template

Replace "ResourceGroupName" with the name of the Azure Resource Group you want to deploy to

Windows PowerShell

```
PS C:\RuleExecutionAzureService> New-AzureRmResourceGroupDeployment -ResourceGroupName ResourceGroupName `
>> -TemplateFile .\azuredeploy.json -TemplateParameterFile .\azuredeploy.parameters.json
```

Observe that the template deploys with no errors

If the deployment is successful, you should see an output similar to this:

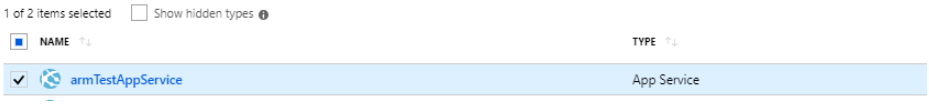
```
DeploymentName          : azuredeploy
ResourceGroupName       : [redacted]
ProvisioningState       : Succeeded
Timestamp               : 9/27/2018 7:45:26 PM
Mode                    : Incremental
TemplateLink            :
Parameters              :
                          Name             Type                       Value
                          ===============  =========================  ==========
                          [redacted]

Outputs                 :
                          Name             Type                       Value
                          ===============  =========================  ==========
                          relayKey         String                     [redacted]
```

Note the value of "relayKey" underneath the "Outputs" heading. This value is your Azure Service Bus key, and will be needed when configuring your Dynamics instance in later steps.

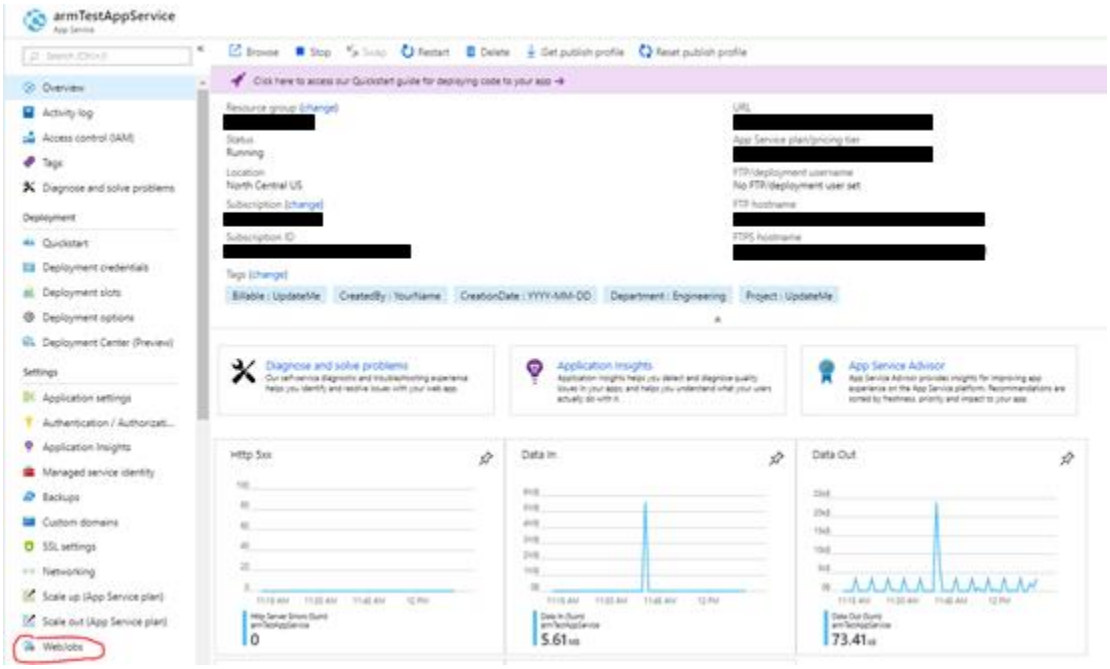
5: Verify Setup

Navigate to the Azure portal and locate the deployed App Service

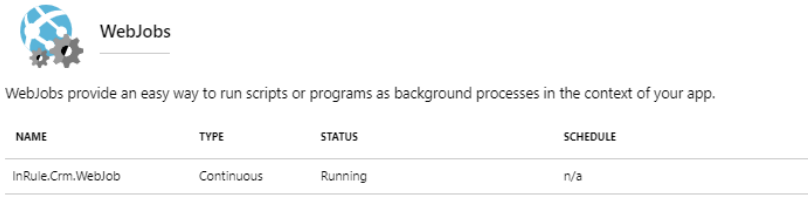


Click on the app service, and on the nav-bar that appears to the left of the resource overview, select WebJobs

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Ensure the InRule.Crm.WebJob is present and its “Status” is set to “Running”



WebJobs

WebJobs provide an easy way to run scripts or programs as background processes in the context of your app.

NAME	TYPE	STATUS	SCHEDULE
InRule.Crm.WebJob	Continuous	Running	n/a

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3.3.4 Upload License file

Regardless of how you choose to deploy the ARM template, you'll need to upload a license file to the web app service in order for both the catalog service and the rule execution app service to properly function. The simplest way to upload the license file is via FTP.

This example leverages Azure CLI in addition to Powershell commands. If you intend to use this method, please run the CLI from Powershell.

Alternative approaches using Powershell only should be possible if desired but are not detailed in this document.

First, retrieve the FTP deployment profile (url and credentials) with the az webapp deployment list-publishing-profiles command and put the values into a variable:

```
# Example: az webapp deployment list-publishing-profiles --name contoso-execution-prod-wa --resource-group inrule-prod-rg --query "[?contains(publishMethod, 'FTP')].{publishUrl:publishUrl,userName:userName,userPWD:userPWD}[0]" | ConvertFrom-Json -OutVariable creds | Out-Null

az webapp deployment list-publishing-profiles --name WEB_APP_NAME --resource-group RESOURCE_GROUP_NAME --query "[?contains(publishMethod, 'FTP')].{publishUrl:publishUrl,userName:userName,userPWD:userPWD}[0]" | ConvertFrom-Json -OutVariable creds | Out-Null
```

Then, upload the license file using those retrieved values:

```
# Example:
$client = New-Object System.Net.WebClient;$client.Credentials = New-Object System.Net.NetworkCredential($creds.userName,$creds.userPWD);$uri = New-Object System.Uri($creds.publishUrl + "/InRuleLicense.xml");$client.UploadFile($uri, "$pwd\InRuleLicense.xml");

$client = New-Object System.Net.WebClient;$client.Credentials = New-Object System.Net.NetworkCredential($creds.userName,$creds.userPWD);$uri = New-Object System.Uri($creds.publishUrl + "/InRuleLicense.xml");$client.UploadFile($uri, "LICENSE_FILE_ABSOLUTE_PATH")
```

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3.3.5 Rule Services Solution for Dynamics 365

At this point, all of the Azure Requirements are met. The Azure Service bus should be listening for incoming communication from Dynamics. We must now setup Dynamics, which can be done in one of two ways:

1. **AppSource Deployment** -- You can install the Rule Services Solution through Microsoft AppSource. The InRule AppSource listing can be found by going to AppSource and searching for InRule. Proceed to Step 4 in the following instructions after installing from AppSource.

2. **PowerShell Deployment --** You can follow the guide below to deploy the Dynamics package via PowerShell script.

Important: You only need to deploy the Rule Services Solution by **one** of the two above methods. Doing both is unnecessary.

Important: Only deploy the solution using one of the two above methods. Do not manually import included solution files.

1: Navigate to the '`\DynamicsDeployment`' directory:

```
Administrator: Windows PowerShell
PS C:\> cd .\DynamicsDeployment\
PS C:\DynamicsDeployment>
```

2: Execute Deploy-CrmPackage.ps1

`Deploy-CrmPackage.ps1`

```
PS C:\> .\Deploy-CrmPackage.ps1
```

If you are using S2S authentication in the execution service, you will need to pass in your Azure Active Directory application ID as the "AzureAppId" parameter to the Deploy-CrmPackage.ps1 script. This is the application ID that was output in Section 3.3.2, and will be used to create a new Application User in Dynamics. If you were not involved in registering your Azure Active Directory application, your Azure administrator should be able to provide you with this ID. If you are using connection-string based authentication in the execution service, you do not need to provide this parameter.

```
PS C:\DynamicsDeployment> .\Deploy-CrmPackage.ps1 -AzureAppId <app ID here>
```

Additionally, the script supports 2 other parameters: "SbNamespaceAddress" and "SasKey". These parameters allow you to pass in the ***fully qualified*** Service Bus Namespace Address (e.g.: https://<<ServiceBusNamespace>>.servicebus.windows.net/<<SB Relay Path>>) and Service Bus Key for the Service Bus your Rule Execution App Service is listening on and will allow the script to auto-configure your Dynamics instance to point at that Service Bus. These parameters operate independently of the "AzureAppId" parameter; they may be alongside it or on their own. However, if you opt to provide a value for either "SbNamespaceAddress" or "SasKey," you must provide a value for the other as well.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

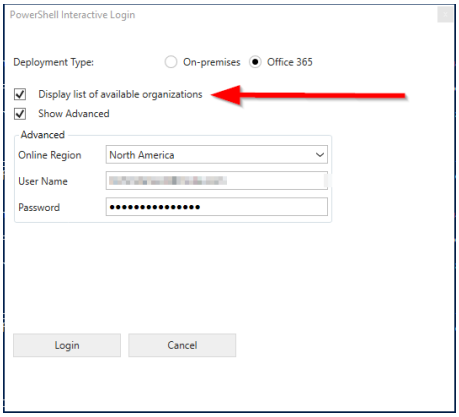
```
PS C:\DynamicsDeployment> .\Deploy-CrmPacke.ps1 -SbNamespaceAddress <service bus name here>
-SasKey <service bus key here>
```

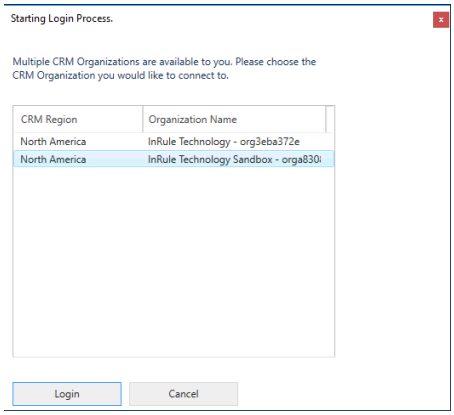
If you choose not to provide the “SbNamespaceAddress” and “SasKey” parameters, you will receive a notification (shown below) after the script finishes executing reminding you that you need to manually configure the SAS Namespace and SAS Key as outlined in Step 4 of this section.

```
The SAS Key needs to be set in the Rule Execution Service
 Endpoint Configuration in order to run rules
```

3: Provide Dynamics 365 Service Credentials

By default, the script will display an interactive login window to connect to Dynamics. Please be sure to check the ‘Display list of available organizations’ to make sure that you select the correct instance of Dynamics to install to!





Alternatively, the Deploy-CrmPackage.ps1 script also accepts a Dynamics Connection String:

```
.\Deploy-CrmPackage.ps1
-CrmConnectionString 'AuthType=Office365;Url=https://irroadsandbox.crm.dynamics.com/;Username=CRM[redacted].com;Password=[redacted]'
```

```
.\Deploy-CrmPackage.ps1 -CrmConnectionString
'AuthType=Office365;Url=https://{DYNAMICSURL}.crm.dynamics.com/;Username={DYNAMICSUSERNAME};Pa
ssword={DYNAMICSPASSWORD}'
```

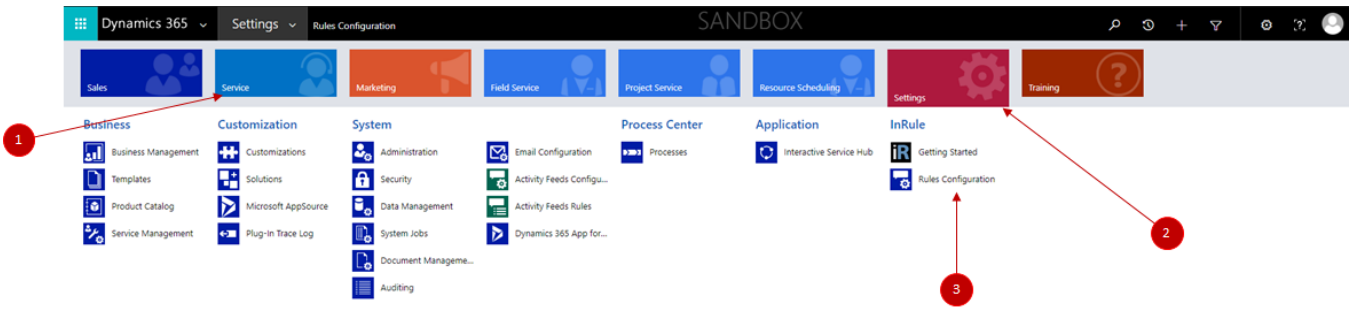
Observe that no errors occur while the Deploy Script is running.

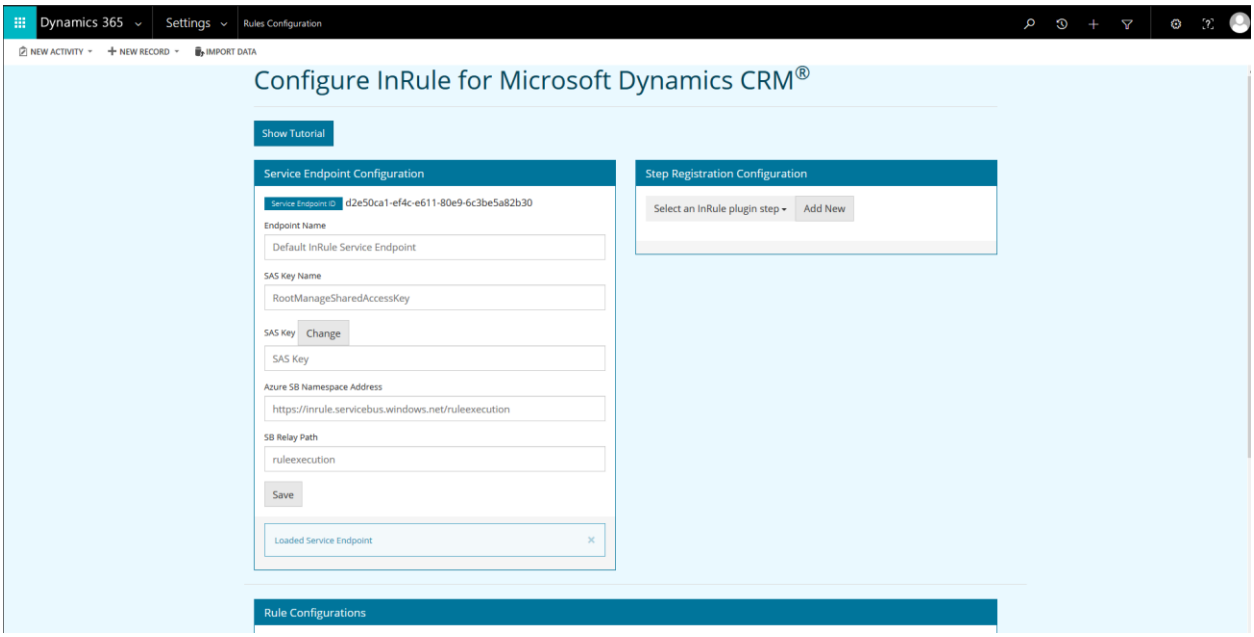
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

4: Configure the Rule Services Solution in Dynamics

Next, we need to configure the Dynamics Solution that has been deployed to point to the Rule Execution Service that is already available on Azure. We will need to have the Service Bus Namespace and the Service Bus Key (SAS Key) ready for this step. You can find more information about these settings in the Rule Execution App Service for Dynamics 365 section.

Navigate to the Rules Configuration page:





You will need to define values in the following fields.

Configuration Form Fields	**Details on these settings are found in "Rule Execution App Service for Dynamics 365" section.**
Endpoint Name	For basic configuration, this can be left as is.
SAS Key Name	The key output by the ARM template is the 'RootManageSharedAccessKey' key, so this value can be left as the default in most scenarios.
SAS Key	This is the key secret value, and is output by the ARM template after completion of the deployment **Note: If you provided the "SasKey" parameter in Step 2 of this section, this value will already be set.**

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Azure SB Namespace Address	Change the service bus namespace here, as the default value of 'inrule-crmonline' will not work for you. The service bus namespace is the same value provided for the relay name in the ARM template parameters file. Using the Service Bus Namespace and the Service Bus Path, construct the address as follows: https://<<ServiceBusNamespace>>.servicebus.windows.net/<<SB Relay Path>> **Note: If you provided the "SasName" parameter in Step 2 of this section, this value will already be set.**
SB Relay Path	Unless you have changed the corresponding setting in rule execution app service, this value can be left as the default, which is 'ruleexecution'

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

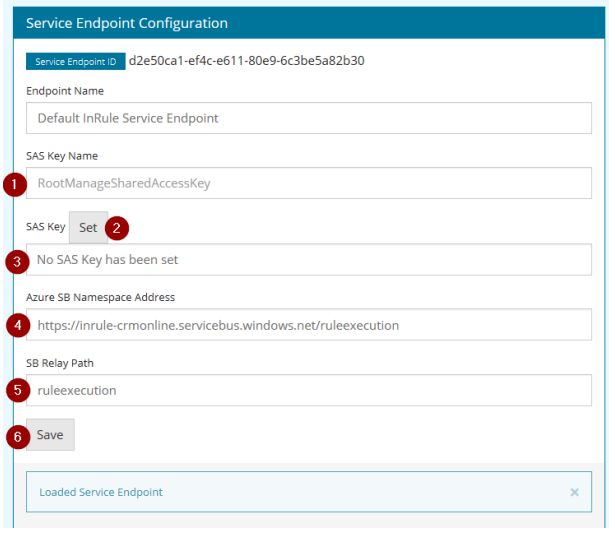
The following screenshots show the Service Endpoint Configuration screen before and after configuration.

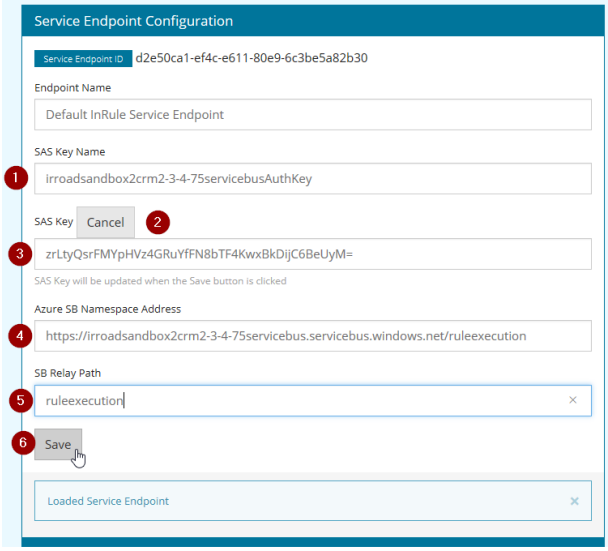
- [#1 in screenshot] SAS Key Name
- [#3 in screenshot] SAS Key
- [#4 in screenshot] Azure SB Namespace Address
- [#5 in screenshot] SB Relay Path

Please note that you must click on 'SET' [#2 in screenshot] before the SAS Key can be entered.

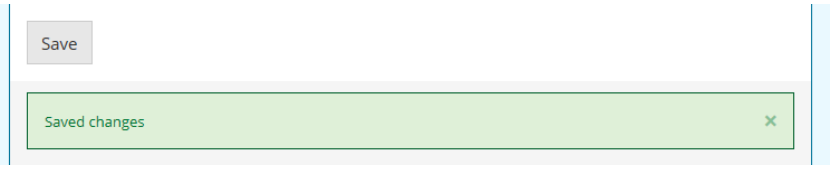
For a more complete description of all of the configuration options available, reference [Appendix F: *Rules Configuration and Settings*]

Notice: If you are using the standard Azure US environment 'AzureCloud', then the provided domain suffix will work fine. However, if you are using an alternate Azure Environment such as US Government or an international environment – be sure to update the suffix to the appropriate value for your region.





Be sure to click save, and to observe the "Saved Changes" notice:



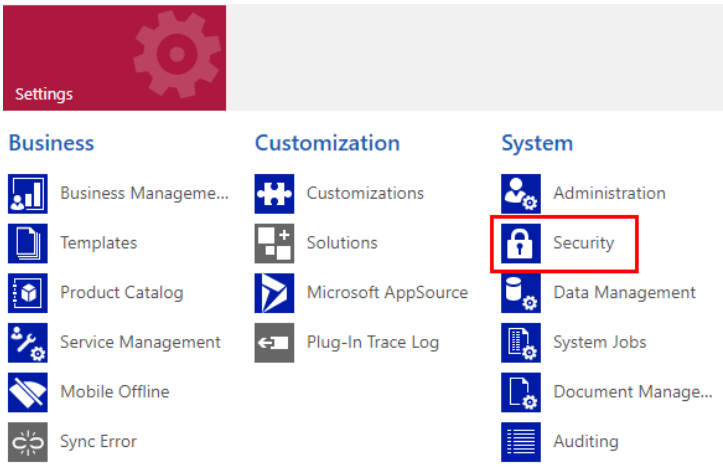
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

5: Manually Configuring S2S Authentication for AppSource Deployments

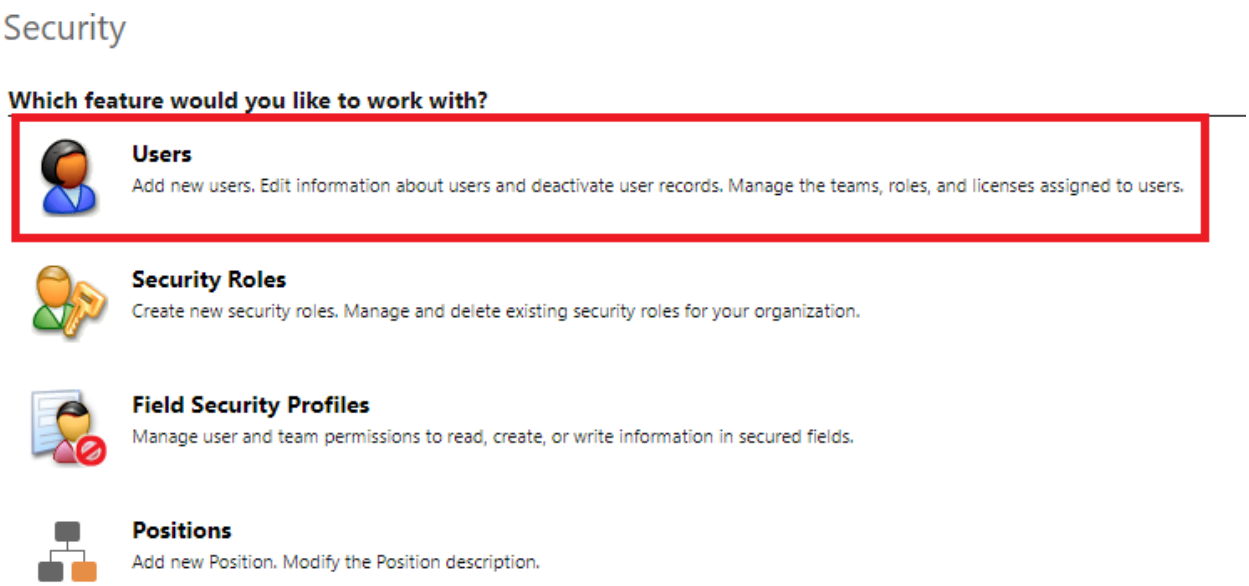
When deploying the solution via AppSource, the creation of the Application User that is used by S2S Authentication between the Rule Execution Service and the Dynamics Rule Services Solution must be manually configured if desired. S2S Authentication is for single tenant AzureAD applications only.

Important: These steps are only necessary for users that have deployed via AppSource and want to leverage S2S authentication with a single tenant Azure AD application.

5.1: Navigate to Settings > Security:

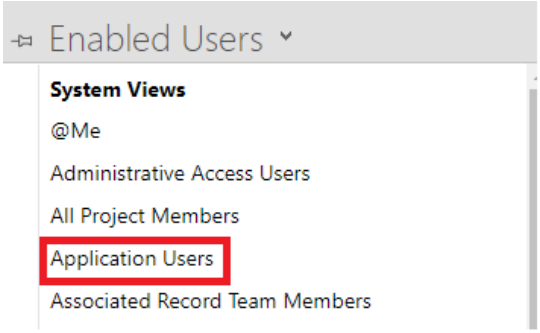


5.2: Select Users:

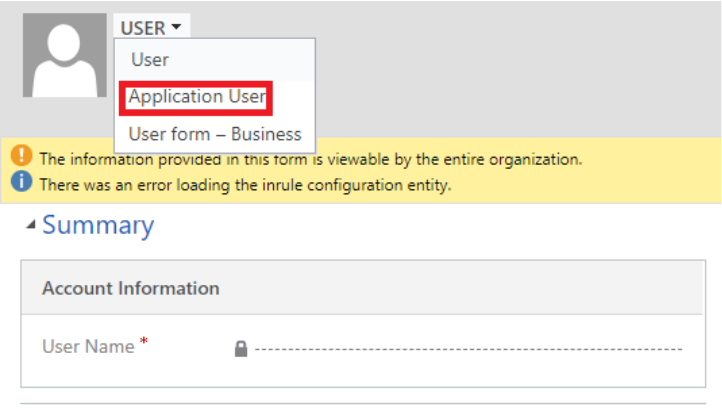


5.3: Toggle to the Application Users View:

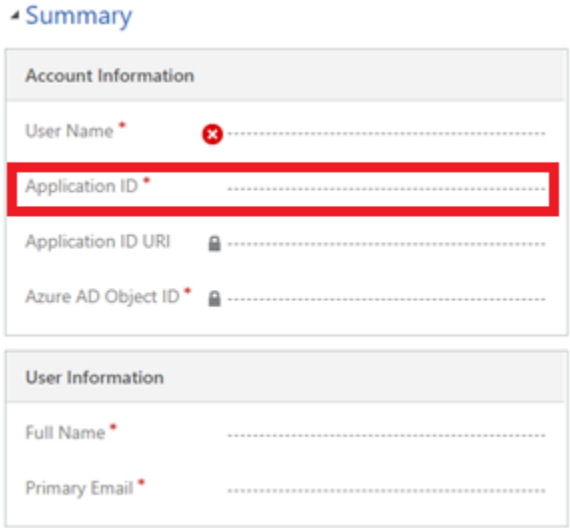
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



5.3: Select New, and Toggle to Application User



5.4: Populate the Application ID based on the configured S2S Application (as outlined in section 3.3.2)

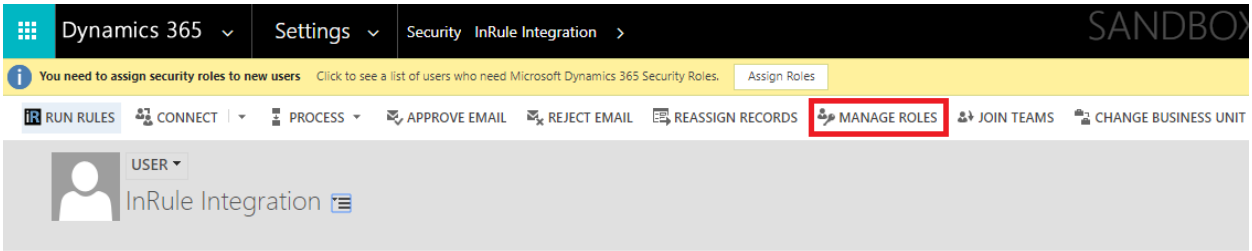


5.5: Populate the Username, Full Name and Primary Email with NEW account info for what the App User should be called

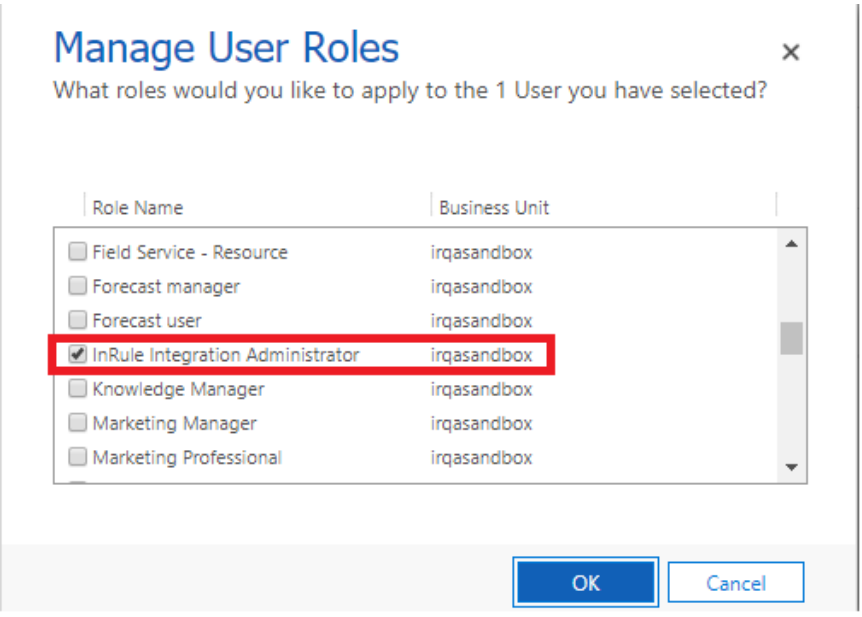
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

5.6: Click "Save"

5.7: Click "Manage Roles" in the top menu bar



5.8: Tick the box for "InRule Integration Administrator" role and click OK

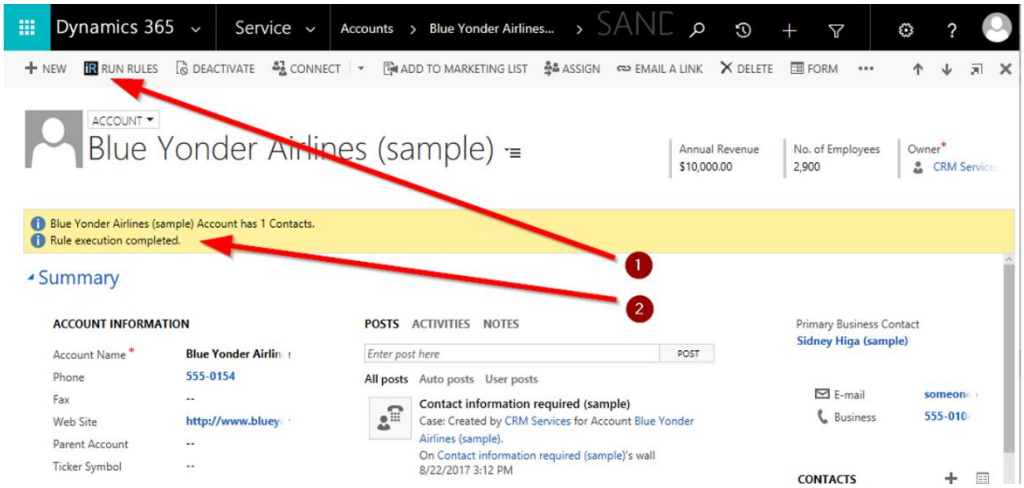


Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

6: Verify a successful deployment!

A ‘`Run Rules`’ button will now exist on the edit form of all Dynamics entities. The way this button behaves will depend on the settings that have been configured under the Rule Configuration section; this document assumes the default configuration behavior. To understand what settings are available and what they mean, please see Appendix F: *Rules Configuration and Settings* of this document.

Open up an Account and execute ‘Run Rules’. The default deployment is configured to run a Rule App from the catalog named `DynamicsRules`, which is what we uploaded to the catalog in the Verify Catalog stage. If everything has been set up correctly, you should see the “Rule Execution Completed” message and the description of the account will be updated to provide the date and time.



If there are any issues with executing rules and you are using an S2S user account from section 3.3.2, check Appendix J: Known Issues, Limitations and Troubleshooting of this document. If you are using the connection string, make sure the user account specified has the necessary privileges to access the particular entity. In most applications, the user account used in the connection string has the role of ‘System Administrator’. This security role is automatically updated with permission when new entities are created. If the user account has the role ‘InRule Integration Administrator,’ it is important to note that the permissions of the role are only updated when the InRule solution is deployed. Adding permission for new entities will have to be done manually.

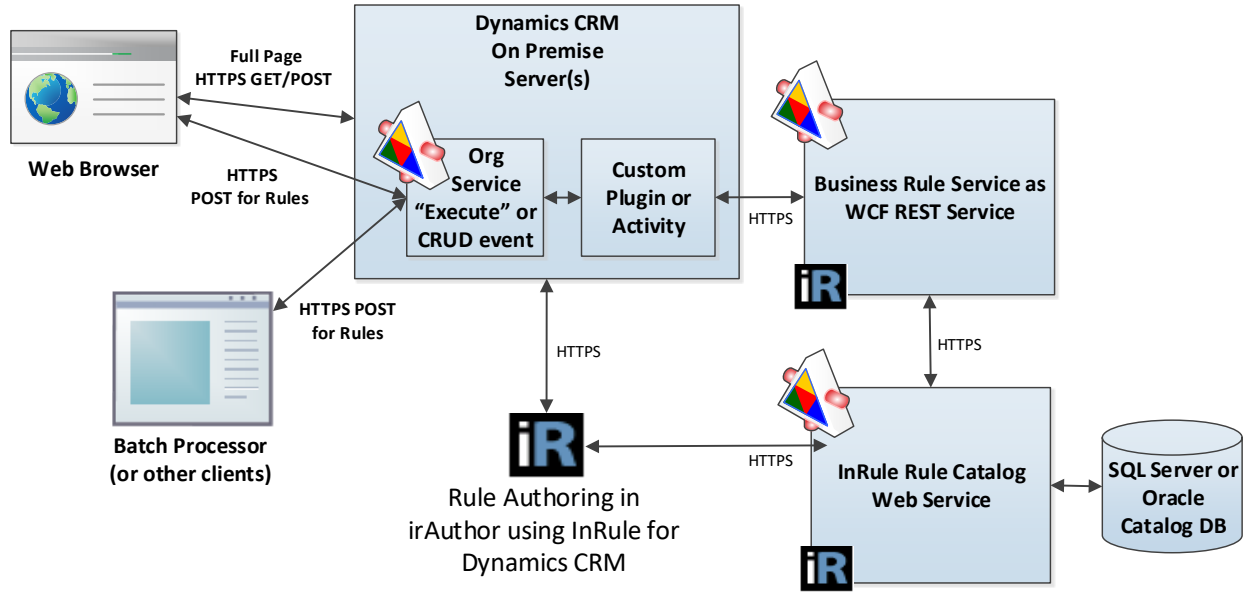
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

4 Performing the Installation: On-Premise

This section discusses the steps needed to integrate InRule with Microsoft's on-premise version of Microsoft Dynamics.

This section assumes that you already have an IIS web server and an on-premise version of Dynamics deployed and configured.

4.1 An overview of the Components needed



Catalog Web Service and Database

A Catalog service will be used to store Rule Apps that will be consumed by the Rule Execution Service. This Catalog Service will be hosted as a web service on an IIS server. The back end of the Catalog Service utilizes a SQL Server or Oracle database for retrieval and persistence of Rule Applications. The Installation Documentation and InRule Installer, which will be used to install the service itself, are available on our support website's downloads section provides instructions for setting up the catalog.

Rule Execution Web Service for Dynamics 365 On-Premise

The WCF Web Service implementation is designed to be generically applied to any given Dynamics Entity and corresponding InRule rules that may apply to that Entity. After the web service is published, the service can be directed to run various Rule Applications against different Entity types by passing arguments in the REST calls. This will require a separate deployment than the Catalog Web Service; this process will be covered in the "Deploy the Rule Execution Web Service" section.

Rule Services Solution for Dynamics 365 On-Premise

The `RuleExecutionServices` solution contains a custom plugin, client resources, and a configuration form. Unlike the Online solution, this is configured to work directly against the execution service, rather than requiring the "middle-man" of the Azure Service Bus.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

4.2 Gathering prerequisites

This section reviews what you will want to have prepared before you begin with the integration steps in the next section.

Required Files

The following files should be downloaded from [our support website's downloads section]() before you begin:

- `InRule for Microsoft Dynamics 365 Integration Framework.zip`
- `InRule Catalog Installer – InRule Installer.exe`

Please download and extract each of these archives to corresponding subdirectories within a working folder.

When you are finished, you should have a directory structure that looks like this:

```
│   InRule For Microsoft Dynamics CRM Integration Framework.zip
│
└──InRuleForMicrosoftDynamicsCRMIntegrationFramework
    │   readme.txt
    │
    ├──DynamicsDeployment
    │   │   Deploy-CrmPackage.ps1
    │   │   ...(many other supporting files)
    │   │
    │   ├──PkgFolder
    │       └  ...(many other supporting files)
    │
    ├──RuleApplications
    │       DynamicsRules.ruleapp
    │
    ├──RuleExecutionAzureService
    │       Register-AzureApp.ps1
    │       azuredeploy.json
    │       azuredeploy.parameters.json
    │       InRule.Crm.WebJob.zip
    │
    ├──RuleExecutionOnPremService
    │       DeployScript.ps1
    │       InRule.Crm.WebService.deploy-readme.txt
    │       InRule.Crm.WebService.SetParameters.xml
    │       ...(many other supporting files)
    │
    └──RuleHelperDeployment
            │  InRule.Crm.RuleHelper.dll
            └ ...(many other supporting files)
```

Files in **bold** will be used directly in the walkthrough steps below.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Rule Authoring Environment

A Rule Authoring Environment is used to upload a Rule Application to your catalog app service. A Rule Authoring Environment is a machine or virtual machine where irAuthor has been installed with the irX for Microsoft Dynamics 365 extension. If you followed the instructions outlined in *irX for Microsoft Dynamics 365 Help Documentation,* then you should already have a rule-authoring environment available to you.

We have made it a point to call out the rule authoring environment separately because it is important to be aware of the licensing implications of this step. You will need to utilize an irAuthor license and an irX for Microsoft Dynamics 365 license for the duration of this process. If you're a system administrator who does not intend to perform rule authoring duties after the deployment is up and running, you can either chose to borrow an environment from someone who will use a rule authoring environment, or you will want to be sure to deactivate your license when you're finished with your deployment responsibilities.

Administrative Accounts

Dynamics On-Prem Organization Service URI: You will want to have the root URL of the organization web service exposed by your Dynamics 365 instance. The server URL is usually in the format of "http://crm-server:port/organization-name"

Dynamics Service Account Login and Password: You will want to have a username and account created specifically for use by the InRule for Microsoft Dynamics 365 Framework Web Service.

Administrative Password to use for SQL Server: You should decide what username and password you want to use for administrative privileges on the SQL Server. This walkthrough will construct a connection string that will utilize these resources to connect to the catalog.

*** This walkthrough will utilize the above administrative login and password for the Catalog Service to connect to the SQL Server Database. In a more secure environment, a separate SQL User should be created that only has access to the single database needed by the catalog. It is up to the reader of this document to go this more secure route.*

Administrative Password to use for Catalog Service: You should decide what username and password you want to use for administrative privileges within irCatalog.

*** This walkthrough utilizes the default login of 'admin' and password of 'password'. It will be up to the reader to go through the process of utilizing the Catalog Manager to change these credentials to be more secure.*

InRule License Keys

You will need your InRule irServer license keys. These can be found at https://support.inrule.com/activations.aspx. You can contact support@InRule.com if you have questions about where to get your license keys. For guidance on how to activate your keys, reference Appendix I.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

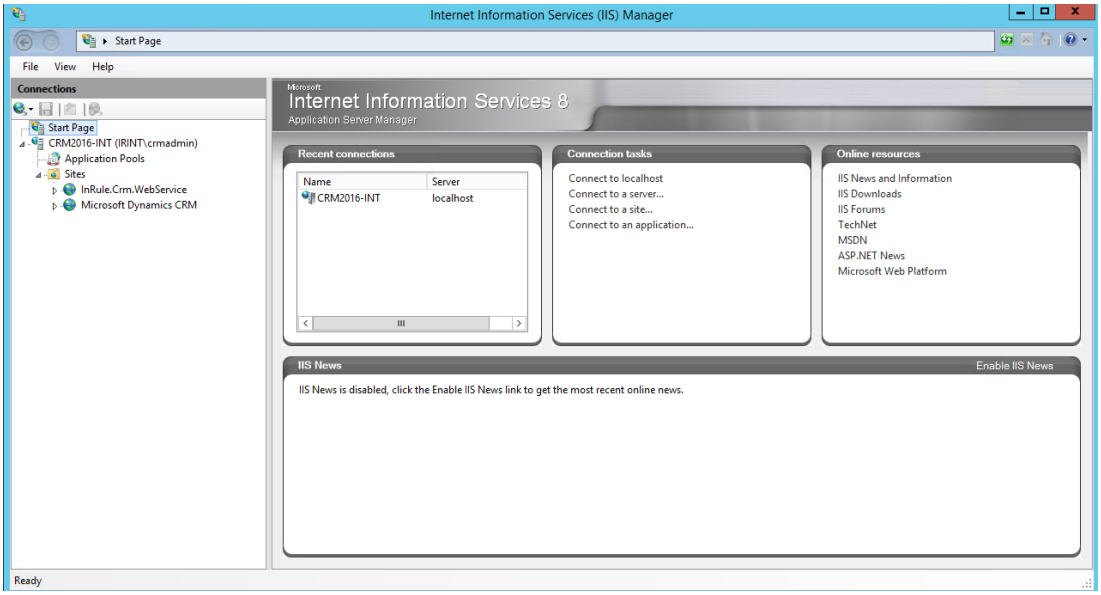
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

4.3 Deploying and Configuring Components

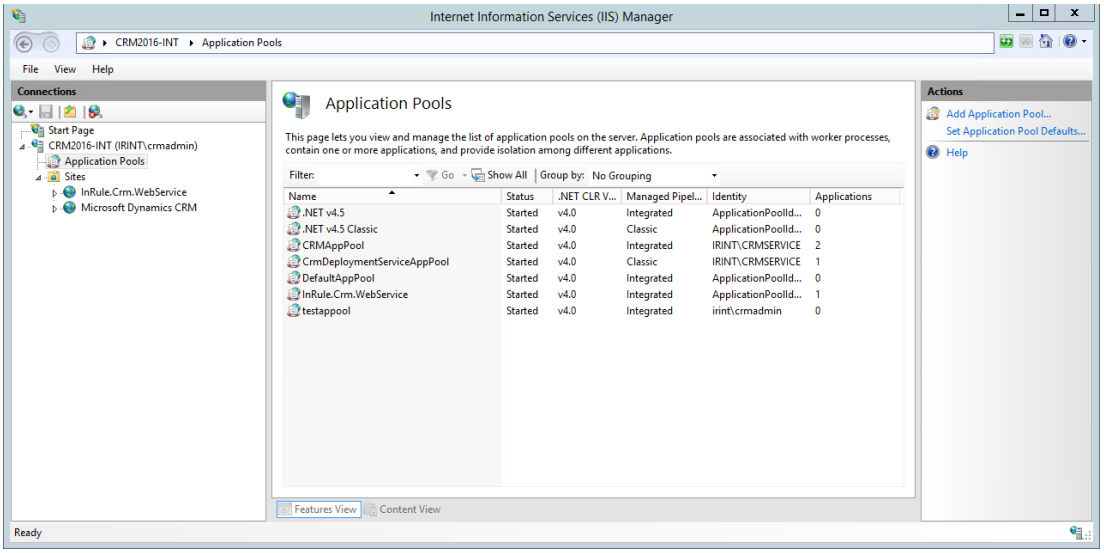
4.3.1 Setting Up an Application Pool in IIS

Before a website can be setup to host the catalog web service, a suitable application pool needs to be created. Any IIS server will already have a DefaultAppPool that can be used; whether this section is necessary will depend on your organizations architecture and requirements.

1: Open IIS Manager

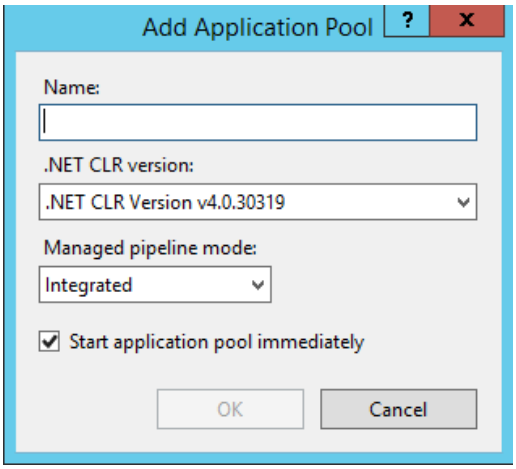


2: Click on Application Pools on the left-hand nav pane



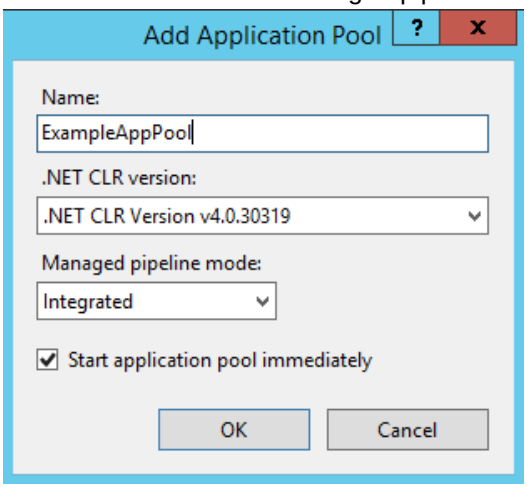
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3: Click "Add Application Pool" under "Actions" in the right-hand pane



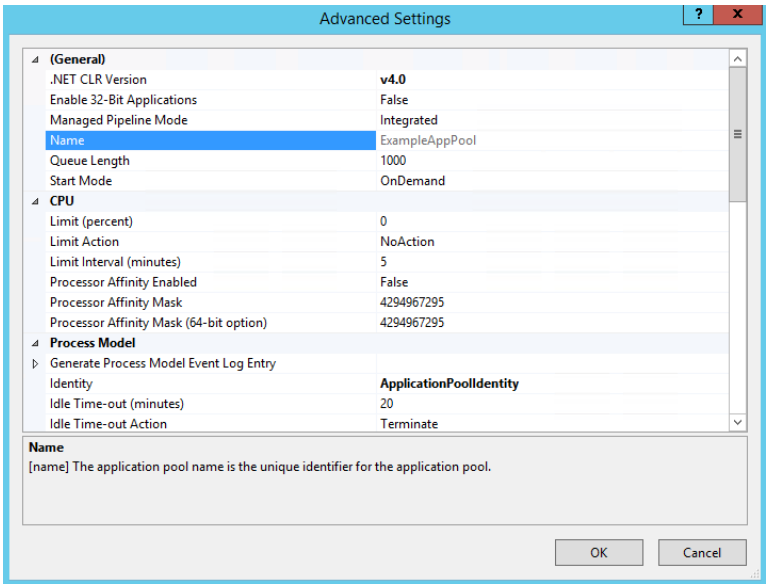
4: Define a name for the Application Pool and Create

.NET CLR Version and Managed pipeline mode should be left as their default values. Press OK

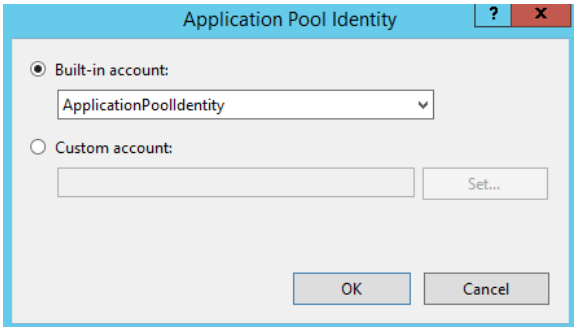


Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

5: Click on the new App Pool and Select "Advanced Settings" on the right-hand pane



6: Under the "Process Model" header, select the small "…" button next to the "Identity" property



Here you can select what kind of identity model you want to use for your application pool. You'll want to either use the Application Pool Identity or create a custom account.

The Application Pool Identity creates a virtual account with the same name as your new application pool. All worker processes within this application pool will run under this account

Custom accounts are the best solution if you want to use your Windows credentials to authenticate to the Catalog web service.

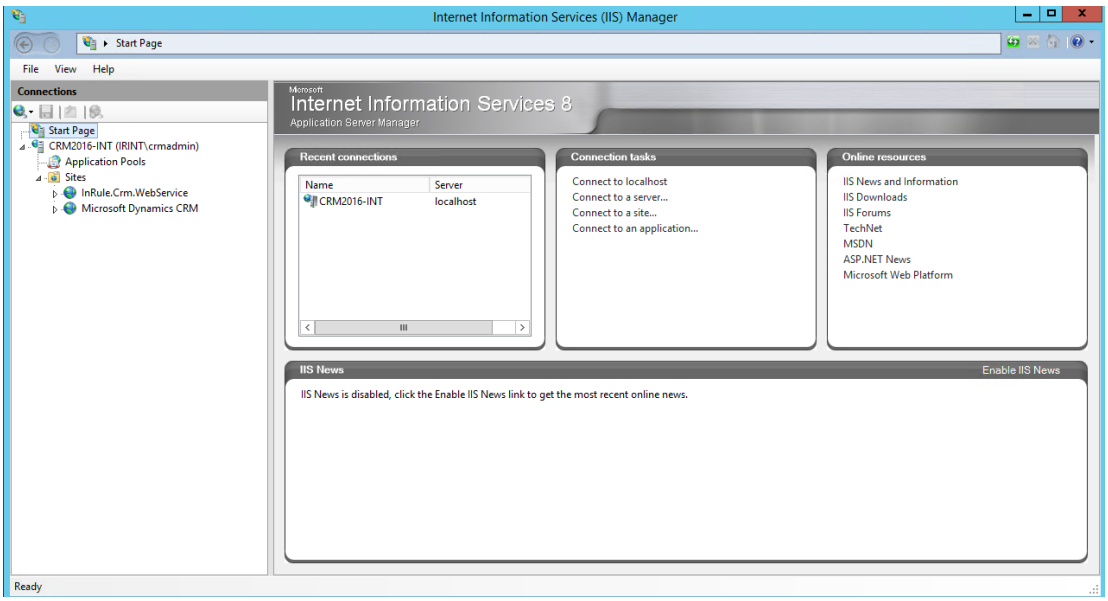
Press "OK" when you have made your selection and are finished.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

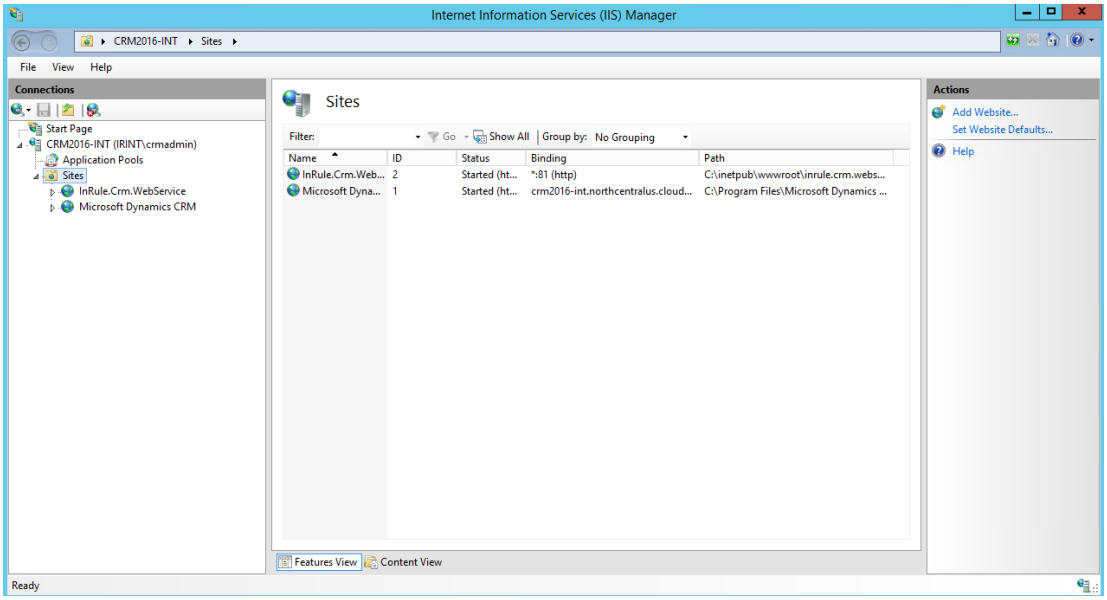
4.3.2 Setting Up the IIS Site for the Rule Execution Web Service

Next, we'll need to setup a website in IIS to host the execution web service. This will require a Windows Server with IIS installed.

1: Open IIS Manager



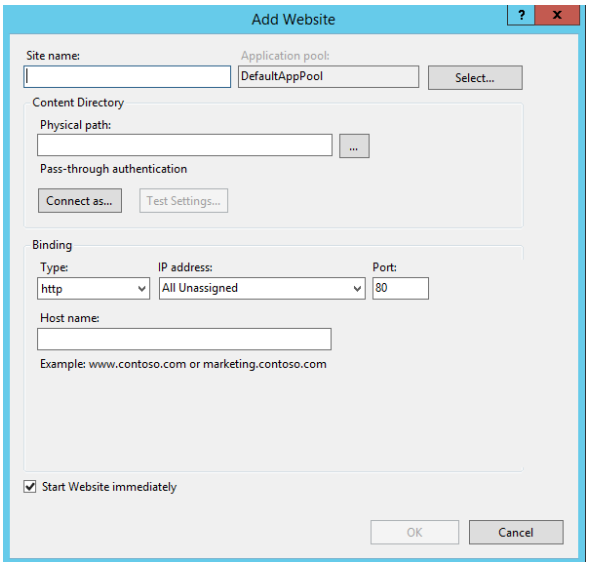
2: Click on sites on the left-hand nav pane



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3: Click "Add Website" under "Actions" in the right-hand pane



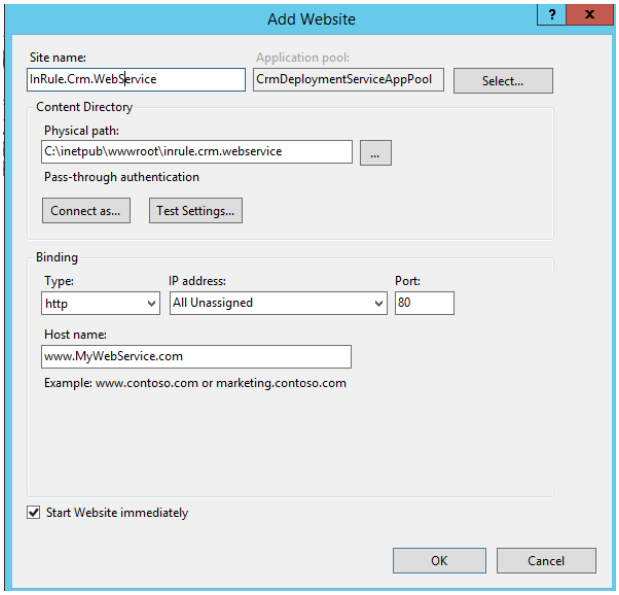
4: Define website information

Define a site name, application pool and host name for the website; these can be configured however makes sense within your organization.

The physical path needs to be set to the actual Dynamics Rule Execution web service folder. This will require you having the InRule components downloaded onto the same server you are setting up the website on

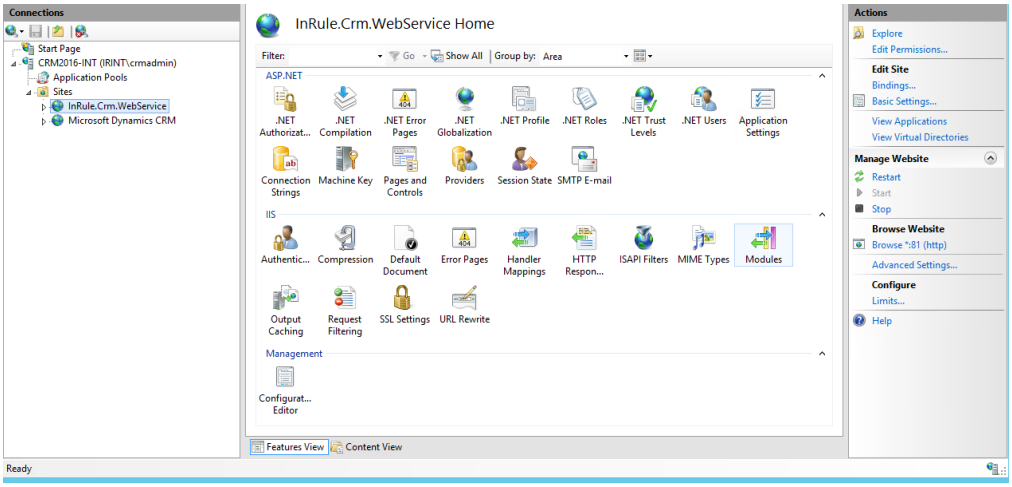
Lastly, select either HTTP or HTTPS, depending on which makes sense for your architecture. If you opt to use HTTPS, you will be required to select a certificate before you can finalize the site.

Once you've populated all the required fields, press OK to finalize and start the website



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

5: Verify your website successfully created



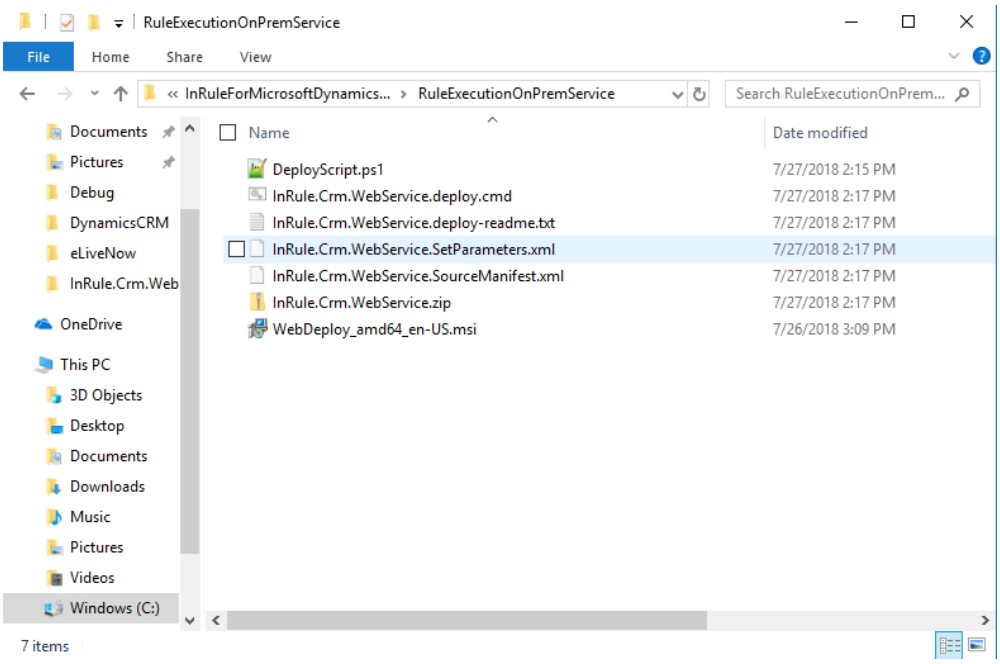
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

4.3.3 Deploy the Rule Execution Web Service

1: Open your "RuleExecutionOnPremService" folder:

For guidance on how to find this folder, reference the "Required Files" section above. The folder will need to be copied onto the IIS server you set up your website on in the previous section.



2: Open the InRule.Crm.WebService.SetParameters.xml file in Notepad or a similar text editor

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

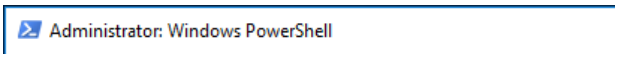
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3: Change placeholder parameter values to real values

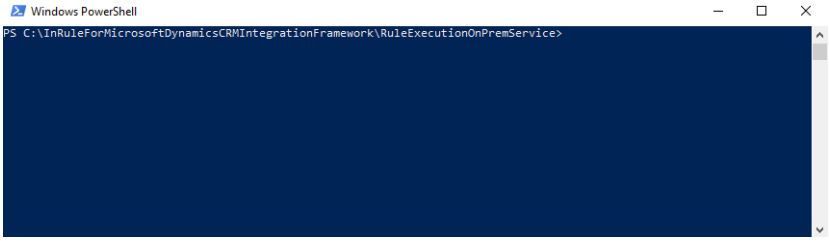
1 IIS Web Application Name	The name of your IIS website as defined in the "Setting up the Catalog Web Service in IIS" section
2 UseInRuleCatalog	Defines if the catalog is being used to store rule apps. Set to true if using the catalog, and false if using file system
3 CatalogUri	The URI for the Catalog Service that will be used
4 CatalogLabel	Defines the label text used to by the service to target the specific rule app in the catalog. Labels are assigned to rule apps in the catalog
5 CatalogUser	Username for Catalog Service, default value is 'admin'.
6 Catalog Password	Password for Catalog Service, default is 'password', please change this using the catalog manager!
7 CatalogSSO	Defines whether to use app pool identity to authenticate to the catalog web service
8 RuleAppDirectory	Specifies the directory where rule apps will be stored if using file system instead of the catalog. If using the catalog, leave as-is.
9 DynamicsCRM-Web.config Connection String	Provide a Dynamics connection string if you are not using S2S authentication, otherwise leave blank. Information on connection string formatting can be found here: https://docs.microsoft.com/en-us/dynamics365/customer-engagement/developer/xrm-tooling/use-connection-strings-xrm-tooling-connect ***Important:*** Some customers have reported needing to provide a username in the format "domain\username" in order to successfully connect using IFD.

4: Save your changes

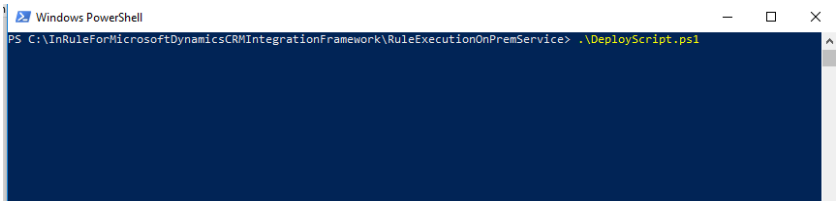
5: Launch PowerShell as an administrator



6: Navigate to the "RuleExecutionOnPremService" folder



7: Run the "DeployScript.ps1" file



Observe that no errors occur while this script does its work.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

4.3.4 Rule Services Solution for Dynamics On-Prem Deployment

At this point, all of the InRule server components are setup. The rule execution service should be listening for incoming communication from Dynamics. We must now setup Dynamics. To make this process easier, we will be using PowerShell.

1: Launch PowerShell as an administrator:

Administrator: Windows PowerShell

2: Navigate to the '`\Dynamics Deployment`' directory:

```
PS C:\work\CRMIntegrationFrameworkTesting\2.3.4.81\InRuleForMicrosoftDynamicsCRMIntegrationFramework\DynamicsDeployment\tools> cd..
PS C:\work\CRMIntegrationFrameworkTesting\2.3.4.81\InRuleForMicrosoftDynamicsCRMIntegrationFramework\DynamicsDeployment>
```

3: Execute Deploy-CrmPackage.ps1

For Dynamics On-Prem v9, run Deploy-CrmPackage.ps1 with the added argument: -OnPrem. If you need to install the Dynamics On-Prem v8.2 compatible version, use -OnPrem_8_2 instead.

version 9.0

```
PS C:\> .\Deploy-CrmPackage.ps1 -OnPrem
```

version 8.2

```
PS C:\> .\Deploy-CrmPackage.ps1 -OnPrem_8_2
```

5: Provide Dynamics On-Prem Service Credentials

You will need to login for this script to continue, please be sure to check the 'Display list of available organizations' to make sure that you select the correct instance of Dynamics to install to!

Alternatively, the Deploy-CrmPackage.ps1 script also accepts a Dynamics Connection String:

```
.\Deploy-CrmPackage.ps1
-CrmConnectionString 'AuthType=Office365;Url=https://irroadsandbox.crm.dynamics.com/;Username=CRM[redacted].com;Password=[redacted]'
```

```
.\Deploy-CrmPackage.ps1 -CrmConnectionString
'AuthType=AD;Url=https://{DYNAMICSURL}.crm.dynamics.com/;Username={DYNAMICSUSERNAME};Password=
{DYNAMICSPASSWORD}'
```

This is a sample connection string; with an on-premise install, there are a wide variety of options that may be relevant, depending on your architecture. For more information on connection string formats, refer to Microsoft's available documentation, found here.

Important: Some customers have reported needing to provide a username in the format "domain\username" in order to successfully connect using IFD.

Observe that no errors occur while the script is executing.

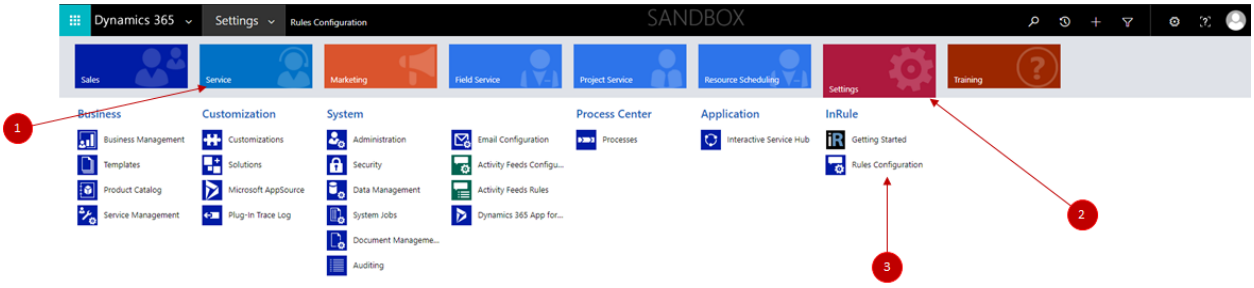
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

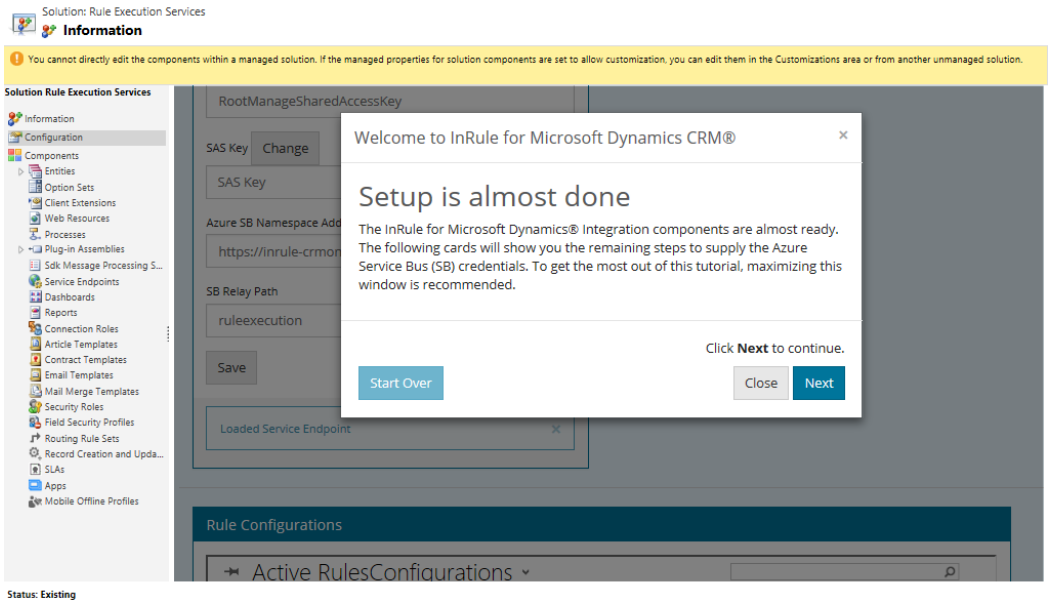
6: Configure the Rule Services Solution in Dynamics

Next, we need to tell the Dynamics Solution that has been deployed where to find the Rule Execution Service deployed earlier. We will need to have the Rule Execution Service URI ready for this step.

Navigate to the Rules Configuration Page



When the Rule Services Solution is entered for the first time, the Welcome to InRule for Microsoft Dynamics 365 wizard will appear:

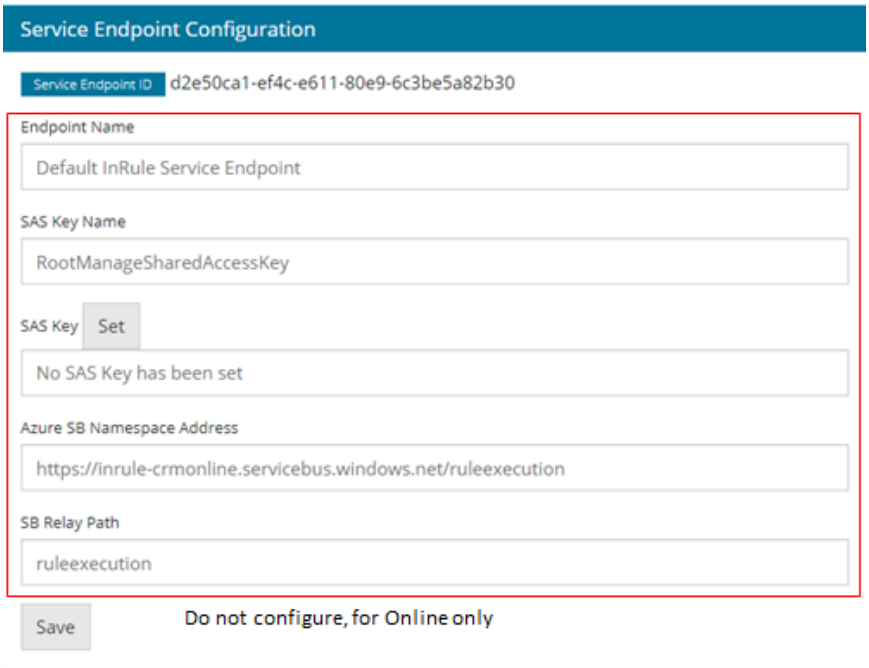


Close the wizard, as it is applicable to Online only.

At the top of the Rules Services Solution configuration page, you will see the Service Endpoint Configuration options for the solution:

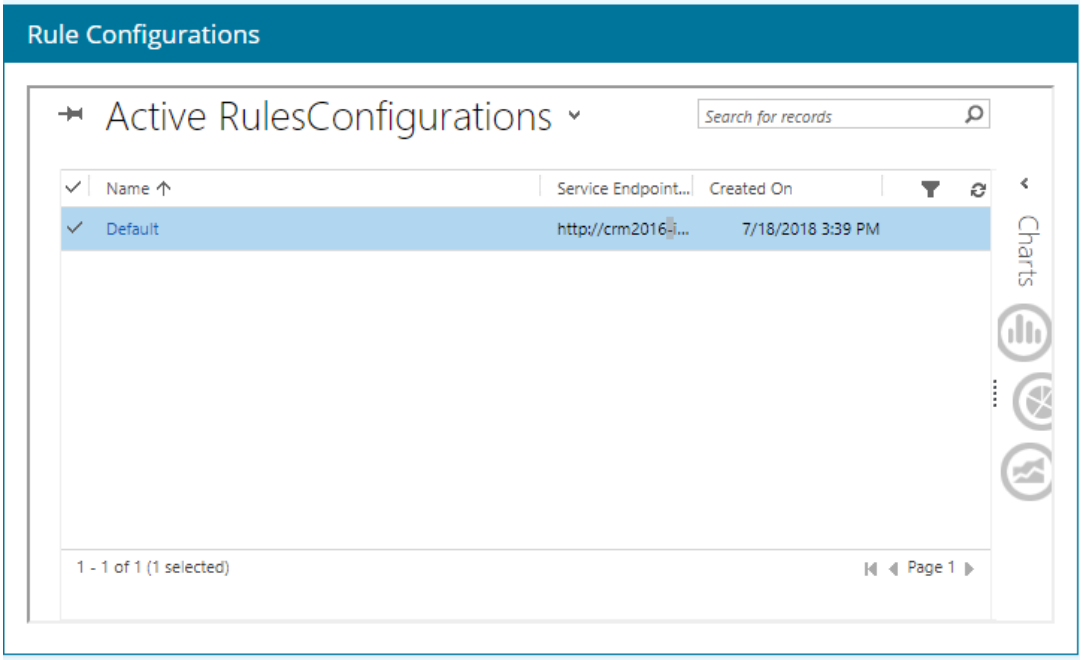
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



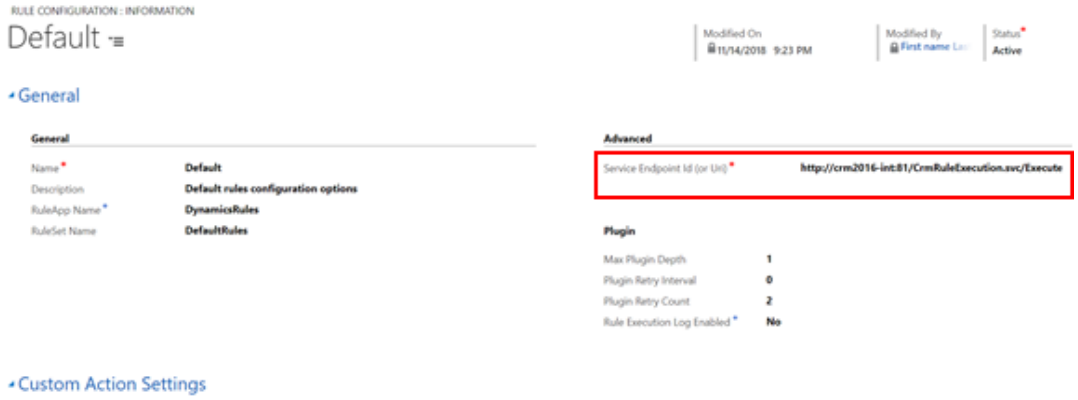
All of these settings can be left as their default values for an on-premise installation, as they pertain to the Online solution only.

Instead of configuring the above, you will need to scroll down all the way to bottom to the “Rule Configurations” section and select the active Rule Configuration

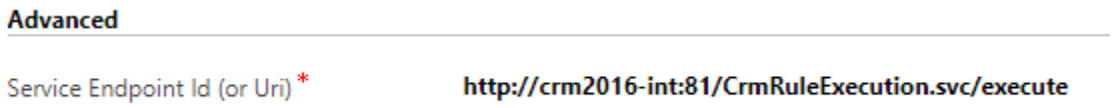


The following popup should appear:

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



You can configure the naming conventions of this rule configuration as you see fit, but under “Advanced” on the right-hand side of the page, you will need to set the Service Endpoint URI to the URI of your Rule Execution Service endpoint



Be sure to click save (the small save button in the bottom-right corner)

7: Verify a successful deployment!

A ‘`Run Rules`’ button will now exist on the edit form of all Dynamics entities. The way this button behaves will depend on other settings that have been configured under the Rule Configuration section; this document assumes the default configuration behavior. To understand what settings are available and what they mean, please see Appendix E: *Methods for Executing Rules from Dynamics 365* of this document.

Open up an Account and execute ‘Run Rules’. The default deployment is configured to run a Rule App from the catalog named `DynamicsRules`, which is what we uploaded to the catalog in the Verify Catalog stage. If everything has been set up correctly, you should see the “Rule Execution Completed” message and the description of the account will be updated to provide the date and time.

It is worth noting that the On-Prem plugin will be deployed **outside** of Sandbox mode. To read more on why this is necessary and the resulting implications, reference Appendix J: *Known Issues, Limitations and Troubleshooting*.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix A: Additional Resources

Having trouble? Relax! InRule offers many additional resources to help you get InRule correctly integrated with Microsoft Dynamics 365.

InRule's Support Website

InRule's support website can be found at http://support.inrule.com. If you do not already have a login for our support site, the client administrator at your company has the ability to create an account for you. If you are unsure of who your client administrator is, please email support@inrule.com.

The InRule Support Website has three different areas that readers should be aware of:

1. Downloads
2. Documentation
3. Forums

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

The Downloads section

The download section of the Support Website contains numerous resources that will likely be of interest to you:

InRule Installer*

Filed under the 'InRule' section of the downloads page, The InRule installer package will launch a graphical user interface that will walk the user through installing many of InRule's offerings. These offerings include both irAuthor which is used to edit Rule Applications, and irX for Microsoft Dynamics 365, which is an extension to irAuthor that allows synchronizing schema between Rule Applications and Dynamics 365.

*It is recommended that the reader of this document has established a rule-authoring environment by installing both *irAuthor* and *irX for Microsoft Dynamics 365* before attempting to deploy the *InRule for Microsoft Dynamics 365 Integration Framework*.

irX for Microsoft Dynamics 365 Help Documentation

Filed under the 'InRule for Microsoft Dynamics 365' section of the downloads page. This document discusses how to utilize the *irX for Microsoft Dynamics 365* *irAuthor* extension to connect to Microsoft Dynamics 365 in a rule-authoring environment to synchronize rule application schema objects against Microsoft Dynamics 365 entities and to test Rule Applications directly against data coming from the Microsoft Dynamics 365 environment.

InRule for Microsoft Dynamics 365 Integration Framework Deployment Guide

This is the guide that you are reading now. The intention of this guide is to give a complete reference for those wanting to get the Integration Framework up and going as quickly as possible without deviating from what we consider to be the most common deployment scenario.

This is a fantastic place to start if you already have some familiarity with *irX for Microsoft Dynamics 365* and are ready to deploy the *InRule for Microsoft Dynamics 365 Integration Framework* as a runtime companion.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

InRule for Microsoft Dynamics 365 Integration Framework

This downloadable archive contains all of the following located off of the root:

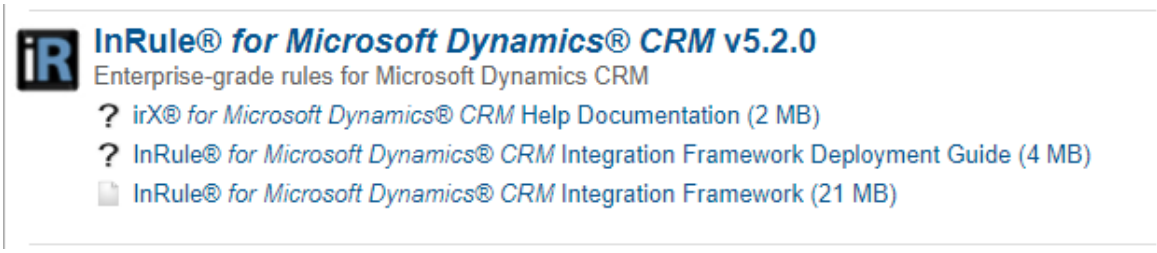
- **Rule Applications \ Dynamics Rules.ruleapp**: This Rule App is intended to be used both to test a basic deployment of the Integration Framework, but also to serve as a starting point for authoring further rules against your Dynamics 365 environment.
- **Rule Execution Azure Service**: This folder contains the app service package and configuration files that will be needed to deploy the Rule Execution App Service to Azure as discussed elsewhere in this document.
- **Rule Execution OnPrem Service**: This folder contains the web app and configuration files that will be needed to deploy the Rule Execution on-prem service as discussed elsewhere in this document.
- **RuleHelper Deployment**: This folder contains the .NET assemblies necessary to utilize the Rule Helper library to dynamically query Dynamics 365 for information during the execution of rules. RuleHelper is included with the Rule Execution Service, but also made available here for deploying to irAuthor's EndPointAssemblies folder
- **Dynamics Deployment**: This folder provides all of the resources needed to deploy the necessary Solution, Plugin, Endpoint, and Step registrations required to enable Dynamics 365 to send Rule Execution requests to an Azure Service Bus with an attached Rule Execution App Service.
- **ReadMe.txt**: This file contains some information about this Integration Framework, including historic release notes, product version information, and license information.

Catalog App Service – Azure Package and Config File

Found in the 'InRule for Microsoft Azure' section of the downloads page, this archive contains both the app service package and app service configuration files that you will need to perform an Azure based installation of irCatalog.

Catalog App Service – Install Documentation

This document will discuss how to install InRule's irCatalog in Microsoft Azure using a PaaS model.



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

The Documents section

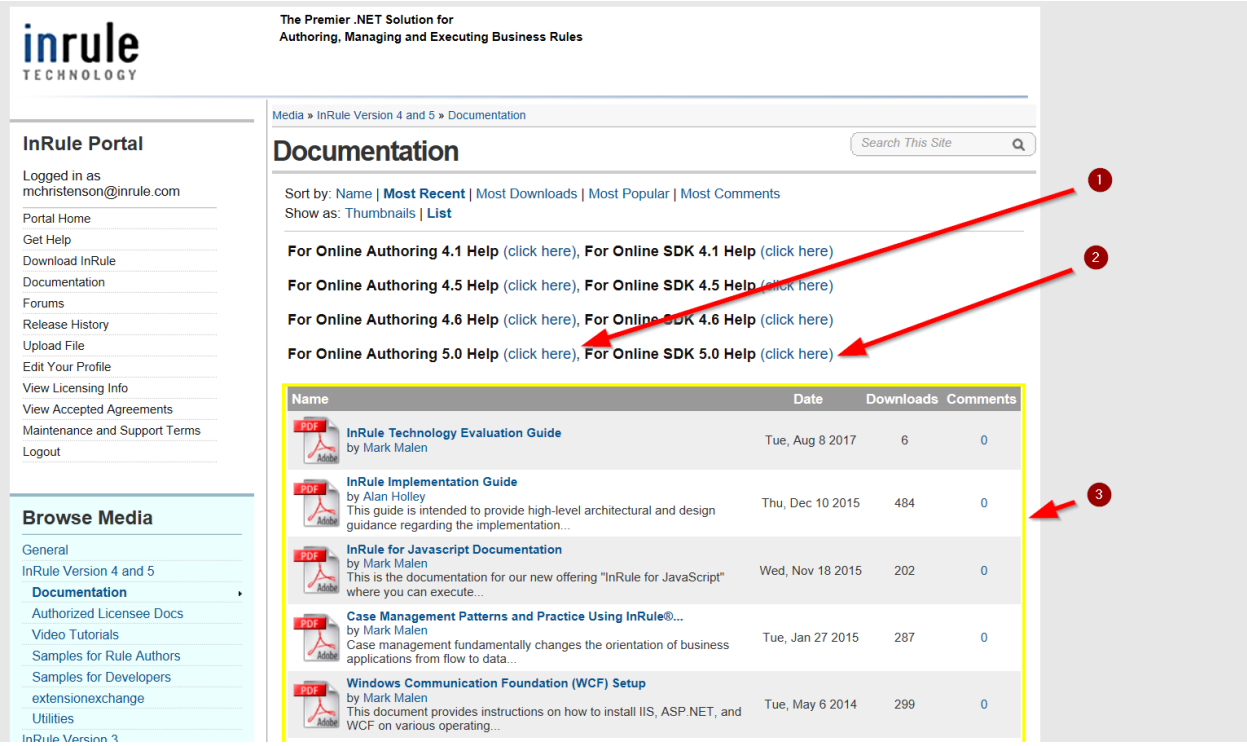
The documents section of the Support Website contains additional documentation about how the various components to InRule operate.

Although the documents section does not contain documentation that discusses integration with Dynamics 365 (that documentation can be found in the downloads section), it does contain the following three areas of interest:

1. Online Authoring Help: This document discusses anything you would need to know about how to author Rule Applications. The majority of this document applies to utilization of irAuthor, InRule's custom graphical authoring product.

2. Online SDK Help: InRule exposes an extensive .NET based API. This document discusses the use of the full SDK for use by software developers that would like to integrate with InRule.

3. Miscellaneous documentation section: Here you will find a variety of special interest documents, such as an installation guide, an implementation guide, and a guide that focusses on InRule's UDF language, irScript.

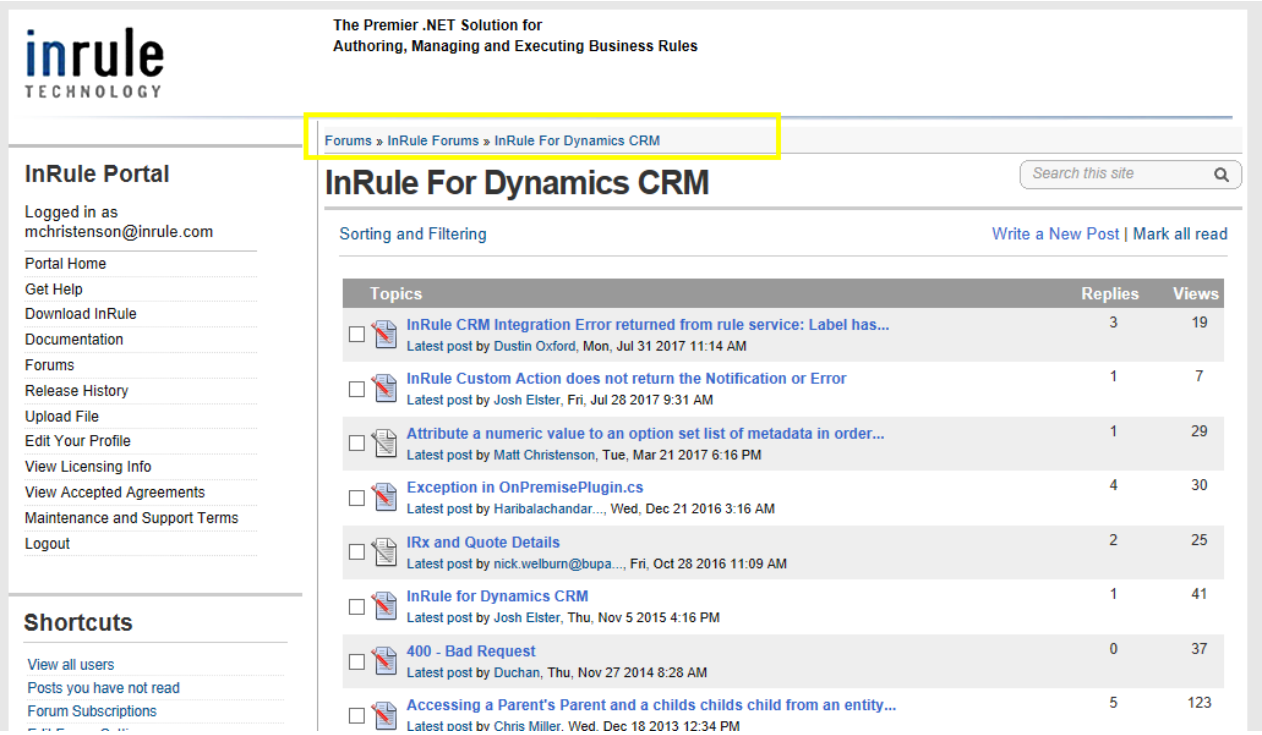


Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Online User Forums

Numerous internal resources at InRule monitor InRule Forums. This is the first place to go if you want to benefit directly from **the helpful and knowledgeable people** at InRule.



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

InRule's Support Team

The support team at InRule is available to help with any product support needs, troubleshooting suspected product bugs, resolving any licensing issues, and free tele-hugs.

The best way to reach Support is through a detailed email sent to support@inrule.com.

You can also reach our support team by calling +1 (312) 648-1800.

InRule's ROAD Team

ROAD Services agreements can be used to engage with ROAD, InRule's professional services team.

ROAD can provide your organization with specialized consulting and tailored Architecture and Authoring Guidance.

ROAD can assist with less common installation requirements, such as deployment to third party cloud providers or integration with custom software.

ROAD can be contacted by emailing ROADServices@InRule.com

InRule Training Services

InRule offers the following interactive training services:

- Onsite and Remote attendance courses
- Hands-On multi-day courses with interactive labs
- Virtual Express training courses delivered online for rapid knowledge transfer

If you are interested in scheduling training services, please contact us at Training@InRule.com.

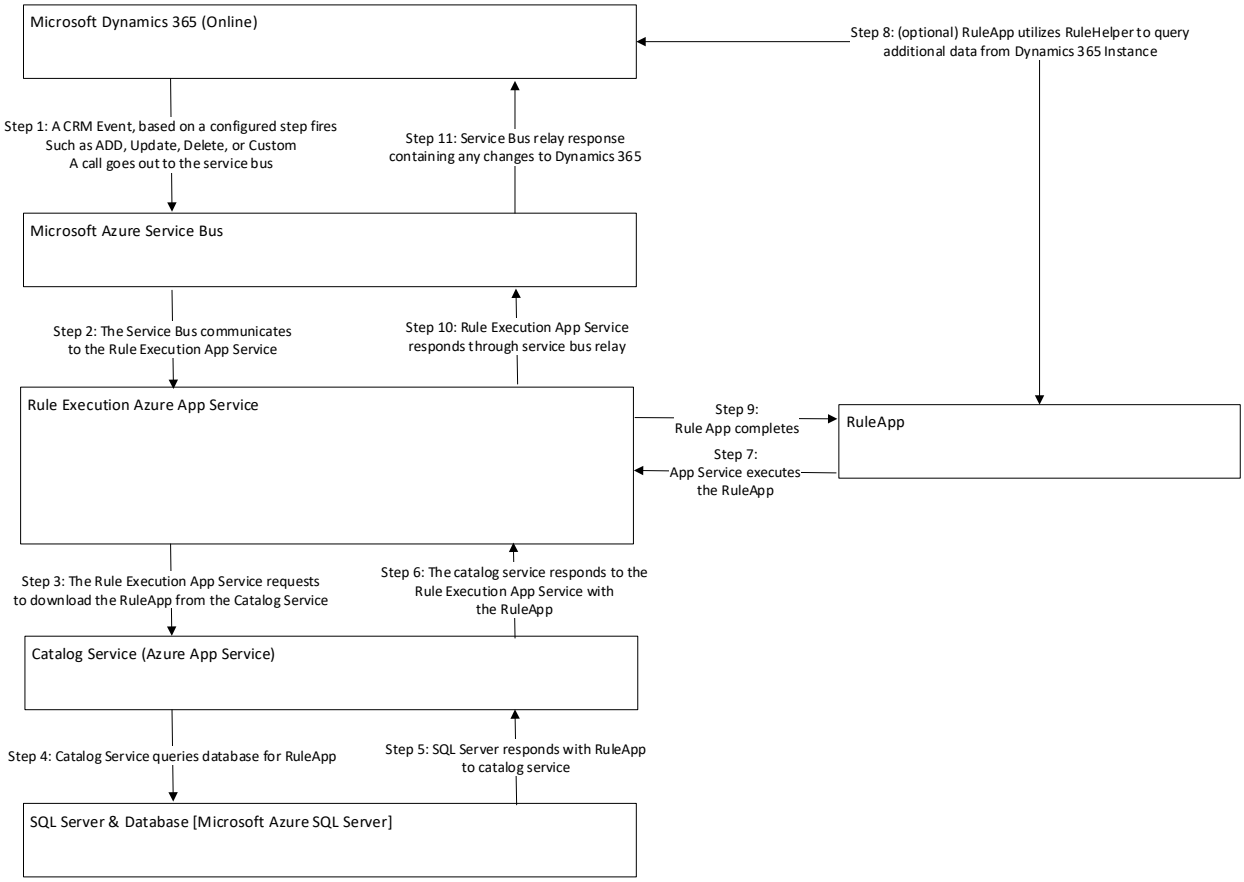
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix B: Anatomy of a Request for Execution of Rules Diagram

The below diagram helps to give a top-level understanding of how InRule is integrated with Microsoft Dynamics 365 (Online). Please note this this diagram is a simplification that does not cover topics like caching, iterations, and multiple environments. It serves to show how the request moves through different Azure resources.

1. A Dynamics 365 Event based on a configured step fires, such as ADD, UPDATE, DELETE, or CUSTOM. This generates a call to the Service Bus, which is setup to relay requests to the connected Rule Execution App Service.
2. The Service Bus receives the request and relays it to the Rule Execution App Service, which has attached itself to the Service Bus as a relay listener. For On-Premise, the rule execution service interacts directly with Dynamics.
3. The Rule Execution App Service makes a request to the Catalog Service, asking for a copy of the requested RuleApp.
4. The Catalog Service Queries its SQL Server based database for a copy of the requested RuleApp.
5. The SQL Server responds with the RuleApp.
6. The catalog service responds to the Rule Execution App Service with the RuleApp.
7. The RuleApp executes inside the Rule Execution App Service.
8. Optionally, the RuleApp has an opportunity to query Dynamics 365 for additional data needed to execute rules.
9. The RuleApp completed execution
10. The Rule Execution App Service responds through the Azure Service Bus
11. The Azure Service Bus relays the response to Dynamics 365, where the receiving plugin synchronizes changes.



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix C: irX General Integration Concepts

Runtime Mapping across Nested Relationships

Much like Dynamics 365, the InRule rule engine offers strong support for hierarchical and relational data. Within a given Rule Application, data can be considered across parent-child relationships within a single rule request. These relationships can take the form of Collections (1 - * relationships) or 1 – 1 relationships.

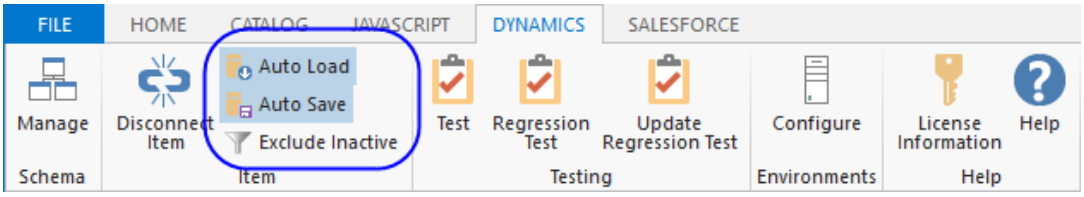
Note: When N:N relationships are imported into InRule, they behave as 1:N Collection relationships within the Rule Application.

In addition to the abilities of both products to handle relational data, both products also offer the ability to declaratively configure “Entities” and “Fields”. Both products also allow for different strongly-typed Entities and Fields to be accessed with loosely-typed SDK interfaces. Because of these inherent similarities and flexible interfaces, it is possible to build a reusable mapping component that can convert any given graph of loosely-typed Dynamics Entities to InRule Entities, and vice versa.

Controlling irVerify Behavior with Load, Save and Inactive Record Settings

When working with a tree composed of many related Dynamics Entities, it is often useful to have explicit control over which relationships are either automatically loaded or automatically considered in change detection for persistence. If a relationship is skipped during the initial load routines, then it is available to be conditionally populated later using rules.

In the irX rule authoring ribbon, there are three buttons that give the rule author the control to denote if a relationship should be automatically loaded, saved or have inactive records excluded.



Note: Automatic loading and saving is enabled by default for all relationships that are imported from Dynamics. The rule author can opt-out of these automatic behaviors by unselecting “Auto Load” or “Auto Save”. When these buttons are selected, metadata attributes are written into the Rule Application for the given relationship. These metadata attributes are used by the irVerify data loader when recursively loading data or detecting changes for persistence.

Important: If loading or saving is disabled for a given relationship, then it is also disabled for all Entities that are children of that relationship.

Since Microsoft Dynamics 365 uses the "inactive" status to do a soft delete of records, it is possible to include both active and inactive records when loading a Collection or relationship. You can use the "Exclude Inactive" button to exclude inactive records, returning only active records. To ensure that this feature works correctly, the child Entity of the Collection or relationship should map the Dynamics 365 **Status** field.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix D: Accessing Dynamics 365 Directly from Rule Helper

In the default Rule Execution setup, all relationships between Entity types must be established before these entities can be used by the rule app. This behavior is intuitive, but it is not ideal for all business problems. The InRule Integration framework provides a 'Rule Helper' assembly that can be used directly in a rule app and allows rules to load, compare, and assign data that is not related in Dynamics before rules are executed.

When to use the Query from Rules Approach

The query from rules approach adds value for the following business problems:

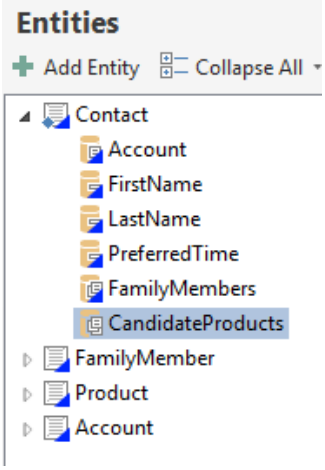
- The rules need to reference "lookup" information that may be in a list or set of Entities that are not specifically related to the current Entity hierarchy
- The purpose of rules is to create new relationships between Entity instances that already exist in Dynamics
- The rules need to compare many combinations of unrelated Dynamics Entities and produce results about best possible matches or scores
- A custom filter is required when loading data for 1:N or N:N relationships

Working with Disconnected Fields when Loading and Saving Data

One of the most important integration concepts when loading Dynamics data from rules is the notion of "Disconnected" Fields and Fields that have "Auto Load" and "Auto Save" disabled.

irX allows the rule author to explicitly control the "Auto Load" and "Auto Save" behaviors of Fields that are connected to Dynamics.

The example below shows a Collection named CandidateProducts. Since the Collection is not marked with a blue triangle, it is not considered to be attached to Dynamics.

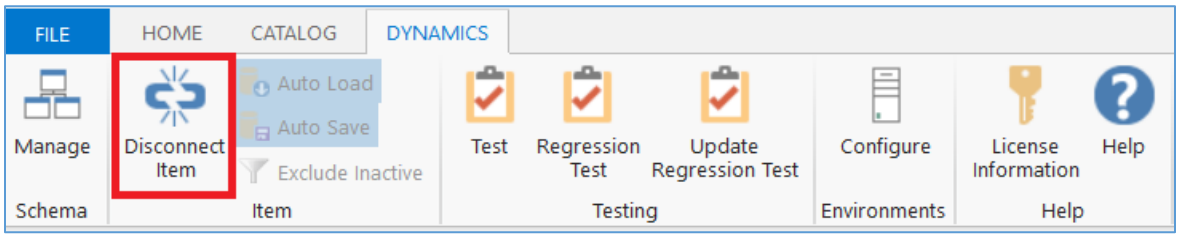


Important: Although Entity Fields and Collections may be "Disconnected" from Dynamics, the types contained by the Collections can be set to types that were imported from Dynamics.

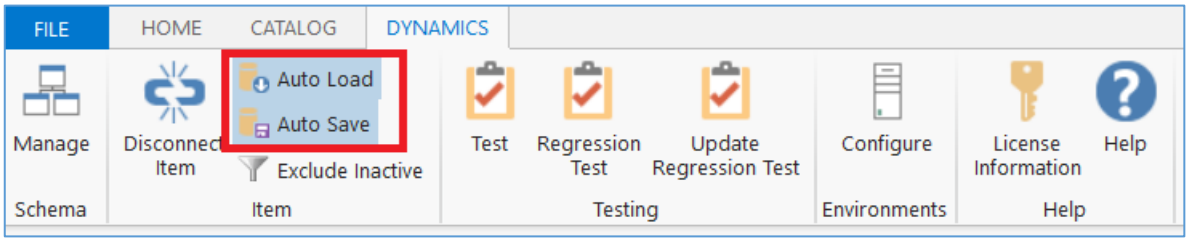
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Note: If a Field is added to the schema using irAuthor, then it will be disconnected from Dynamics. If a Field has been imported from Dynamics using irX, then it can be disconnected from Dynamics by clicking the “Disconnect Item” button in the irX ribbon.



Important: Two additional settings appear in the irX ribbon that offer additional control over automatic loading and saving behaviors for Fields that remain connected to Dynamics. In the example below, both buttons are “lit up”, which denotes that the settings are enabled. By default, automatic loading and saving is enabled for all Fields that are connected to Dynamics.



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

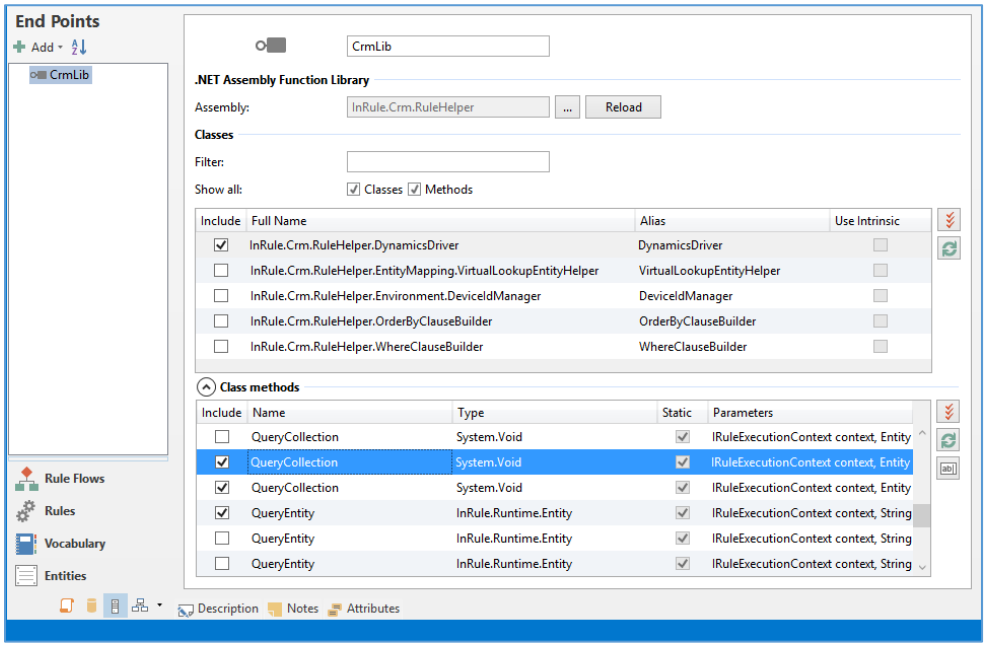
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Integrating the Rule Helper Component

InRule provides a sample rule application (DynamicsRules) that is already configured for RuleHelper usage. You can simply edit this rule app, or, if you wish to integrate RuleHelper into an existing rule app, you can copy both the UDF Library "RuleHelper" and End Point 'CrmHelper" from the DynamicsRules rule to another rule application.

If you wish to manually create the UDF Library and End Point in irAuthor, follow the steps below.

1. Create a new rule app using the irX add-in for irAuthor.
2. Create a new ".NET Assembly Function Library" end point and bind the end point to the InRule.Crm.RuleHelper.dll assembly. Select the DynamicsDriver class and then select the methods that should be callable from rules. Edit the name of the end point to "CrmLib" or similar. Select the methods from the DynamicsDriver that are needed for the Rule Application. You do not need to select all the methods—only import the methods that will actually get used by rules. Additional methods can always be imported later be revisiting the endpoint screen and reloading the assembly.



3. Add a User Defined Function library and set the name to "CrmHelpers" or similar. This library will contain functions that the rules will call to query Dynamics.
4. Add a User Defined Function to the new library. The example below shows a UDF that will be used to execute the QueryCollection method on the DynamicsDriver. Fill out the UDF with script that will call a method on the DynamicsDriver.

Note: The methods on the DynamicsDriver are designed to be reused for more than one Entity type, Field, or set of Fields. The name of the Target Field or Collection should be supplied as a string. When querying a Collection of results, an optional "where" clause can be provided that will be forwarded to calls against the CRM SDK. In addition, an "order by" clause can be provided to return sorted results.

Important: This integration pattern relies on the "Context" object that is available from irScript. The Context object returns information based on the context under which a given UDF is executed. For example, when executing an Entity Rule Set, the Context.Entity returns

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

a reference to the Entity against which the current Rule Set is executing. The Context and its child properties are passed to the DynamicsDriver so it has enough information to form calls to Dynamics 365 and map responses back to the InRule Rule Session.

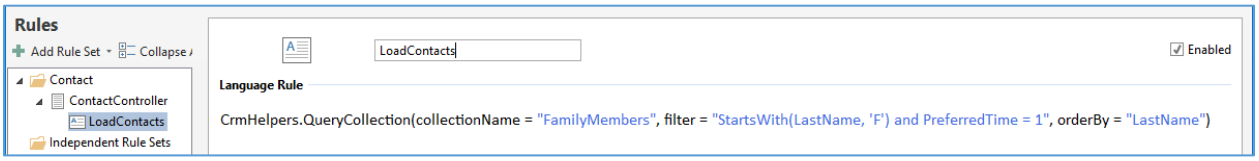
Note: The Context.FunctionLibraries property can be used to create calls to the .NET assembly library methods, such as the methods imported in Step 5 above. The following script example demonstrates how to use the Context object in irScript to form a call to a static .NET method:

```
Context.FunctionLibraries.DynamicsDriver.QueryCollection(Context,
Context.Entity, collectionName, filter, orderBy, connectionString);
```

Important: The "connectionString" argument is optional due to overloading of the methods on the DynamicsDriver class. If the connectionString is not passed in the by the rule engine, then it will be looked up from either the .NET config file based on the given environment. A sample connection string that need to be defined in the irAuthor.exe.config XML is included below. For information on Dynamics connection string formatting, refer to https://docs.microsoft.com/en-us/dynamics365/customer-engagement/developer/xrm-tooling/use-connection-strings-xrm-tooling-connect

```
<connectionStrings>
  <add name="DynamicsCRM"
  connectionString="AuthType=Office365;Username=jsmith@contoso.onmicrosoft.com;
  Password=passcode;Url=https://contoso.crm.dynamics.com"/>
</connectionStrings>
```

5. Rules can now be authored to execute methods on the DynamicsDriver. These methods can be used to load Collections, single Entities, or single Fields from Dynamics based on conditional logic within rules.



Note: The example above includes a call to the default business language templates for a method on the DynamicsDriver. The InRule vocabulary features can be used to modify these templates to be more user-friendly for business users.

Important: The Target Collection in the sample rule is called "FamilyMembers". This is a Field that either does not exist in Dynamics (only for use in rules), or has been imported and then "disconnected" from Dynamics using the "Disconnect Field" button, or has "Auto Load" disabled.

Note: Please see the following sections for more details on the creating the "filter" clauses similar to the one used this example.

Filtering Queries using the Where Clause Builder

When loading data from Dynamics during rule execution, it is critical that the rule author is able to author logic to specify which Entity data to load. Using the RuleHelper, this is accomplished by allowing the rule author to pass in a "filter" or "where" clause into the calls against the DynamicsDriver class.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

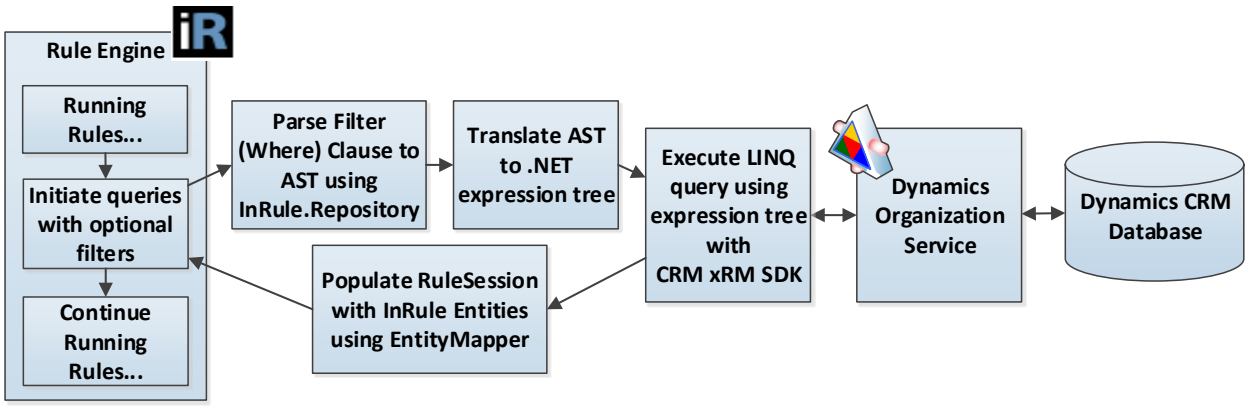
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

During execution of the DynamicsDriver, the filter clause is parsed into an Abstract Syntax Tree (AST) and then translated into a LINQ expression tree, so it can be consumed using the CRM SDK. The filter clause is based on the InRule function syntax format.

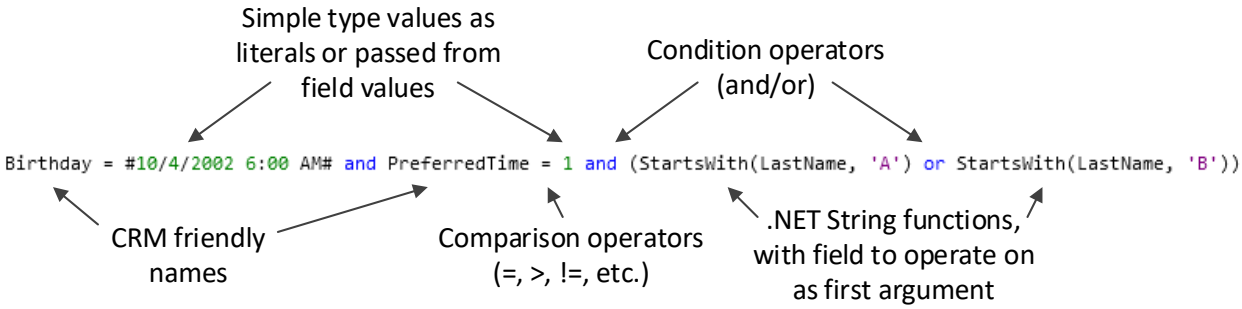
Note: The InRule function syntax format is used for the following reasons:

- The syntax rule format is consistent with the rule authoring experience used throughout irAuthor
- This format can make good use of the InRule AST parser that is included as part of irSDK

The diagram below depicts the logical flow of steps used by the DynamicsDriver and WhereClause builder classes to query data from rules.



The diagram and notes below contain some additional information about forming the filter clause in a rule:



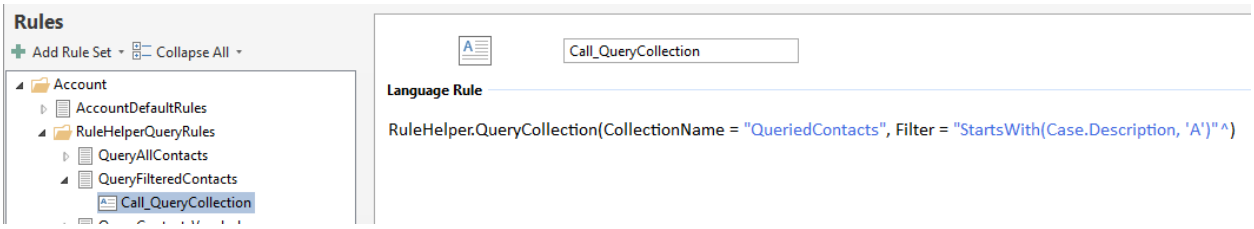
- All the Field names that are used are the Dynamics friendly names that are used in the Rule Application. These names are mapped back to the Dynamics Field names in the parser.
- String literals should be wrapped in single quotes, date literals should be wrapped in pound signs.
- Simple operators are supported to compare values, such as =, !=, >, < (ex. Age > 21, Name != 'Ralph').
- Multiple conditions can be chained together using 'and' and 'or' keywords.
- The expression tree builder will attempt to call a given .NET framework function if it appears in the expression.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

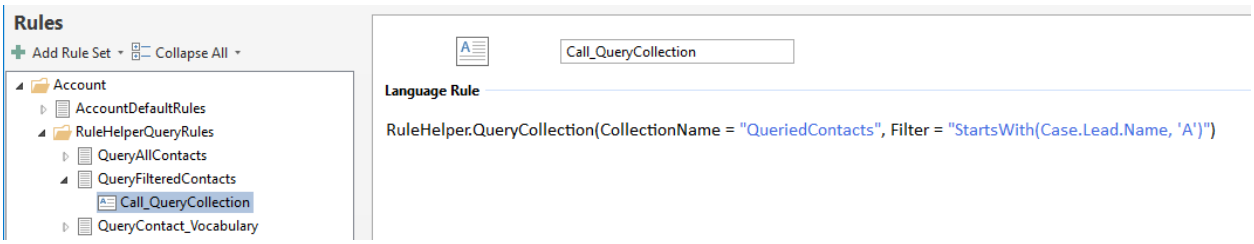
- The .NET call must be expressed as a function, with the first argument being the name of the Field to operate on. The remainder of the arguments will be passed through in order to the .NET function call when it is formed.
- Examples: StartsWith(FirstName, ‘A’) EndsWith(LastName, ‘ez’) Contains(LastName, ‘Smith’)
- Any non-static methods on the .NET String class are candidates to be called as functions using the parser syntax above. However, InRule has only tested the StartsWith, EndsWith, and Contains methods on the String class.

- The following keywords and operators are supported by the InRule AST parser and expression tree translation code: **=, <>, !=, +, -, *, /, or, and, xor, >, >=, <, <=, ^, %**

The filter expression also supports querying against related entities, simply by appending the related entity name in front of the relevant query field. Querying against related entities requires that all **entities** queried in the relationship chain be imported into irAuthor. The queried **fields** on each of the queried entities do not necessarily need to be imported, but if they are not, the entire Dynamics schema field name must be used, rather than their irAuthor aliases.



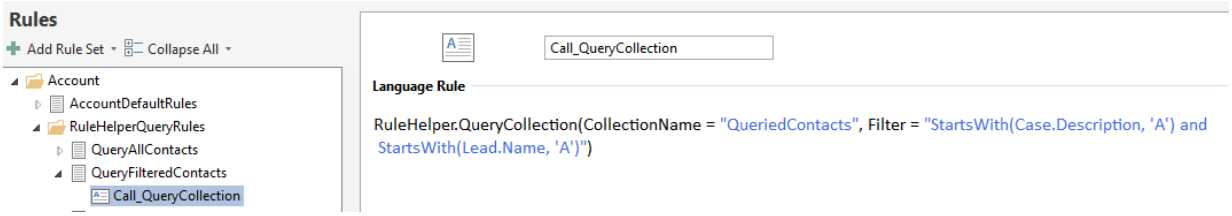
In this example, we are populating a collection by querying the Contact entity, which is the “parent” entity here. We are then applying a filter statement to return only contacts with related Cases that have Descriptions starting with “A.” Cases in this example is the “child” entity. Notice how the hierarchy of the related entity down to field is denoted. If you wanted to drill down another layer to a “grandchild” entity (in this example, an entity related to Case), it would be accomplished by simply continuing the chain from entity to field. Below is an example of a “grandchild” case:



This example would return Contacts that have related Cases with related Leads with names starting with “A.” The filter expression can support querying in this manner up to 10 “layers” deep, including the initially queried entity. Put another way, you can have up to 10 total different related entities in a single filter expression.

The filter expression supports querying against multiple properties from different related entities. In the below example, we are querying for Contacts with Cases that have Descriptions starting with “A” and also have Leads with Names starting with “A.”

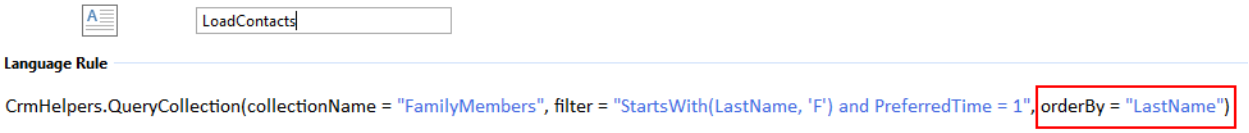
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



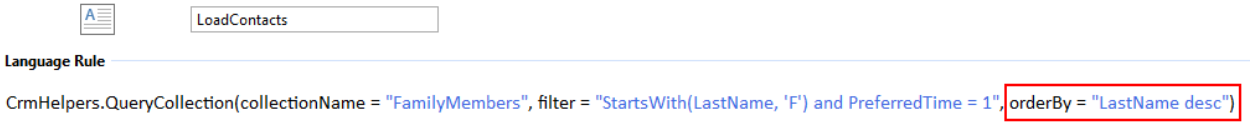
Ordering Query Results with the OrderByClauseBuilder

The DynamicsDriver class also supports the ability to control the order of the results returned from Dynamics by passing in an optional “order by” clause. The order by clause can accept only a single Field name, which should be the name of the Field in the Rule Application. The results are always sorted in ascending order, unless the Field name is followed by the “desc” syntax. Please see the examples below:

To sort ascending, pass the Field name to use in the sort:



To sort descending, pass the Field name to use in the sort followed by the “desc” keyword:



Note: The “order by” clauses generally contain much simpler expressions than “where” clauses. However, InRule syntax rules format is used for the order by clause to be consistent with the where clause approach.

Important: The parsing and translation of the order by clause is handled in the Rule Helper contained in the OrderByClauseBuilder class.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Methods Available in the Rule Helper

The following table lists the public, static methods that are available in the DynamicsDriver

Method Name	Description
`LoadMappedChildCollection`	Populates a child Entity Collection based on an existing 1:N relationship in Dynamics. The Collection is populated based on existing parent-child relationship data in Dynamics.
`LoadMappedChildEntity`	Populates a child Entity Field based on an existing 1:1 relationship in Dynamics. The Field is populated based on existing parent-child relationship data in Dynamics.
`QueryCollection`	Populates an Entity Collection with a set of a given Entity type. An optional filter clause (where clause) can be used to define selection criteria for the Entity set. The Collection does not need to correspond to a 1:N relationship in Dynamics.
`QueryEntity`	Populates an Entity Field or variable based on a query to Dynamics. An optional filter clause (where clause) can be used to define selection criteria for the Entity. The Field does not need to correspond to a 1:1 relationship in Dynamics. If more than one Entity is returned from the query to Dynamics, then the first Entity in the set is used.
`QueryField`	Populates a primitive Field or variable based on a query to Dynamics. An optional filter clause (where clause) can be used to define selection criteria for the Entity. If more than one Entity is returned from the query to Dynamics, then the Field value from the first Entity in the matching set is used.
`QueryNtoNCollection`	Loads entities across an N:N Collection that has been defined in Dynamics. The relationship name and parent and child ID Fields must be provided.
`LoadMappedNtoNCollection`	Similar to QueryNtoNCollection, except that the relationship must be imported into the Rule Application.
`GetUserLocalTimeFromUtc`	Converts a Dynamics UTC time value into the local time for a given user. This method makes use of the LocalTimeFromUtcTimeRequest that is part of the CRM SDK.

Additional Flags Available to Control Loading and Caching Behaviors in the Rule Helper

During a given query operation, there may be advanced use cases that require specific control over loading or reloading data from Dynamics 365. The optional overloads of the QueryEntity and QueryCollection methods expose a set of optional Boolean flags that help control caching and depth of loading behaviors. The table below list these parameters:

Parameter Name	Description
`loadChildren`	Denotes if the execution service should recurse the Entity graph and load all children. If false, no children are loaded below the Collection Members that are loaded. The default value is true.
`useCaching`	Denotes if previously loaded Dynamics Entities should be reused from the InRule entity cache, or if new Dynamics instances should be created. If false, a copy of the Entity is created, and its Instance ID is not set to the GUID of the Dynamics Entity. The default value is true.
`overwriteIfLoaded`	Denotes if a previously loaded Dynamics Entity should be repopulated with the latest values in Dynamics. This behavior will overwrite Field values stored in the cache. The default value is false.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

`cacheInAppDomain`	Denotes if the result of the query should be saved in the persistent AppDomain cache. The difference between this parameter and the 'useCaching' parameter above is that enabling this parameter will save the query result in a cache that will persist across multiple different rule executions, where the above parameter only enables caching within the scope of a single rule execution. For more information, refer to Configuring the AppDomain Cache

Note: The default values should always be used for the cache settings unless there is a specific use case that requires different behaviors.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix E: Methods for Executing Rules from Dynamics 365

InRule for Microsoft Dynamics 365 provides several methods for the execution of rules. The default and most obvious method included with the solution is the 'Run Rules' button, which is enabled by default on every Dynamics entity form. This allows users to easily execute rules on demand, but other options are also available, which are listed below. Detailed instructions for how to configure these methods are included in the linked sections:

1) Plugin Events
2) Run Rules Button
3) Workflow Activity
4) Form Events
5) Custom JavaScript
6) Custom Action

All of these options can be used together, and each have factors to consider when choosing which to use. Some of the key considerations are outlined below.

User Control

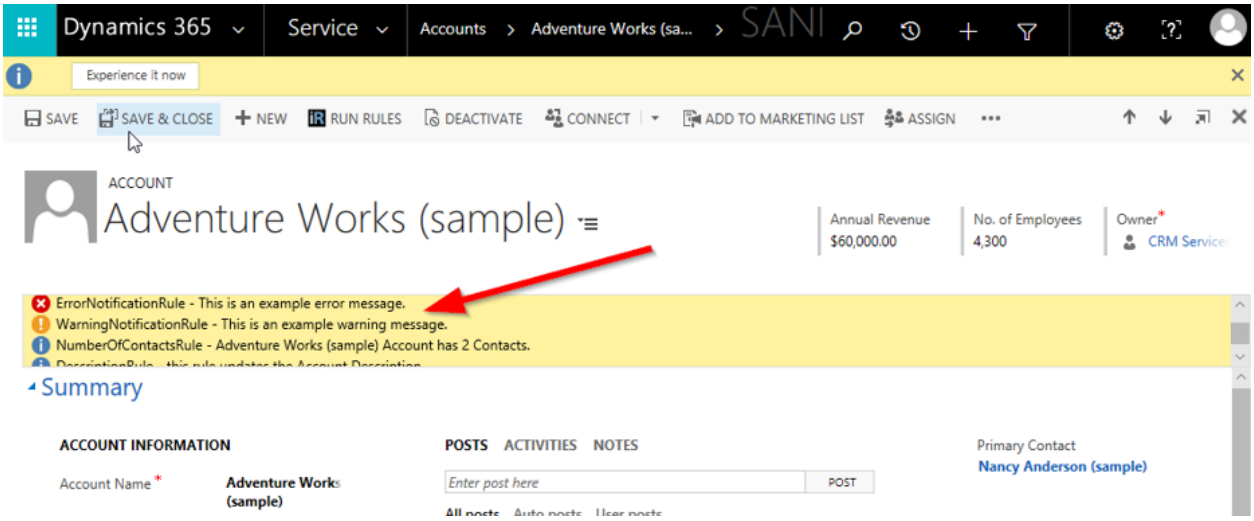
If users need to be able to execute rules on demand, the Run Rules button and Workflow activity are the easiest choice to work with. The Run Rules button is displayed on the command bar of every entity, so it can easily be clicked at any time by end users to get feedback from rules. Using the custom workflow activity also allows rules to be run on demand, potentially as part of a more complex process that contain a rules processing step. Rules can be invoked indirectly by the user by tying rule execution to form events. This can cause rules to be run on field change or form save, for example. Custom JavaScript can also be used for more advanced form scenarios, like integrating with another command bar button. The Custom Action can even be used to respond to user events outside Dynamics, by providing a web API endpoint that can be accessed by external systems.

Display in the User Interface

If you want to display information to the end user based on rule execution, the run rules button is the easiest method. When executing rules from the button, any Errors, Warnings and Informational notifications will be shown in the notification pane, along with any Validations. Executing rules via the included form events helper function or calling the executeRules function from custom JavaScript will also display the same information.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Using the custom workflow activity itself will not display any data in the UI, but the workflow activity does provide output variables with the results of rule execution that can be displayed in a dialog or output via some other method if desired.

Plugin events will not typically display any info in the UI, but any errors or validations will show up in the plugin error pop-up if the plugin is set to run synchronously.

Execution Against Dirty Data

If you need to execute rules against dirty form data, or any entity state other than what is saved to the database, you can enable the 'Use Dirty Entity Image' setting as described in An Explanation of the InRule Custom Action Options. If this option is enabled when running rules from the 'Run Rules' button, the current form values will be collected and sent to the rules. This can be used to perform form value validation, or to provide insight into a what-if scenario prior to changes being saved. This same config value also governs the behavior for form events or any custom JavaScript that calls executeRules.

The custom workflow activity and plugin events do not provide the ability to specify a dirty entity image. They can only run against the data provided by the plugin pipeline. However, in the case of a plugin entity update event the entity change image is automatically included. In addition, the custom action provides a dirtyEntityImage field that accepts a json-serialized entity image.

Responding to API Events

If you want to always execute rules when an entity changes, no matter how those events are initiated, you need to register the included plugin to an event. Running rules based on plugin events means that not only will rules be executing when creating or saving entities from within the Dynamics 365 Entity Forms UI, but they are also executed when entities are changed through API calls.

API calls are frequently consumed by 3rd party add-on user interfaces, Extract-Transform-Load (ETL) synchronization with other software products, and by custom software solutions as an integration point with Dynamics 365. By utilizing this option, an implementer can take advantage of rule execution for needs beyond the Dynamics 365 Entity Forms UI.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

1 Plugin Events

The Rule Services Solution contains a plugin that can act as a handler for events fired by Microsoft Dynamics 365. By registering new steps to Create or Update events against Microsoft Dynamics 365 Entities, rules execute against the entities that are associated with an event as it occurs.

Configuring Plugin Events

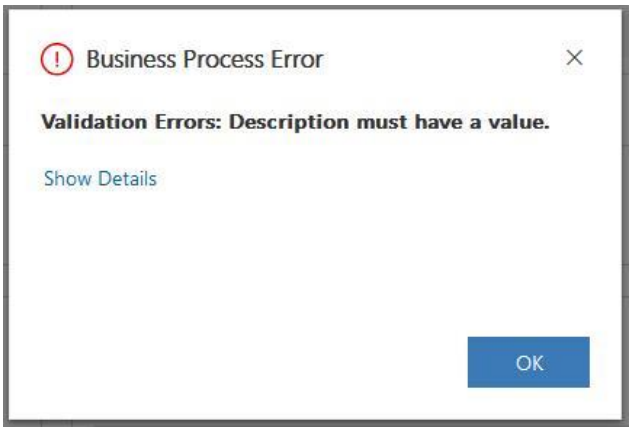
You can register plugin steps using the InRule Rules Configuration page included with the Rule Services Solution. Alternatively, for advanced scenarios, you may use the Plugin Registration Tool from the Microsoft Dynamics 365 Software Development Kit (SDK), but this is typically only needed for special circumstances where greater plugin management is warranted.

There are two steps required to register a plugin event:

1. **Create a Rule Configuration Record** (or use the Default Rule Configuration) - create or update a Rule Configuration record according to the steps in Updating InRule Rules Configuration Records
2. **Associate the configuration record** to the create or update message of an entity by following the steps in Associating an InRule Configuration record to an Entity

Validation and Cancellation

When running rules from a plugin, you may want to cancel any changes that have been made to the entity as well as any changes made from rules. One way to accomplish this is by returning a validation error from the rules. If the plugin detects any validation errors, it will skip saving any changes made by the rules and stop the plugin pipeline to prevent any further changes. If an end user is saving or creating an entity and this happens, they will see the following error:



For a better validation user experience, you can also run rules on the 'OnSave' event of a form, as described in the Form Events section.

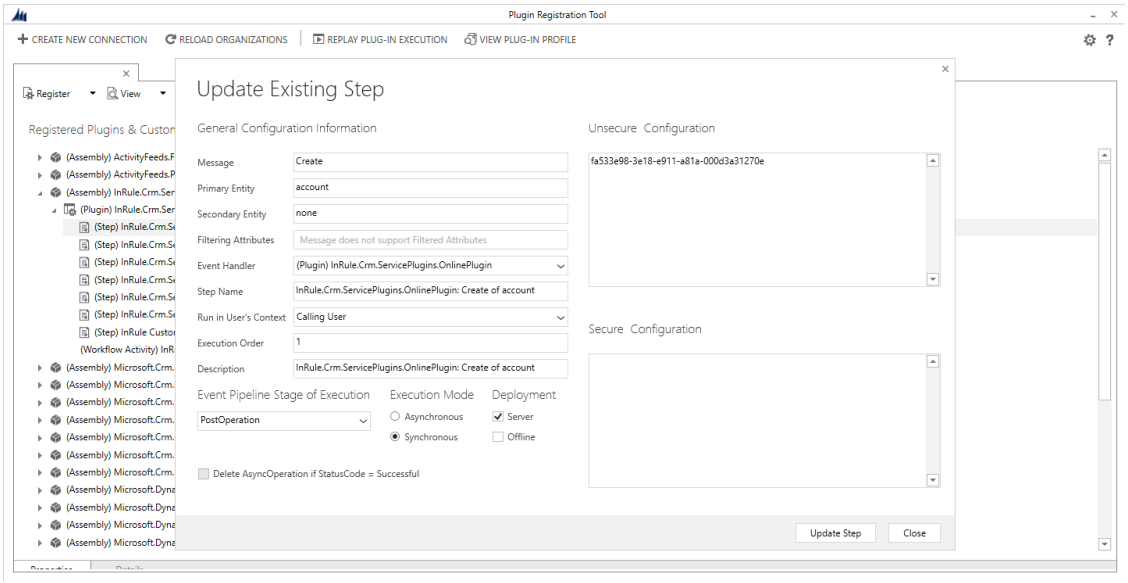
Plugin Registration Tool (Advanced Scenarios Only)

Important: This tool registers all steps to Post Operation stage of execution. If an alternate stage is desired the registration will need to be modified with the Microsoft Plugin Registration

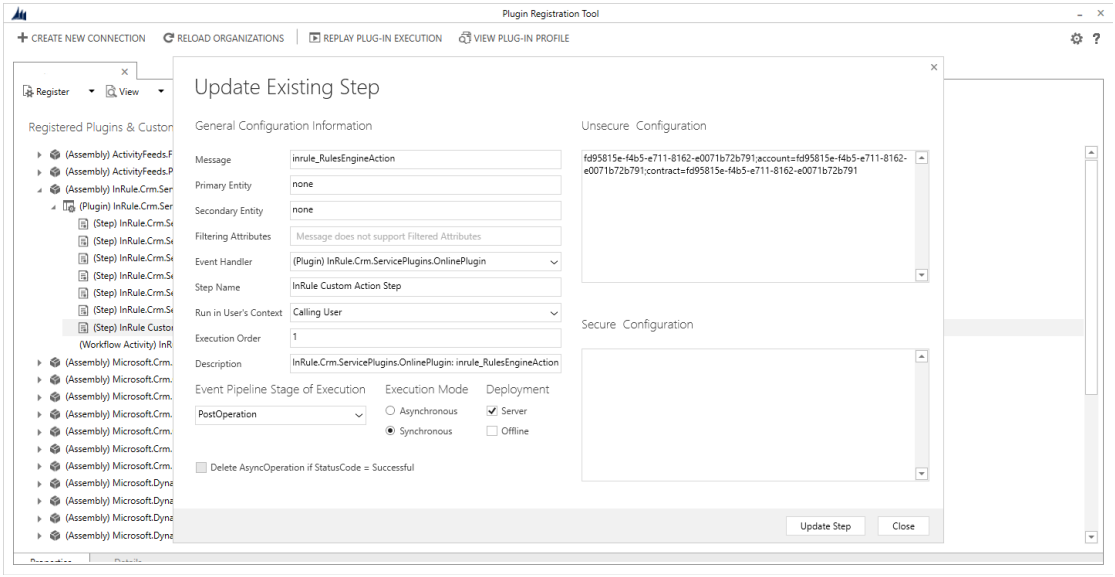
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

tool linked and pictured below. Other custom configuration scenarios may include changing the Execution Mode to 'Asynchronous' or adjusting the Execution Order.

You will also need to use the registration tool if you want to register the plugin to events other than 'Create' or 'Update'. Link to download SDK tools: https://docs.microsoft.com/en-us/dynamics365/customer-engagement/developer/download-tools-nuget



The unsecure configuration field contains the GUID identifier for the associated Rule Configuration. If there are additional inrule_RulesEngineAction steps they will all be stored in this configuration field on the InRule Custom Action Step as entity name, ID pairs.

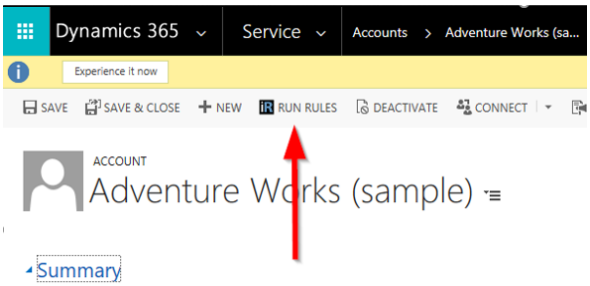


Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

2 Run Rules Button

The 'Run Rules' button makes it easy for the end user to choose when to execute rules. The Rule Services Solution places this button on the ribbon of each Dynamics 365 Entity Form, as shown here:

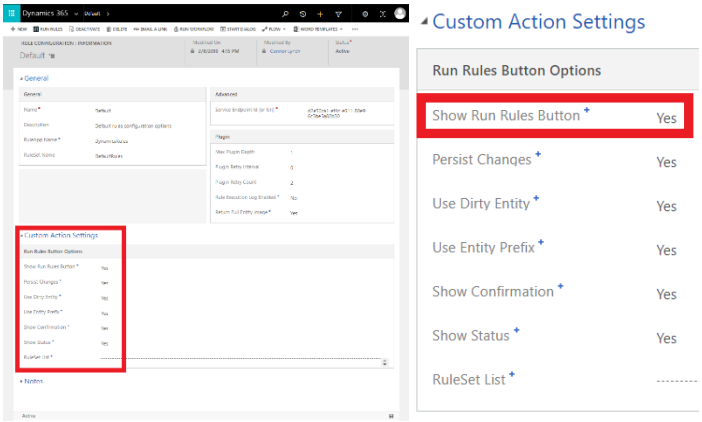


When the Run Rules button is pressed, the included JavaScript will execute rules against the current entity based on the settings defined in the Rule Services Solution – Rule Configuration Form, described in Appendix F: Rules Configuration and Settings. Once you've set up the Integration Framework and validated all functionality, you can hide this button from all forms where it's not used in order to prevent confusion for end users. To do this, follow the steps below to hide the button in the default configuration, and then create new configurations to show the button only on the entities that have associated rules.

Disabling the InRule Run Rules Button

Out-of-the-box, the Run Rules button is used to manually initiate the execution of rules. In some instances, the Run Rules button may not be needed, and the button can be disabled as described below.

1. Create or update a Rule Configuration according to Updating InRule Rules Configuration Records. If you want to hide the button for all forms on all entities, edit the default record. Otherwise, create a new record for the specific entity you'd like to show or hide the button on.
2. Set the **Show Run Rules** Button under the **Custom Action Settings** section



3. If you edited the default record, you can skip this step. Otherwise, you'll need to associate the Configuration record you just modified with a particular entity by following the steps in Associating an InRule Configuration record to an Entity

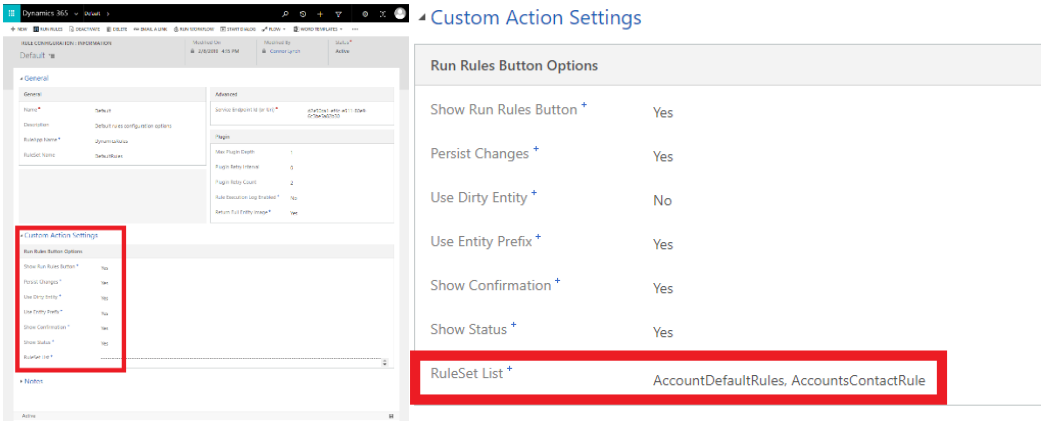
Using the RuleSet List

Out-of-the-box, the Run Rules button is scoped to only run the default rule set that is specified in the configuration file. With the RuleSet List field, you have the option to add a comma separated list of rule

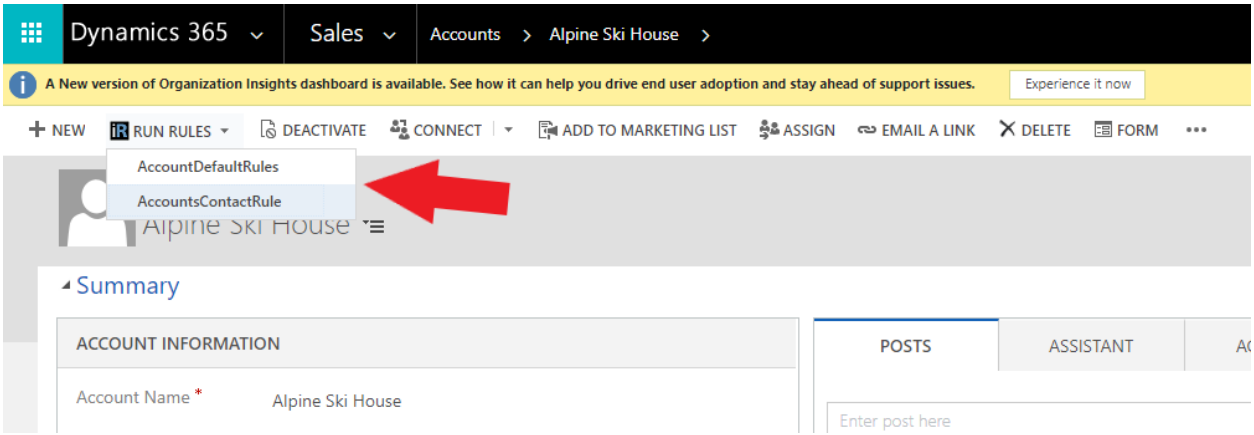
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

sets that may be selected. When this is populated the Run Rules Button will have a dropdown menu added to allow you to select the rule set when running the rule. To enable this functionality, follow the steps below:

1. Create or update a Rule Configuration according to [Updating InRule Rules Configuration Records](). Unless you want this list to appear on all entities, make sure don't modify the Default record.
2. Fill in the ruleset names that you wish to use in this field. Make sure that they are separated with a comma or the menu will not load correctly.



3. Associate the configuration record above with the entity you'd like the list to appear on by following the steps in [Associating an InRule Configuration record to an Entity]()
4. Go to the entity page for the configuration that you have just edited and ensure that the Run Rules button now has the desired ruleset options listed.



3 Workflow Activity

If you need to run rules from a workflow, the solution provides a custom workflow activity that invokes the included custom action. The custom workflow activity passes in the entity ID and type in automatically for you, and provides the rest of the custom action settings as input fields on the activity. The activity also returns the details of the response as individual fields, instead of raw json. These are the fields returned:

- Informational Notifications
- Warning Notifications

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

- Error Notifications
- Validations
- Errors

When using the custom workflow activity, you can select the option to throw an exception on failure. If you choose to throw an exception on failure, the workflow will end immediately if the custom action fails, and the user will see an error message if they are running the workflow synchronously. If you want to implement conditional logic to handle failures, you can disable this option and check for the existence of data in the Error return variable to determine if the activity failed.

If you don't want to use the default configuration for a particular entity, create a new InRule Configuration record according to Updating InRule Rules Configuration Records and then associate it with that entity by following Associating an InRule Configuration record to an Entity.

Important: Before you can use the custom workflow activity, you will need to update the max plugin depth setting for the custom action step to be at least 2, instead of the default value of 1. This is because causing a plugin to be run from a workflow adds 1 to the current plugin depth. For more information on plugin depth, please refer to Changing the Max Plugin Depth

4 Form Events

While the 'Run Rules' button provides an easy way to run rules on demand, you can also run rules automatically on form events, such as 'OnSave'. To help with this, the invokeCustomAction.js web resource provides a helper function called 'executeRulesOnEvent' that you can easily register to an event.

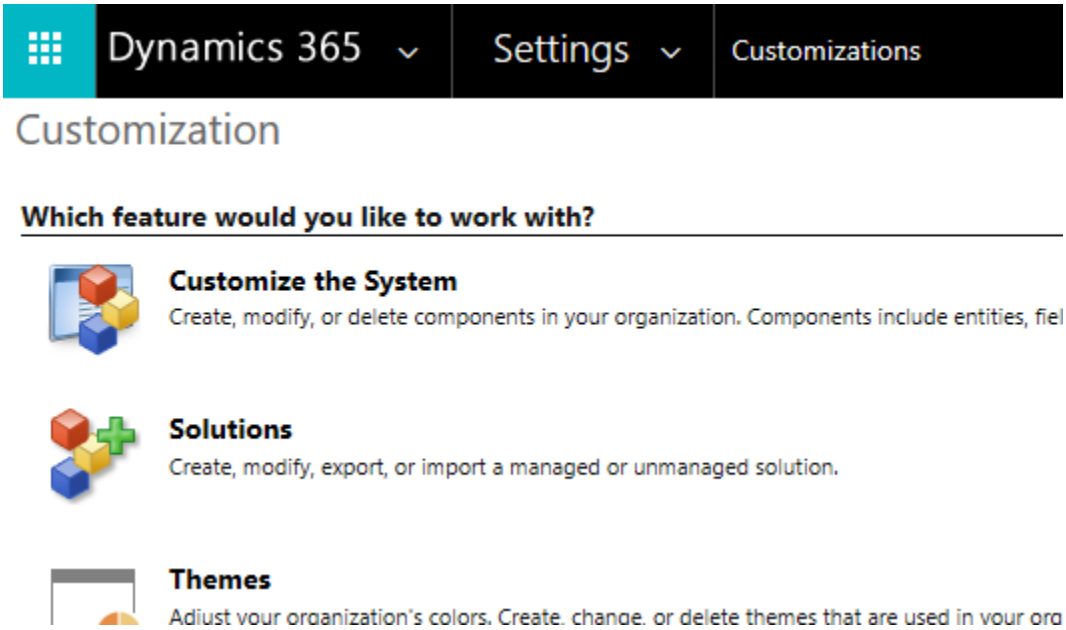
When this function is registered to the 'OnSave' event of a form, it will trigger the rules configured for the entity or event (see specific steps below). Any validation errors returned by the rules will automatically cancel the save operation and display the messages in the notification pane. Alternatively, if you need to use validations to prevent saving for updates made from the Dynamics API, you'll need to configure rules to run on the 'Update' or 'Create' event for an entity as described in the Plugin Events.

I **Important**: If you want to use the dirty field values currently on the form, and not the values already saved to Dynamics when running rules, make sure you set 'Use Dirty Entity Image' to true when setting the InRule Configuration for this entity. You can do this by following the steps in Updating InRule Rules Configuration Records and Associating an InRule Configuration record to an Entity.

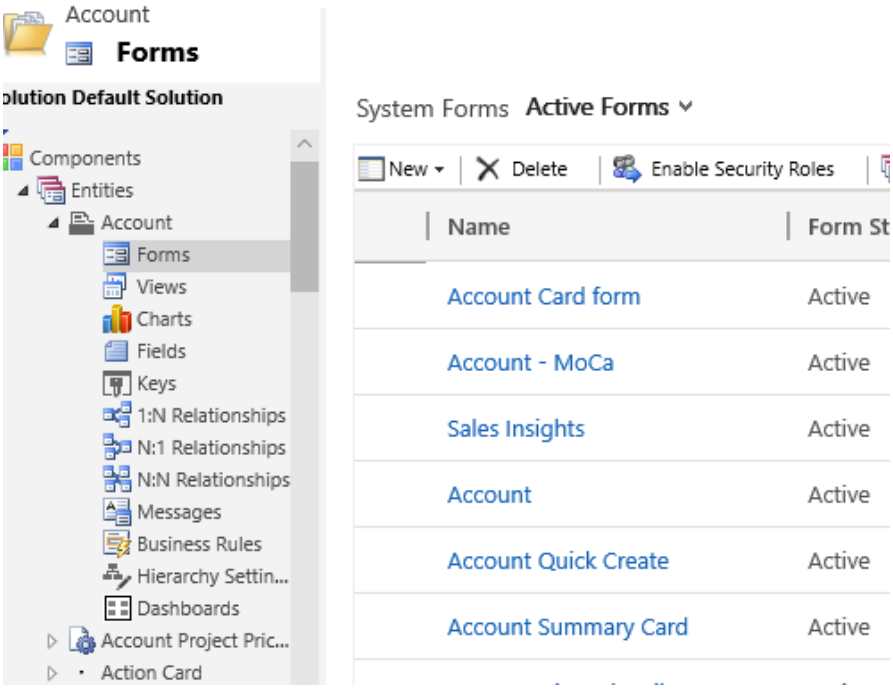
1. Registering a function to a form event requires customizing the entity form. If you want to make this change in a solution navigate to that solution first. Otherwise, navigate to Settings -> Customization -> Customize the System

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

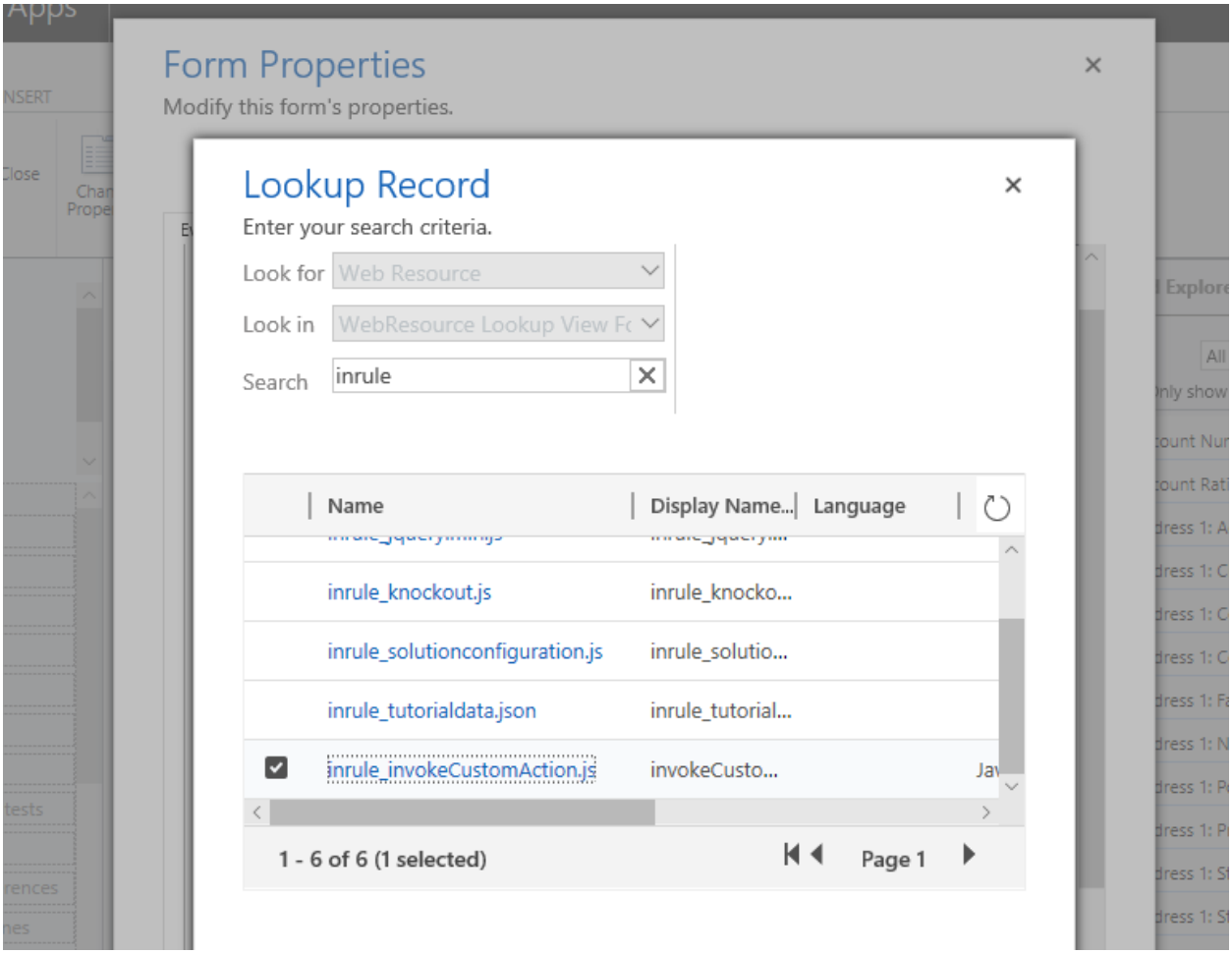


2. Navigate to the entity and form you'd like to customize and select it



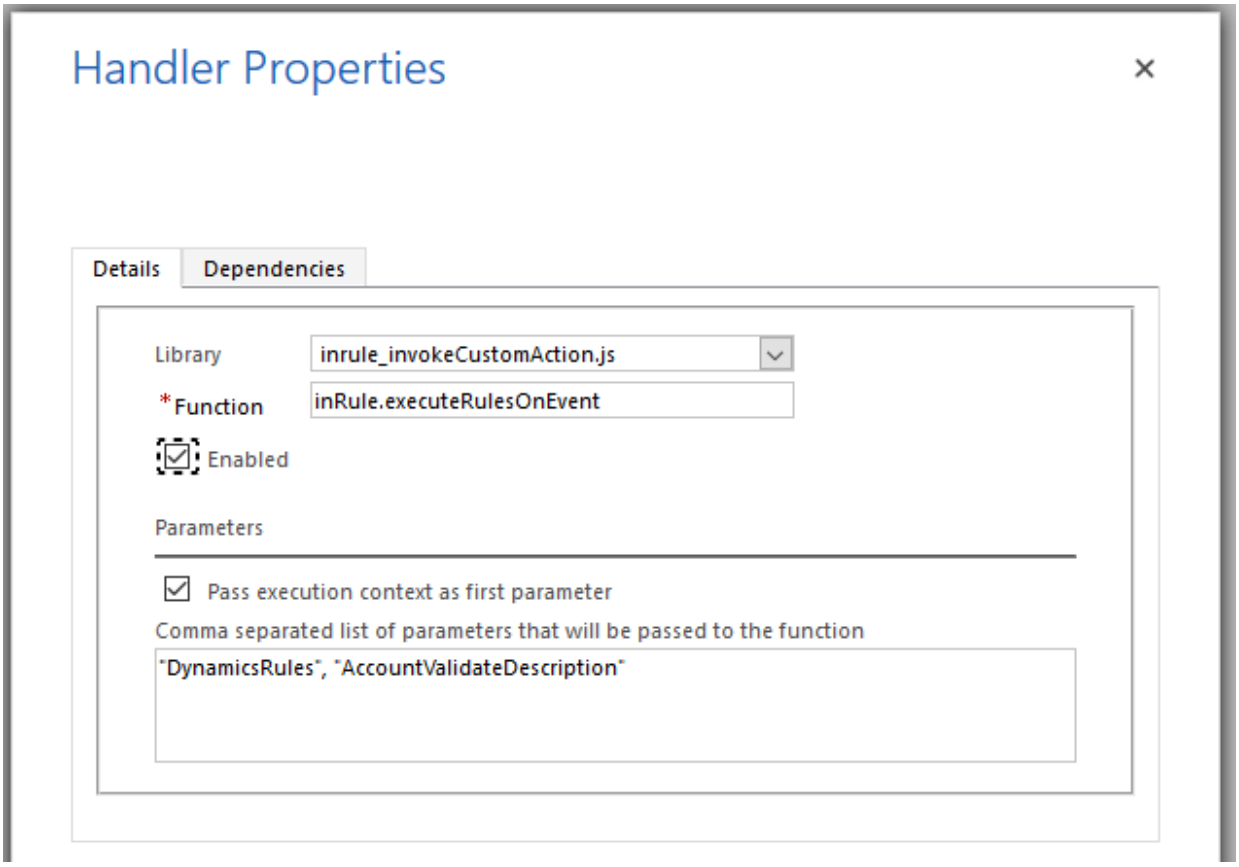
3. Once the form is open, select 'Form Properties' in the ribbon bar, click 'Add' under 'Form Libraries', find inrule_invokeCustomAction.js and add it

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

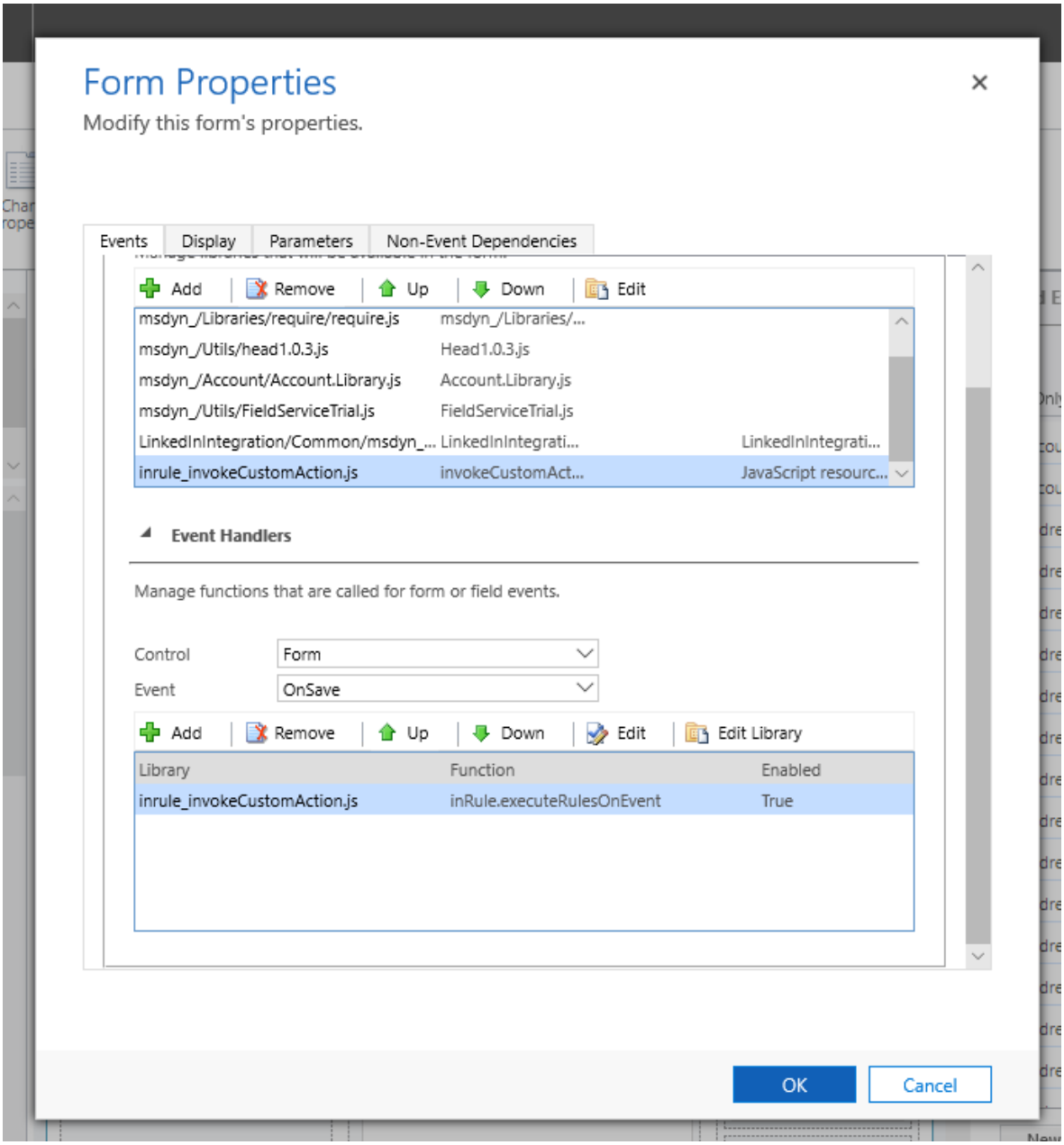


4. Select the desired Control and Event under ‘Form Handlers’ and click ‘Add’. In the window that pops ups, select ‘inrule_invokeCustomAction.js’ for the Library, and enter ‘inRule.executeRulesOnEvent’ for the Function. Under the ‘Parameters’ section, ensure the ‘Pass execution context as first parameter’ is checked. If you want to override the default ruleset and ruleapp configured for the entity, the ruleapp can be passed in as the first parameter, and the ruleset as the second. Both must be passed in order for this to work.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



5. Click 'Ok', 'Ok' and then save and publish the form

5 Custom JavaScript

While InRule provides the ability to easily run rules on demand with the 'Run Rules' button and form events, you can also write your own JavaScript for more advanced scenarios and consume the functions provided in the included invokeCustomAction.js resource directly.

Overriding Rule Configuration behavior with user Options and custom JavaScript

The **Run Rules** ribbon button executes the InRule.executeRules() method without passing in any parameters. The method will use the configuration defined by the Rule Services Solution – Rule

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Configuration Form. However, if an implementer chooses to call the InRule.executeRules() method from within custom JavaScript code, a userOptions object can be passed in that will override the Rule Configuration.

The above JavaScript clip demonstrates how to load up a userOptions object with the values, and then pass those options on to inRule.executeRules(userOptions).

```
var formContext = executionContext.getFormContext();
var userOptions = {
      persistChanges: true,
      useDirtyEntity: false,
      entityTypeName: formContext.data.entity.getEntityName()
      entityId: xpage.data.entity.getId,
      useEntityPrefix: false,
      showConfirmation: true,
      showStatus: true,
      ruleAppName: "MyRuleApplication",
      ruleSetName: "MyRuleSet"
};
inRule.executeRules(userOptions);
```

6 Custom Action

The Rule Services Solution contains a plugin that has a preregistered step called the InRule Custom Action. The InRule Custom Action allows developers to trigger the execution of rules at arbitrary times by calling into Microsoft Dynamics 365's API. Actions provide an easy way to trigger functionality in Dynamics and can be invoked using multiple different methods. Registering an action creates a corresponding web API endpoint. This endpoint can be used by external code, or in the case of the 'Run Rules' button, called from JavaScript. Actions can also be called from workflows directly with the workflow designer, or through a custom workflow activity.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

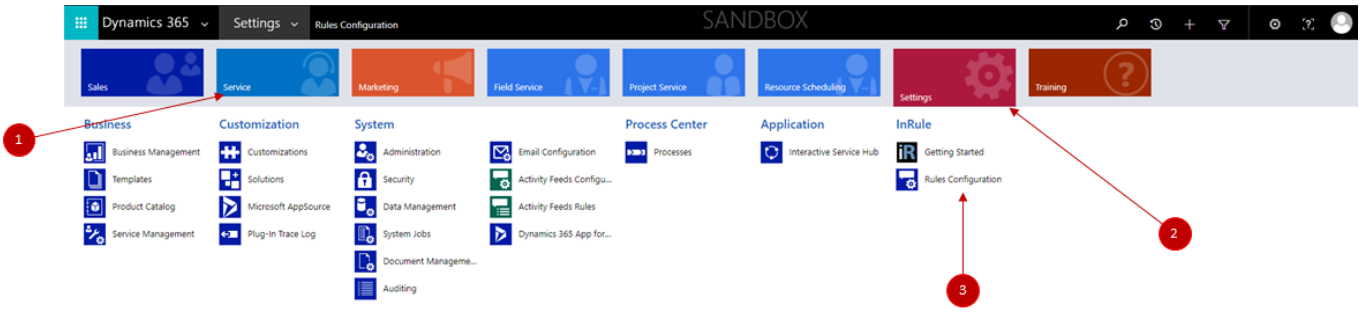
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix F: Rules Configuration and Settings

The *Rule Services Solution – Rule Configurations Form* provides a central location where you can configure the Plug-in provided by the solution. Some of the tasks that you can perform using this form include:

- Change the properties of the default 'Rule Configurations' entry.
- Establish multiple 'Rule Configurations' each which can be associated with different steps.
- Choose the RuleApp Name that will be executed with a given Rule Configuration
- Choose the RuleSet Name that will be executed with a given Rule Configuration
- Choose the Service Endpoint that will be used to contact the *InRule Rule Execution App Service for Dynamics 365*
- Setting the maximum plugin execution depth
- Choose the retry interval and retry counts
- Choose what options will be used by default from the InRule Custom Action JavaScript module to the *Rule Configuration Form*

  Drop down the chevron next to the first menu item after Dynamics 365, Select Settings, Select Rules Configurations under the InRule heading.

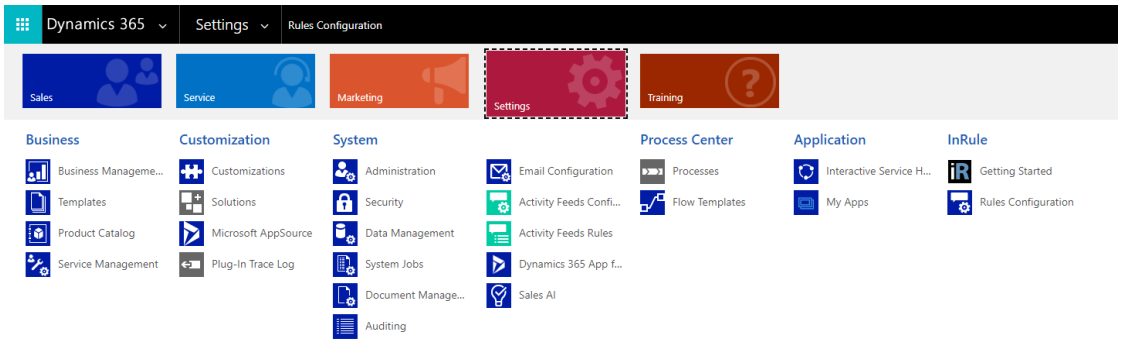


Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

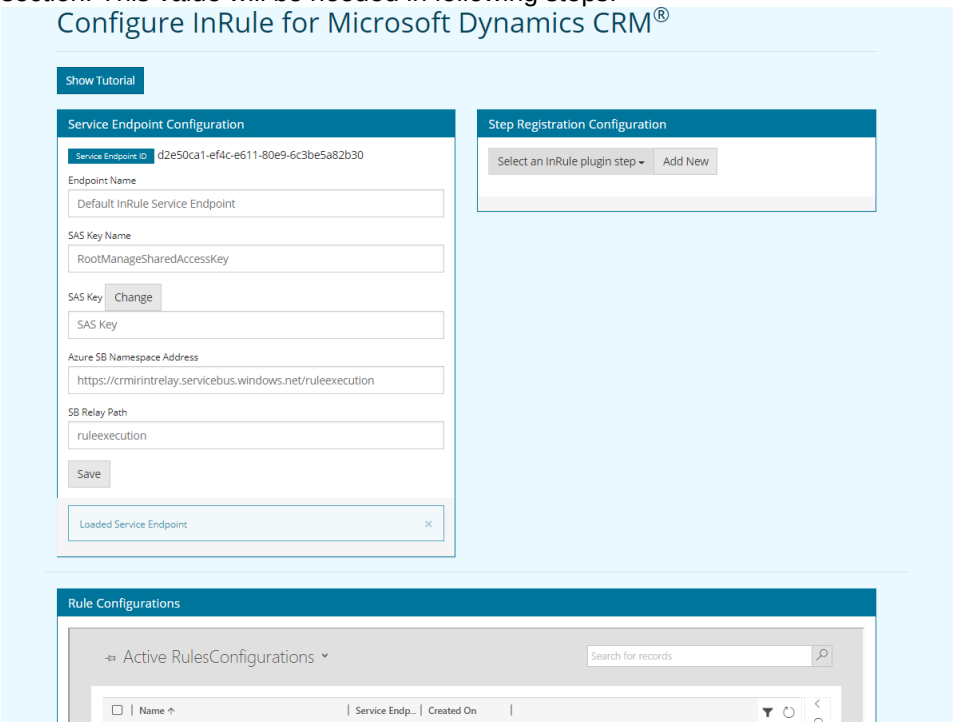
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Updating InRule Rules Configuration Records

1. Login to Dynamics 365 and go to the “Settings” tab and select Rules Configuration under the InRule section



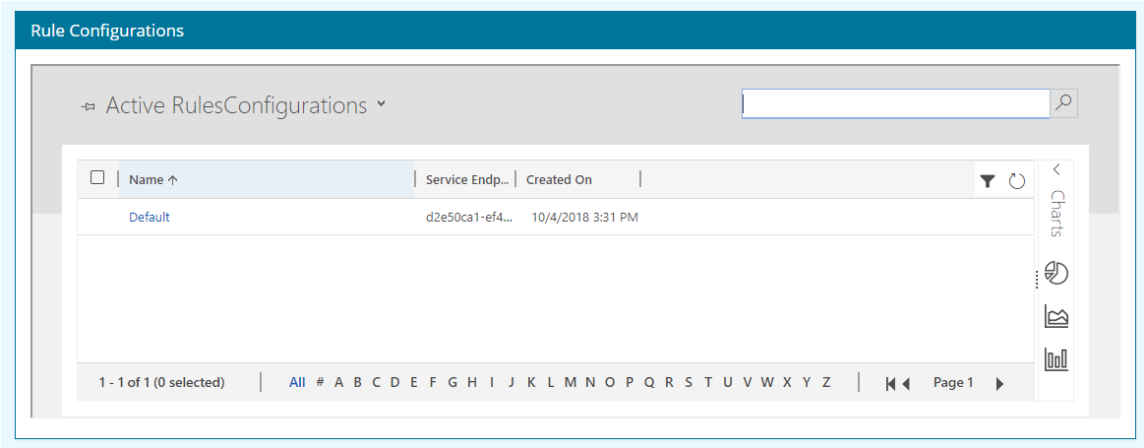
2. Note the Service Endpoint ID that appears at the top of the “Service Endpoint Configuration” section. This value will be needed in following steps.



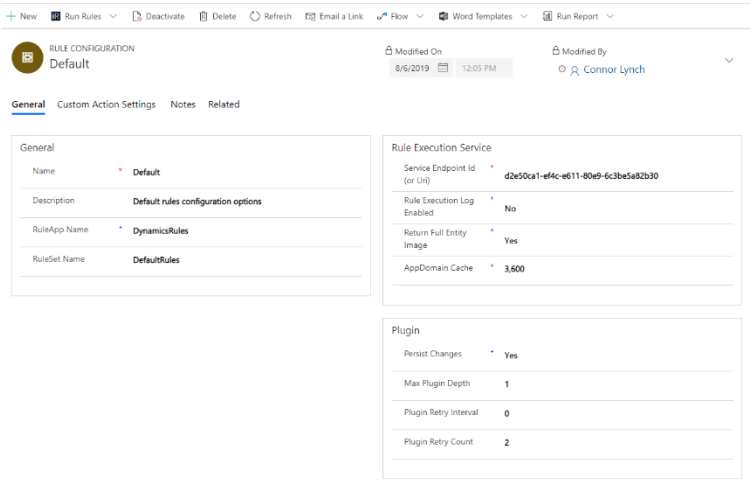
3. Scroll down to Active RulesConfigurations

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

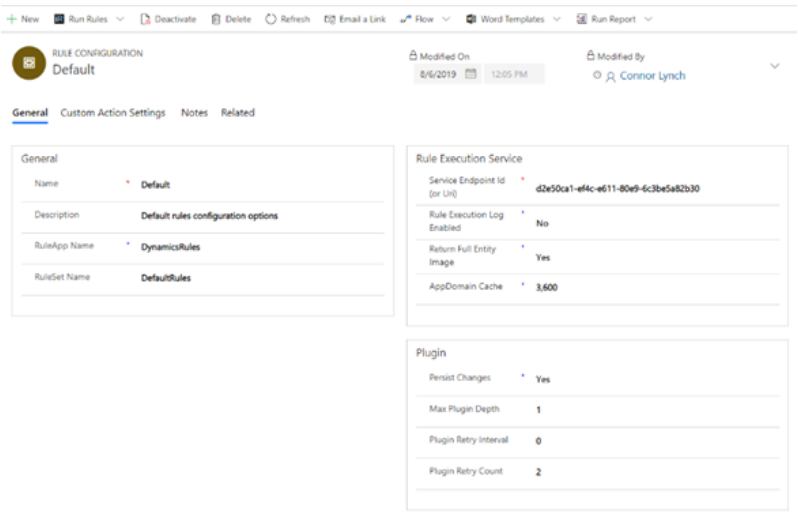
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



4. Right click on “Default” and select “Open in New Window.” The following should open up in a new tab:

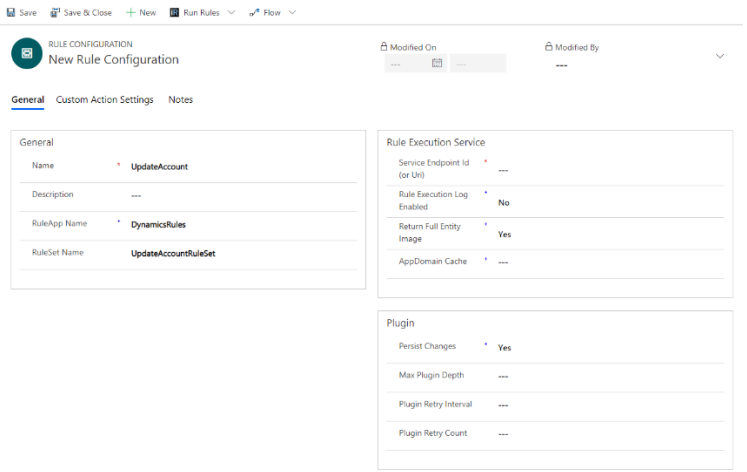


5. From here you can either edit the existing record or select “New” in the top right



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

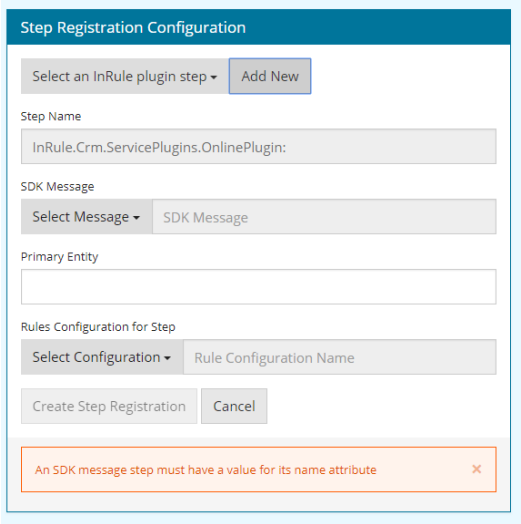
6. Configure the Rule Configuration as follows:
    a. Choose a name
    b. Create a description
    c. Enter the RuleApp containing the RuleSet you want
    d. Enter the RuleSet you want executed when the step is invoked
    e. Set “Service Endpoint Id” on the right side equal to the ID noted in Step 2
    f. Press “Save” in the top-left corner
    g. Repeat this process for every step you want to register



Associating an InRule Configuration record to an Entity

If you want to use different InRule Configurations for different Dynamics entities, you can create multiple configuration records and then associate them to a particular entity. The same configuration record can be associated to multiple different entities if desired.

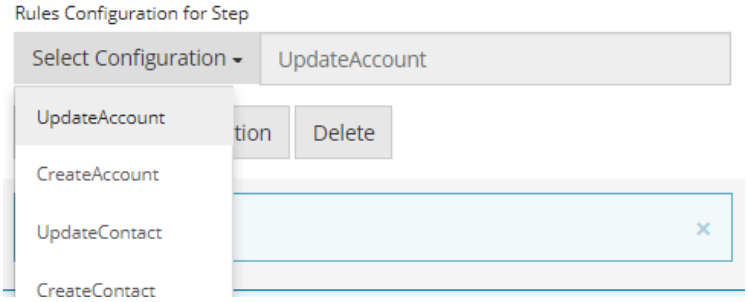
1. Create or update the configuration record according to the steps above
2. Return to the Rules Configuration page and navigate to the Step Registration Configuration section and click **Add New**



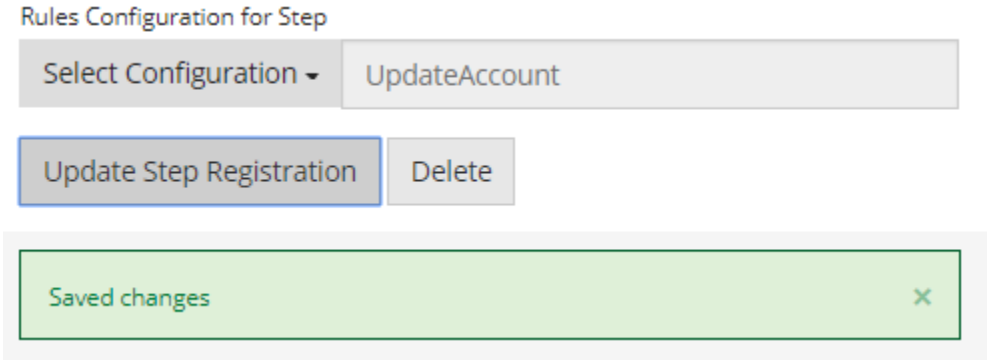
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3. Choose the SDK Message (Update, Create, inrule_RulesEngineAction)
    a. Update – Update of the primary entity
    b. Create – Create of the primary entity
    c. inrule_RulesEngineAction – The custom action message invoked when the 'Run Rules' button is clicked. Choose this to override the default behavior for a particular entity whenever the 'Run Rules' button is clicked, custom JavaScript is executed using the included invokeCustomAction.js resource, the workflow activity is used, a form event is used, or the custom action is invoked through some other means.

4. Select a Primary Entity for your rule registration to use
5. Open the Rule Configurations for Step dropdown menu that appears and select the Rule Configuration that you made above.



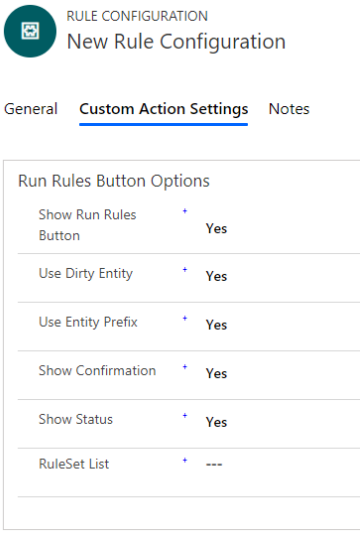
6. Click "Update Step Registration" and verify that the registration successfully saves



An Explanation of the InRule Custom Action Options

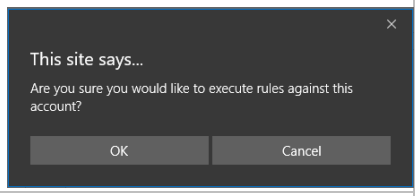
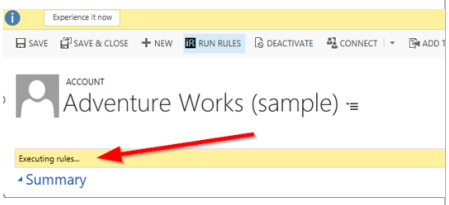
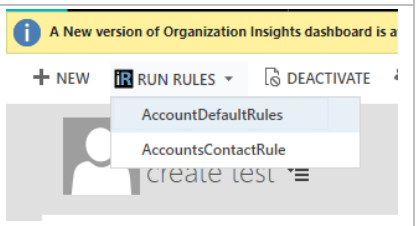
The below options must be passed into the InRule Custom Action. InvokeCustomAction.js exposes InRule.executeRules() which will use values defined in the Rule Services Solution – Configuration Form. Alternatively, a consumer of InRule.executeRules() can pass in an object that provides specific values that should be used on a call by call basis

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Show Run Rules Button	If true, then the *Inrule Run Rules Button* will be visible on all pages. If false, then the *InRule Run Rules Button* will be globaly removed. This will only effect the the use of the button, but not other registered plugin steps.
UseDirtyEntity	If true, the InRule Custom Action will expect that form data will be included in the API call. The form data is used to populate the root entity in the Rule Engine. This is useful in scenarios when data has changed on the Dynamics form but the save button has not yet been pressed. This allows a user to run rules against data before a save takes place. If false, the only data passed to the Custom Action is the Entity ID (guid) of the entity, and the Custom Action will query Dynamics directly for that entities data. This limits the InRule Custom Action to being aware only of data that has been fully saved to Dynamics.
UseEntityPrefix	If true, then the supplied *RuleSetName* is interpreted as a suffix to the *EntityTypeName*. For example: if the supplied *RuleSetName* is "DefaultRules" and the *EntityTypeName* you are dealing with is "Account", then the *RuleSetName* actually used will be "AccountDefaultRules". If false, then the supplied *RuleSetName* is interpreted literally.
ShowConfirmation	If true, this will ask the user in the User Interface with the following visual prompt before actually executing rules: 
ShowStatus	If true, a status messages will be displayed within the Dynamics interface when rules are executing, when rules are finished, and if rules are cancelled by the confirmation dialog. 
RuleSet List	This is a free form text box that allows for you to specify a list of rule set names that can be run with the particular rule set configuration. When this is populated, the default Rule Set Name will be ignored and these options will be populated into a dropdown on the run rules button. 

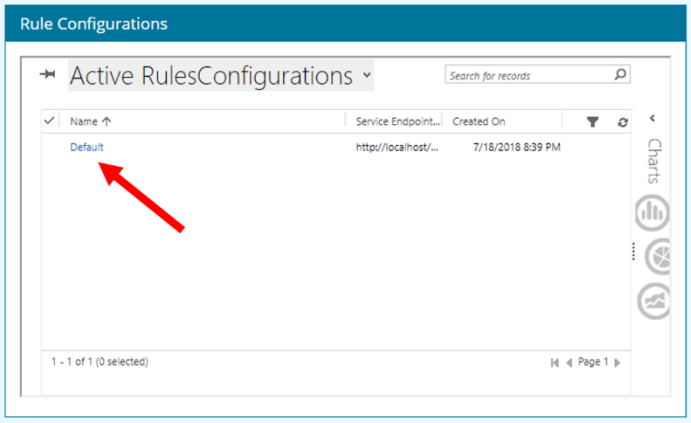
Displaying the Entity Image in Trace Logs

When troubleshooting running rules, it can sometimes be helpful to see the entire image returned from the execution service. By default, this image is printed out in the trace log as json. However, printing this out can cause other parts of the trace log to be truncated, so this can be conditionally disabled from the Rule Configuration Form. To disable the entity image, follow the steps below:

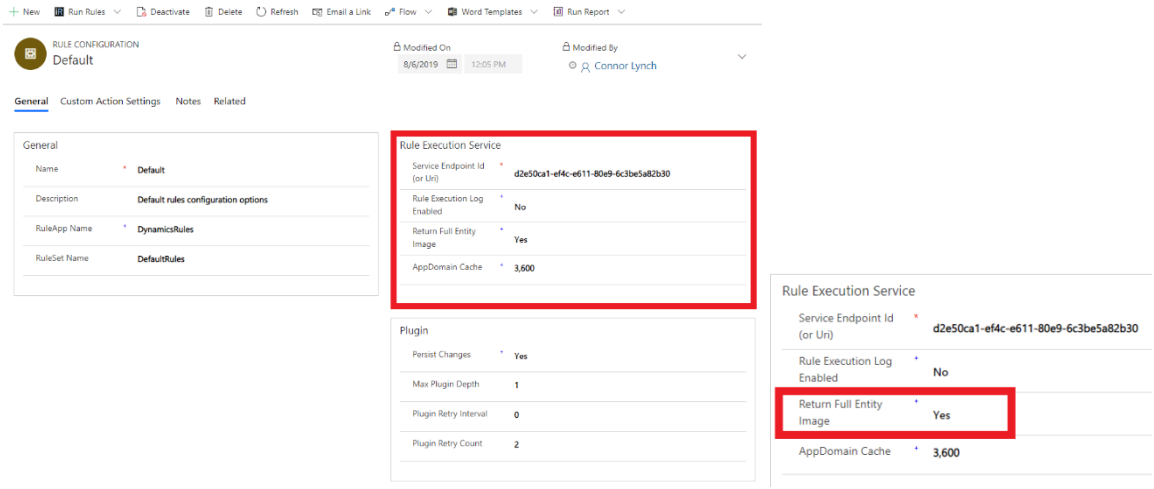
1. Navigate to the Rule Services Solution Rule Configuration Form
2. Scroll down and select the 'Default' record under the 'Rule Configurations' section. Please note that the change made in this specific record will take effect across the whole system.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

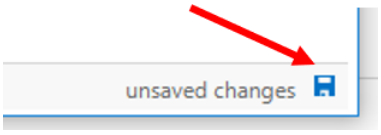
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



5. Set the **Return Full Entity Image** Button under the **Rule Execution Service** section



6. Save the Entity in the bottom right hand corner



Changing the Max Plugin Depth

Dynamics provides the plugin depth property to the plugin context to track the current depth of the call stack. Whenever another plugin or process is executed within the scope of the current transaction, this value is incremented by 1 in the invoked plugin. This value is commonly used to prevent infinite loops, where multiple plugins keep firing each other. The Max Plugin Depth setting in the InRule Configuration will stop execution of the plugin whenever the depth is greater than the specified value. By default, this value is set to 1, but you may need to change it in certain cases, such as using the included custom workflow activity, or running rules from within your own custom plugin. To update this value, change the 'Max Plugin Depth' setting using the steps in Updating InRule Rules Configuration Records, and optionally associate the record with a specific event and entity by following the steps in Associating an InRule Configuration record to an Entity

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

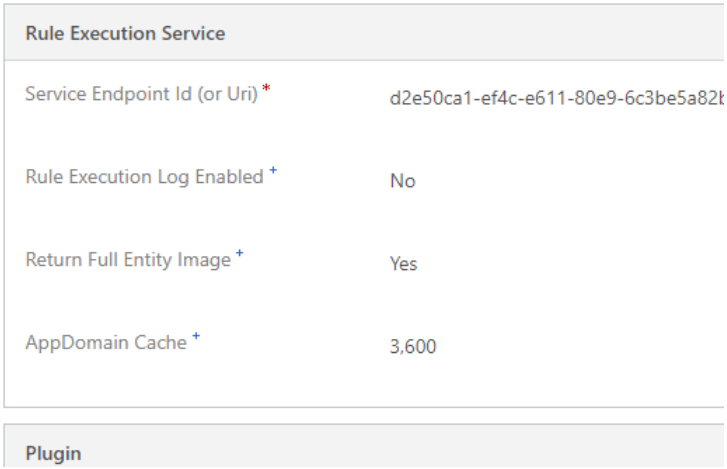
Disabling Persist Changes

When Persist Changes is set to enabled all changes made to all entities by InRule are persisted back to Dynamics. When disabled any changes made by InRule are sent back in the response and can be consumed by the caller, but they are not persisted back to Dynamics.

The Persist Changes setting can be found in the Plugin section of the Rule Configuration.

Configuring the AppDomain Cache

When using the Rule Helper, the integration framework provides the ability to save query results in a persistent cache in the execution service AppDomain. This is useful if you have relatively static data that is used consistently in one or more rule apps, since this cache persists across rule executions. By default, this cache is disabled, but you can enable it by setting a value for the cache timeout in the Rule Configuration. To update this value, change the 'AppDomainCache' setting to the desired retention time in seconds using the steps in Updating InRule Rules Configuration Records, and optionally associate the record with a specific event and entity by following the steps in Associating an InRule Configuration record to an Entity. If this value is set to zero, the next time the rule helper is used from the execution service the cache will be cleared. The cache will also be cleared if the execution service is ever restarted. For more information on how to use the rule helper, refer to Appendix D: Accessing Dynamics 365 Directly from Rule Helper



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix G: Azure App Service Plan Configuration

Azure App Service Plan Overview

The Dynamics Rule Execution Azure App Service service runs on an Azure App Service Plan. The ARM Template deployment process outlined in Section 3.3.3: Rule Execution App Service for Dynamics 365 will, by default, automatically deploy an App Service Plan for you.

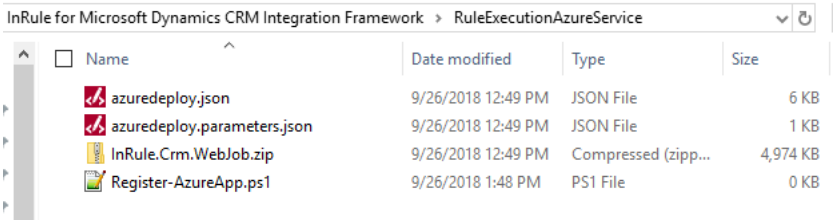
This App Service Plan will be deployed to the subscription and resource group provided during the deployment process. Additionally, the plan will be configured with a "B1" (Basic, Small) pricing tier. This is the ***lowest*** tier possible for the Rule Execution App Service to run in, as it is the lowest tier that allows running in **Always On** mode. **Always On** is required to permit a continuous web job to run on the app service. Thus, regardless of whether or not you opt to allow the ARM template to create an App Service Plan for you, or use a pre-existing one, the plan must, at a minimum, be of the "B1" tier or higher.

Should you wish to use a pre-existing Azure App Service Plan rather than have a new one created for you, a few configuration steps within the ARM template itself are necessary.

Configuring the ARM Template to Use an Existing Azure App Service

1: Locate azuredeploy.parameters.json

The ARM template parameters file is located in the *RuleExecutionAzureService* folder as, defined in Section 3.3.3: Rule Execution App Service for Dynamics 365



2: Populate "appServicePlanName" parameter

Open the file in your text editor of choice. First, populate the **"appServicePlanName"** parameter at the bottom of the parameters file. Set the value equal to **the name of your app service plan**.

```
    },
    "appServicePlanName": {
      "value": "yourAppServicePlanName"
    }
  }
}
```

3: Create "createAppServicePlan" parameter

Next, just below the "appServicePlanName" parameter you will add an additional parameter called "**createAppServicePlan**" Detailed below (Note, be sure to add the pictured comma immediately following the first bracket after the "appServicePlanName" parameter):

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

```
    "appServicePlanName": {
      "value": ""
    },
    "createAppServicePlan": {
      "value": false
    }
  }
}
```

4: Create "servicePlanResourceGroupName" parameter

Lastly, we need to add a parameter to inform the ARM template what resource group your App Service Plan is in. Create the "servicePlanResourceGroupName" parameter as shown below and define the value as **the name of the resource group your App Service Plan exists in.**

```
    "createAppServicePlan": {
      "value": false
    },
    "servicePlanResourceGroupName": {
      "value": "yourResourceGroupName"
    }
  }
}
```

5: Save azuredeploy.parameters.json and continue deployment

Save and close the file. You can now proceed with the deployment process outlined in Section 3.3.3: Rule Execution App Service for Dynamics 365 as normal; your rule execution app service will now deploy to the App Service Plan you defined in the steps above

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

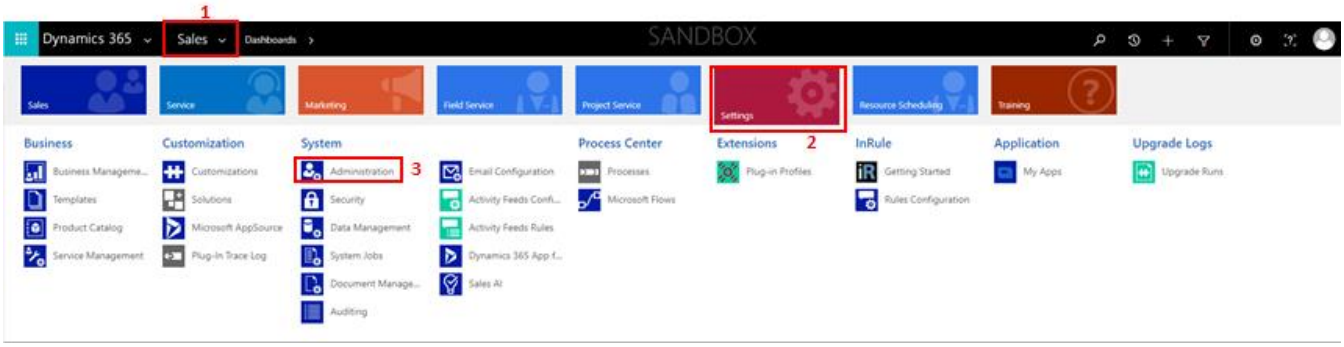
Appendix H: Dynamics 365 Tracing and InRule Event Logging

Dynamics 365 Plug-In Trace Logging

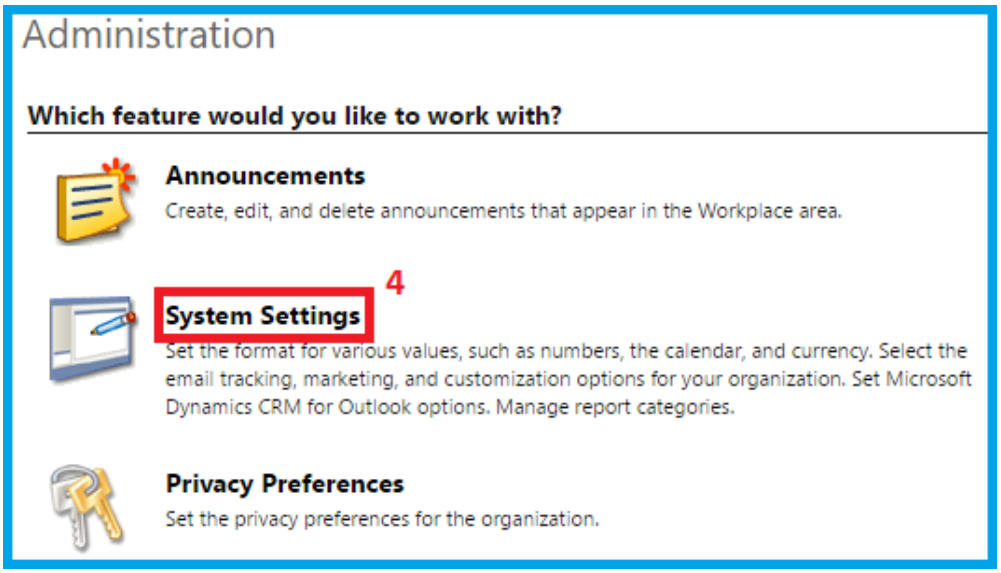
The Dynamics Plug-In Trace Log feature is a debugging tool made available within Dynamics itself for reviewing plugin events and exceptions. This section will highlight how to enable this feature within Dynamics and how to utilize it. Currently, Plug-In Trace Logs only work with the Execution Service in online deployments. For more detail on the limitations of on-prem deployments, reference Appendix J: Known Issues, Limitations and Troubleshooting.

Enable Plug-In Trace Log

To enable the Plug-in Trace Log, first navigate to **Settings** > **Administration**.

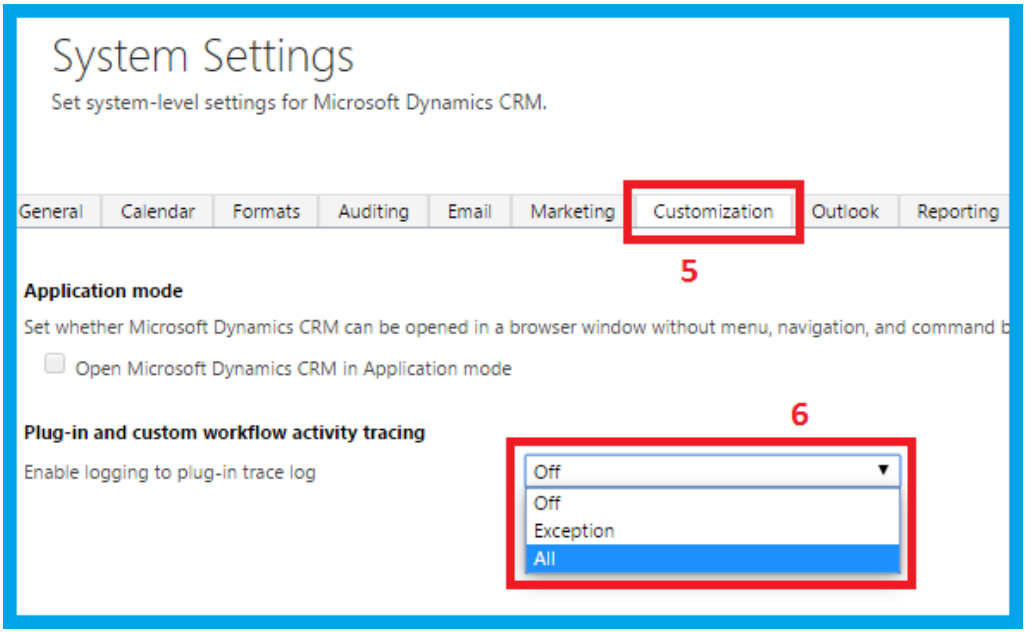


Once you're in the Administration section, click on **System Settings.**



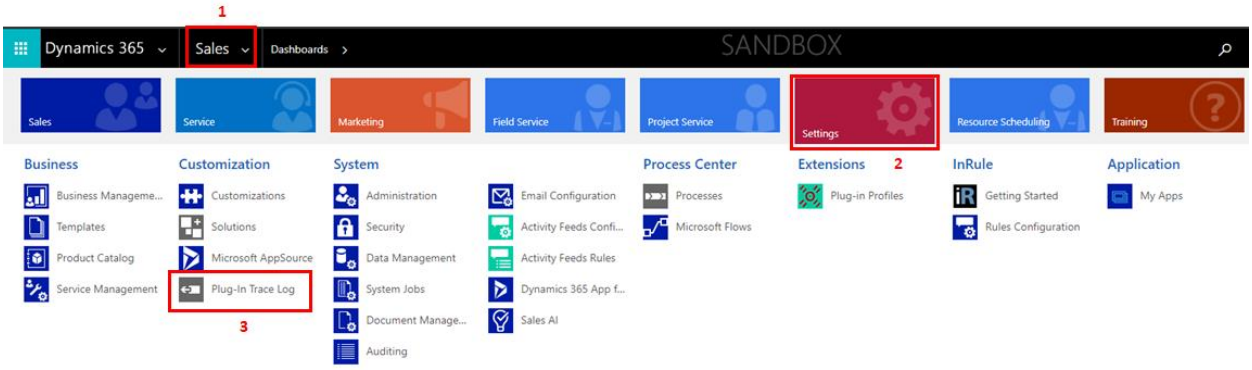
From here you can navigate to the **Customization** tab and select **All** for the **Enable logging to plug-in trace log** field. Then click **OK**. This will create trace logs for all plugin events within your Dynamics environment. In the context of InRule, this includes Create, Update, and Custom Action events.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Viewing Plug-In Trace Log

To review Plug-In Trace Logs after they have been enabled, navigate to **Settings** > **Plug-in Trace Log**.



Here, there will be a list populated with all logged plugin events. If you do not see a list of logs in a similar fashion as below, that means no plugin events have been logged.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

More Actions

System Creat...	Operation Ty...	Type Name	Message Name	Execution Start Time ↓
Yes	Plug-in	InRule.Crm.ServicePlugins.Onli...	inrule_RulesEngineAction	1/21/2019 12:53 PM
Yes	Plug-in	InRule.Crm.ServicePlugins.Onli...	inrule_RulesEngineAction	1/21/2019 12:52 PM
Yes	Plug-in	InRule.Crm.ServicePlugins.Onli...	inrule_RulesEngineAction	1/21/2019 12:52 PM
Yes	Plug-in	InRule.Crm.ServicePlugins.Onli...	inrule_RulesEngineAction	1/21/2019 12:52 PM
Yes	Plug-in	InRule.Crm.ServicePlugins.Onli...	inrule_RulesEngineAction	1/21/2019 12:11 PM
Yes	Plug-in	InRule.Crm.ServicePlugins.Onli...	inrule_RulesEngineAction	1/21/2019 12:10 PM
Yes	Plug-in	InRule.Crm.ServicePlugins.Onli...	inrule_RulesEngineAction	1/21/2019 12:10 PM
Yes	Plug-in	InRule.Crm.ServicePlugins.Onli...	inrule_RulesEngineAction	1/21/2019 12:10 PM

For this example, we will view the resulting log of executing a Custom Action from manually running rules. To view a specific log, simply click on the hyperlinked **Type Name** text.

The first section within a specific trace log is the Configuration section, which details the various configuration details about the event, including the Rule Configuration ID used, the event type, the plugin step ID, and more. This section can be useful for debugging by providing a quick means of determining whether or not the rules executed were properly configured.

Configuration

General

System Created *	Yes		
Type Name	InRule.Crm.ServicePlugins.OnlinePlugin		
Message Name	inrule_RulesEngineAction	Primary Entity	none
Configuration	fd95815e-f4b5-e711-8162-e0071b72b791	Secure Configuration	--
Persistence Key	00000000-0000-0000-0000-000000000000	Operation Type	Plug-in
Plugin Step Id	070ad2c4-ae70-e711-811a-e0071b6a6141		

Context

Depth	1	Mode	Synchronous
Correlation Id	d3b2ce70-4af3-4fd7-a8fb-0a38792ef397	Request Id	4eae987f-1be5-467f-b5f5-cff3942fbeae

The second section, the Execution section, is typically the most useful for debugging. It will provide a block of all messages logged by the plugin during its execution, as well as any exception details, if the event resulted in an exception.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Execution Start Time	1/21/2019 1:51 PM	Execution Duration (ms)	26,34

Message Block

```
Original Input parameters:
Parameter name: entityId, Value: {E1F7327C-9D1A-E911-A82C-000D3A31299D}
Parameter name: entityTypeName, Value: contact
```

Exception Details

```
Unhandled exception:
Exception type: System.ServiceModel.FaultException`1[Microsoft.Xrm.Sdk.OrganizationServiceFault]
Message: An unexpected error occurred from ISV code. (ErrorType = ClientError) Unexpected exception from plug-in (Execute): InRu
set to an instance of an object.Detail:
<OrganizationServiceFault xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/xrm/201
<ActivityId>2a4757b4-fe60-44a6-97d4-45543c322b93</ActivityId>
<ErrorCode>-2147220956</ErrorCode>
<ErrorDetails xmlns:d2p1="http://schemas.datacontract.org/2004/07/System.Collections.Generic" />
<Message>An unexpected error occurred from ISV code. (ErrorType = ClientError) Unexpected exception from plug-in (Execute): Inl
not set to an instance of an object.</Message>
<Timestamp>2019-01-21T19:51:39.3705065Z</Timestamp>
<ExceptionRetriable>false</ExceptionRetriable>
<ExceptionSource>PluginExecution</ExceptionSource>
<InnerFault i:nil="true" />
<OriginalException>System.NullReferenceException
at InRule.Crm.ServicePlugins.OnlinePlugin.Execute(IServiceProvider serviceProvider)
at Microsoft.Crm.Sandbox.SandboxAppDomainHelper.&lt;&gt;c__DisplayClass14_3.&lt;Execute&gt;b__0()
--- End of stack trace from previous location where exception was thrown ---
at System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()
at Microsoft.Crm.Sandbox.SandboxAppDomainHelper.Execute(IOrganizationServiceFactory organizationServiceFactory, Dictionary`2
pluginSecureConfig, IPluginExecutionContext requestContext, Boolean enablePluginStackTrace, Boolean chaosFailAppDomain)
at Microsoft.Crm.Sandbox.SandboxAppDomainHelper.Execute(IOrganizationServiceFactory organizationServiceFactory, Dictionary`2
pluginSecureConfig, IPluginExecutionContext requestContext, Boolean enablePluginStackTrace, Boolean chaosFailAppDomain)
at Microsoft.Crm.Sandbox.SandboxWorker.Execute(SandboxCallInfo callInfo, SandboxPluginExecutionContext requestContext, Guid
pluginTypeName, String pluginConfiguration, String pluginSecureConfig, SandboxRequestCounter&amp; workerCounter, Boolean re
<TraceText>
Entered InRule.Crm.ServicePlugins.OnlinePlugin.Execute()
Plugin context retrieved. Message is inrule_RulesEngineAction
Loading configuration entity with ID 133ca426-9d1a-e911-a82c-000d3a31299d
```

The message block logs rule execution “milestones.” In the event that the plugin fails to execute properly, the message block is useful for determining at what point it is failing. The Exception Details block will provide any related details to any exception thrown and is generally the first place to look to diagnose the nature of a plugin execution failure.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

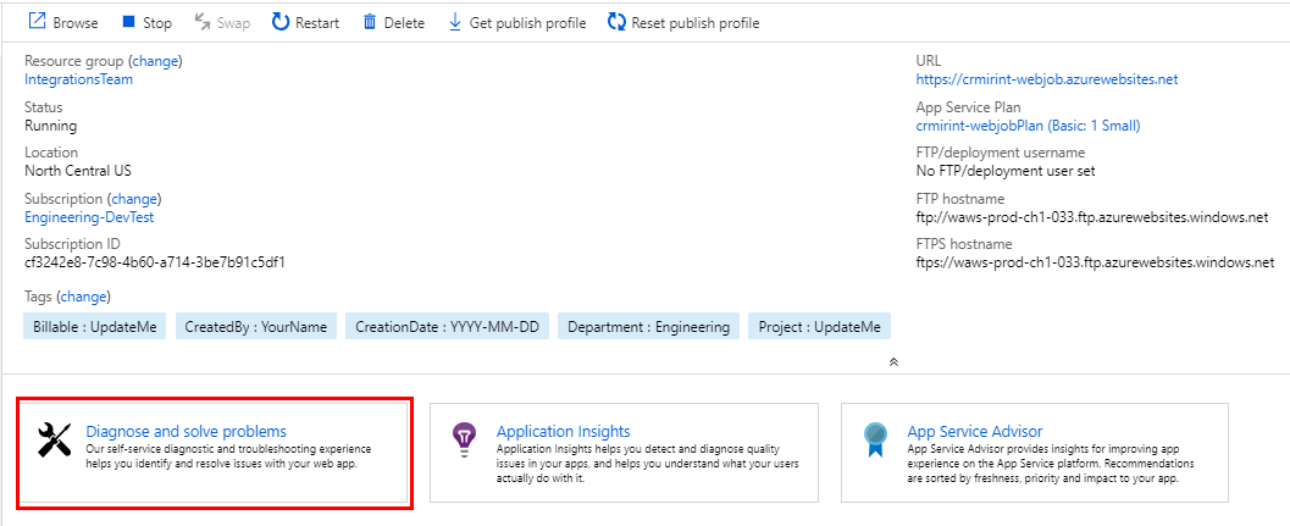
Rule Execution Service Event Log

Event logging can be enabled in the Rule Execution App Service to monitor application events. These logs can be tremendously useful for debugging any issues encountered with the Rule Execution Service.

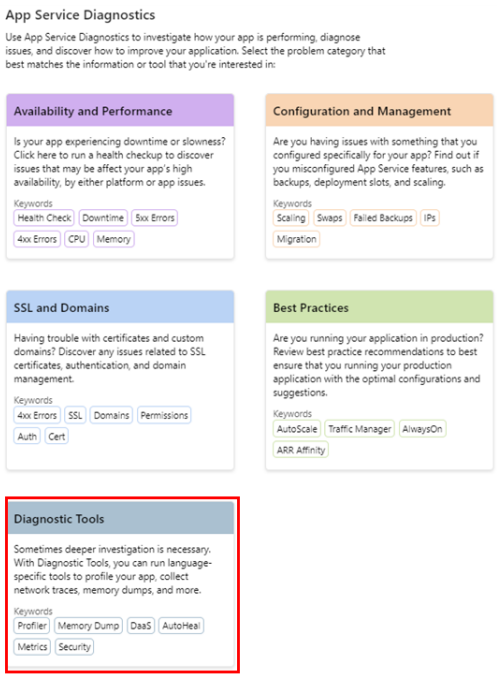
Viewing Application Event Logs

To enable event logging, login to Azure and navigate to your App Service as created as a part of the Azure deployment process detailed in Section 3.3.3: Rule Execution App Service for Dynamics.

Once you're looking at the overview of your app service, select **Diagnose and solve problems**



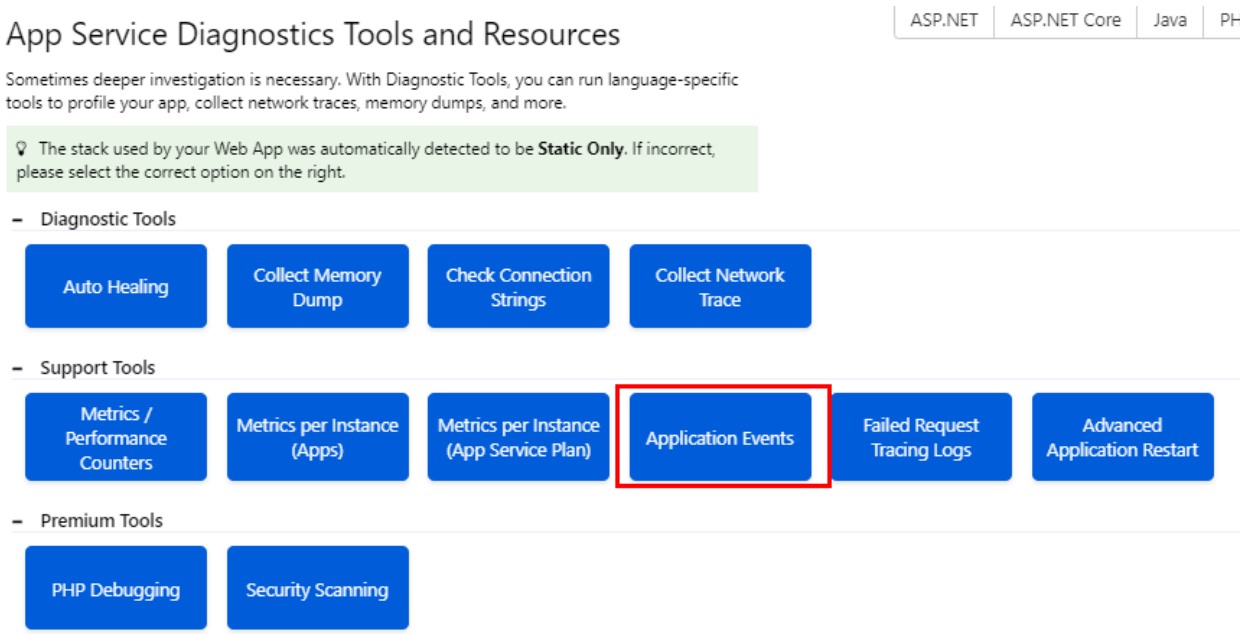
Select **Diagnostic Tools**



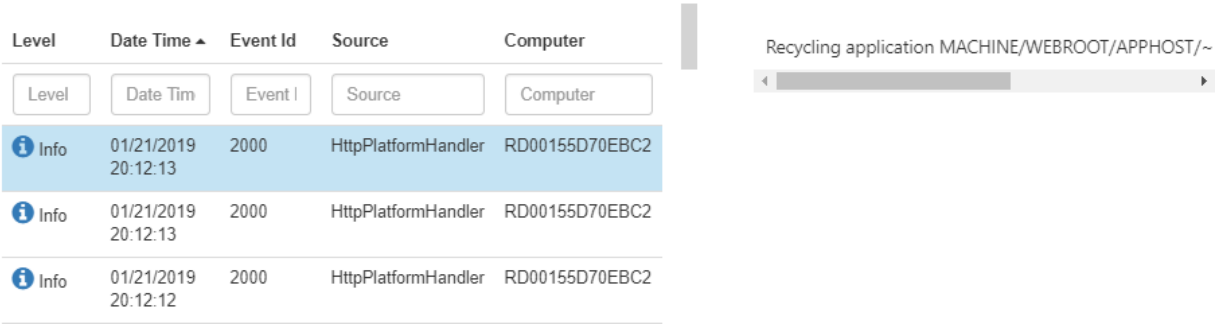
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Select **Application Events**



You should see a list of all application events logged by the rule execution service, denoted with the notification level, timestamp, event ID, source and web server. Selected an event log will cause the log details to appear in a separate column on the right-hand side of screen.



In the event of rule service issues, error events in this log stream can be useful for debugging purposes. For example, below is an example of an error event log in an instance where the rule service contacting the catalog looking for a rule application that didn't exist:

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

```
Time: 10/30/2018 7:19:54.872 PM, Source: WcfCatalogProxy, Message:
Rule application 'DynamicsRules2' does not exist in the catalog.
Installer Version: 5.2.0.166
irSDK Version: 5.2.0.166
HostAppDomainHeapMemoryMB: 18
, Detail:

Operating System: Microsoft Windows NT 10.0.14393.0
Processor Count: 1
CLR Version: 4.0.30319.42000
CLR Mode: 32-bit
Thread Culture: 1033
ThreadID: 15
ProcessID: 0
Installer Version: 5.2.0.166
irSDK Version: 5.2.0.166

Error Information:
InRule.Repository.Service.InRuleCatalogException: Rule application 'DynamicsRules2' does not exist in
the catalog.
```

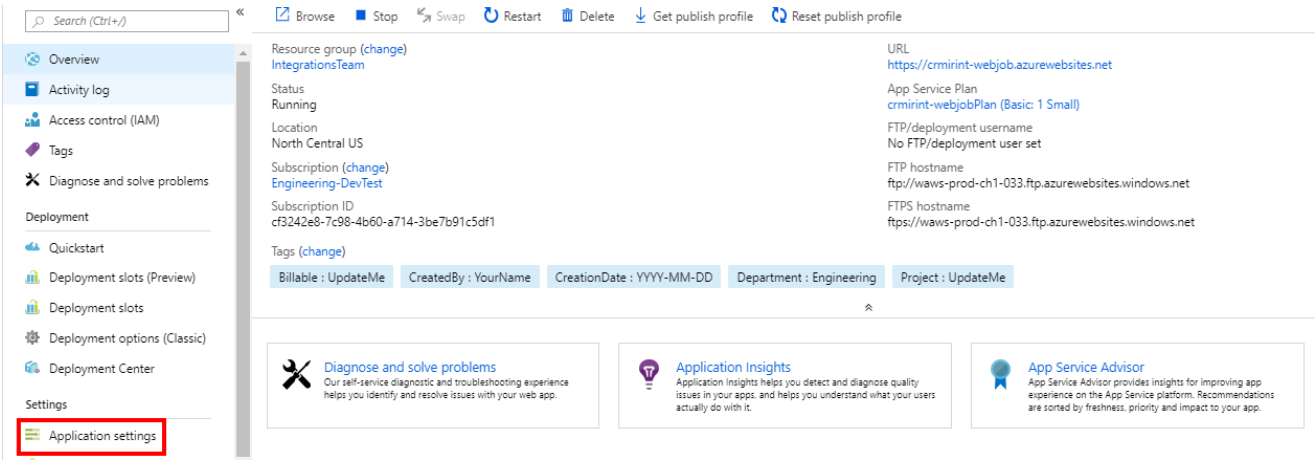
Typically, the response message at the beginning of the log and the Error Information section provide the most pertinent debugging information.

Adjusting Logging Levels

The Application Event Log can quickly become bogged down with too many logs, making finding specific logs that you may be interested in more difficult. To cut down on excessive informational logs, the Rule Execution Service, by default, will be deployed with a logging level of “Warn,” meaning only Warnings and Errors will be logged. However, this can be adjusted as needed for whatever your needs may be.

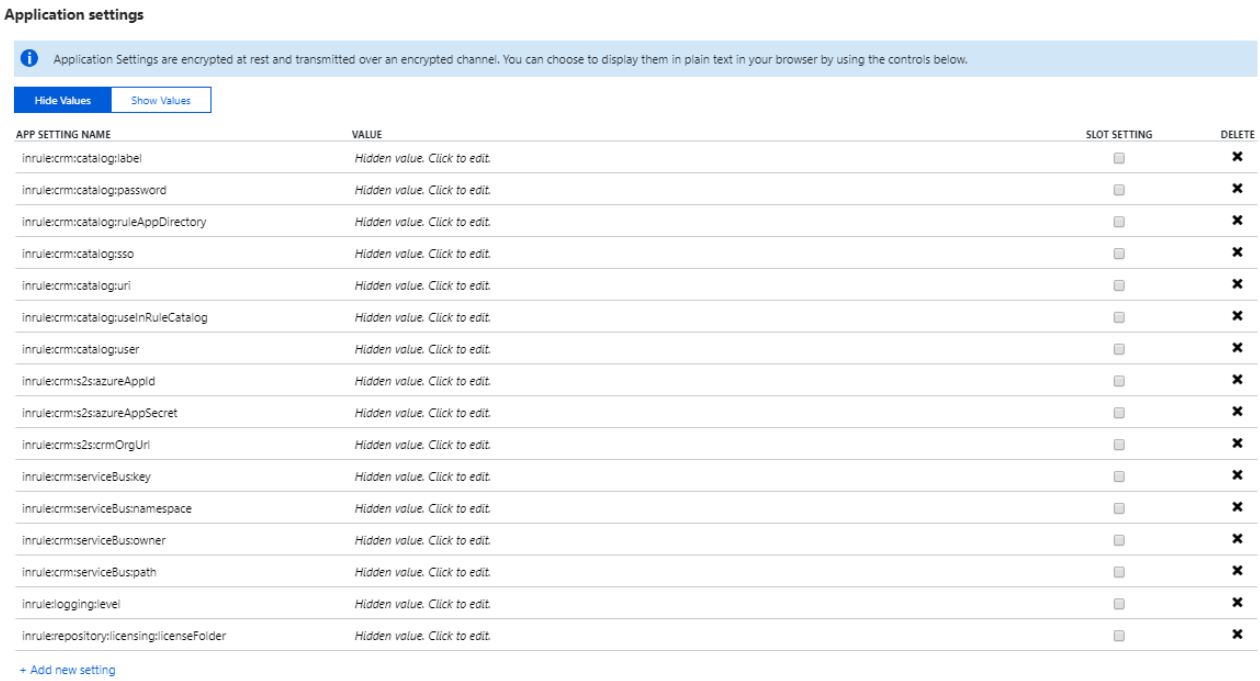
To adjust your Rule Execution Service’s logging level, navigate to your App Service as created as a part of the Azure deployment process detailed in Section 3.3.3: Rule Execution App Service for Dynamics.

Once you’re looking at the overview of your app service, select **Application Settings** in the settings menu:



Scroll down until you see the **Application settings** section:

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Locate the **inrule:logging:level** setting. Note that its value is currently set to “Warn.”

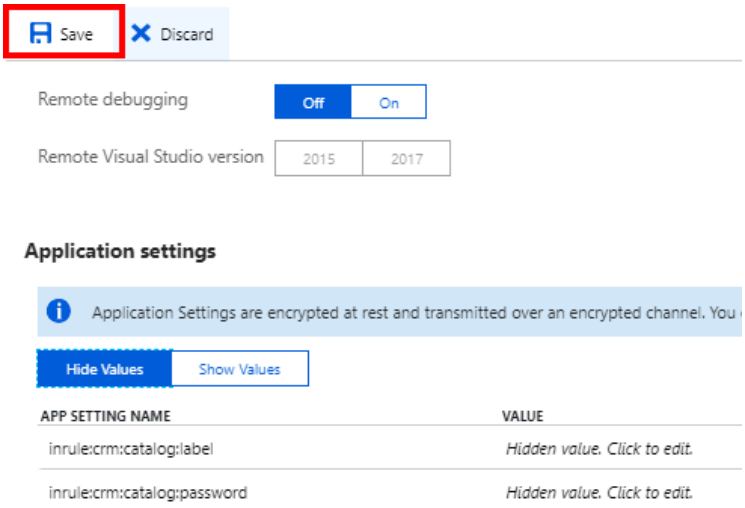
Simply change the value to the desired logging level. You may select from one of the following levels that the Rule Execution Service leverages:

Logging Level	**Details**
Info	Logs all application events, including Informational events that track the general flow of the application
Warn	Logs Warning and Error events. Warning events highlight abnormal or unexpected events in the application flow, but don’t otherwise cause application execution to stop
Error	Logs only Error events. Error events result in the halted execution of the application’s current activity due to a failure

For more detailed, technical explanations of what is included with each logging level, please reference the [Runtime Event Log Details documentation on the support site.]

Once you have configured the setting to the desired to level, press Save at the top of the page:

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix I: Activating Your License Keys

Whether you're deploying the Online or On-Prem solution will determine the appropriate method for activating your InRule licenses. For Online installations, you will have an Azure license file provided to you by InRule which will be deployed to your Azure app service via FTP. This process is detailed in the Performing the Installation: In Azure section.

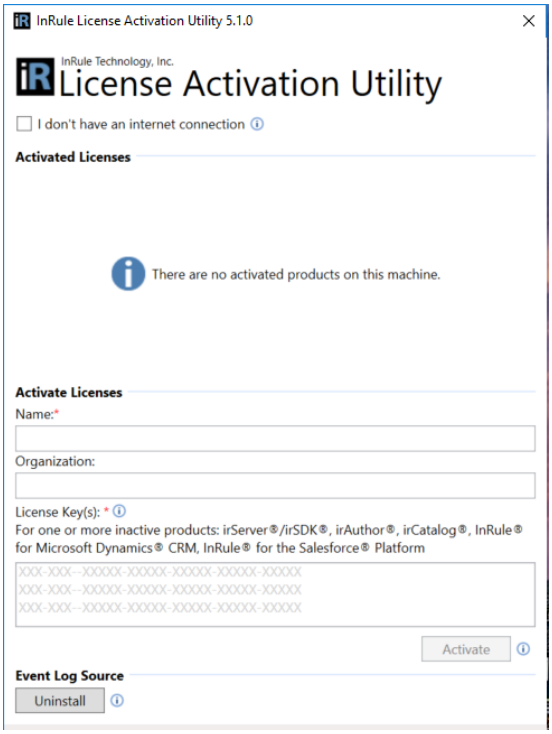
For On-Prem, you will leverage the InRule License Activation Utility and follow the walkthrough below.

1: Download the license activation utility:

Download and install the InRule Activation Utility from support.inrule.com on the server where you intend to deploy the InRule Execution Service.

2: Run the Activation Utility:

Run the activation utility as an Administrator to install Event Log Source.



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

3: Find your license keys:

Go to support.inrule.com and select Licensing Info on the left-hand navigation bar to find your irServer license keys. Which one(s) you'll need are dependent on what environment you intend to setup your InRule components in.

License Activations

[redacted] **(irServer® Production)**
Expiration: [redacted]

**** No Activations ****

[redacted] **(irServer® Disaster Recovery)**
Expiration: [redacted]

**** No Activations ****

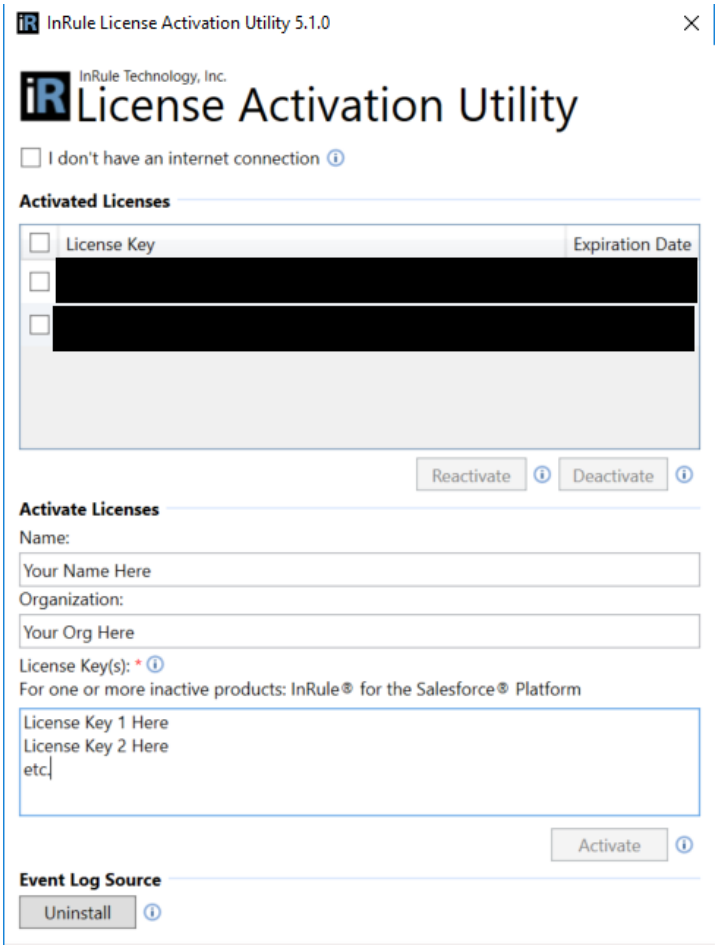
[redacted] **(irServer® Development)**
Expiration: [redacted]

CRM2016-INT (1 cores)	irServer®/irSDK® (v5.1.0.150)	7/19/2018	[redacted]

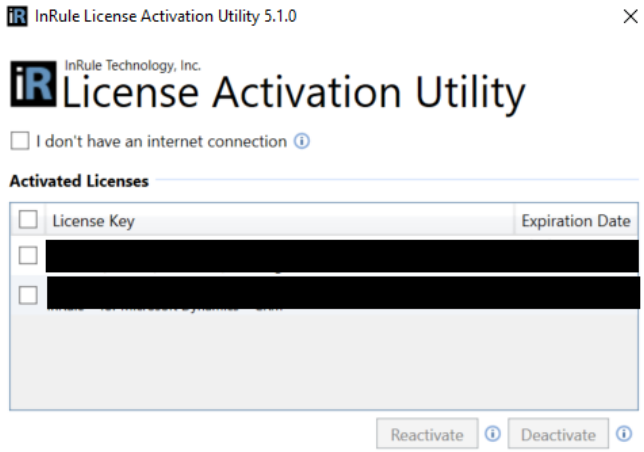
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

4: Enter your name, organization name and license keys into the Activation Utility:

Enter your name, organization and the relevant license keys into the Activation Utility and press Activate



5: Verify your license keys have activated:



Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix J: Upgrading Versions

In most cases, updating InRule for Dynamics is a relatively straight forward process. You will simply go back through the installation steps in this document and deploy the new version of the Dynamics package and execution service over the existing versions. This appendix discusses some special cases and considerations to be aware of when upgrading.

SAS Key and other Customizations

Setting the SAS Key for communicating via the Service Bus is accomplished by customizing the default endpoint included with the Dynamics solution. When this solution is re-deployed, the SAS Key is cleared out. If you are deploying the Dynamics package with the included PowerShell script, you can avoid having to re-set this value in Dynamics by ensuring you use the 'SasKey' and 'SbNamespaceAddress' parameters as documented here. If you are deploying the package via app source, you will need to set the SAS Key again after upgrading by following the steps here. Regardless of the deployment method, if you have used the Plugin Registration tool to change the 'Run in User's Context' setting on the 'InRule Custom Action Step', you will also need to set this back when updating.

Additionally, in the unlikely event you have made any other customizations to resources included in the solution, such as the JavaScript web resources, you will need to re-apply these customizations after updating the solution.

Upgrading from Cloud Service-based versions

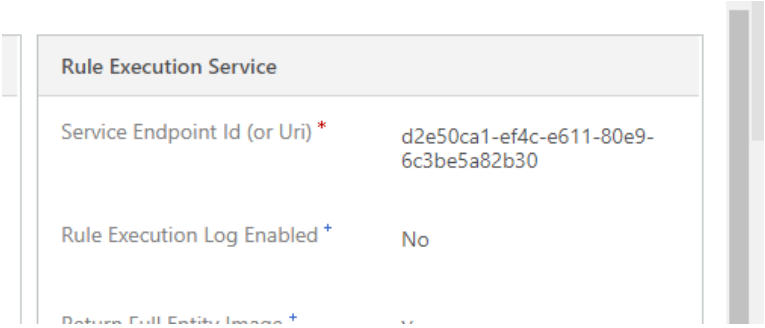
Earlier versions of InRule for Dynamics used Azure's legacy Cloud Service platform for the rule execution service. The last version to support this was 5.1.1. Versions from 5.2.0 and on now use Azure App Service for the same purpose. When upgrading from one of these versions, simply follow the steps for creating, deploying and configuring the new Azure resources. Both the App Service and Cloud Service versions use Azure Service Bus for communication with Dynamics, but in most cases, it is simpler to let the provided ARM template provision a new Service Bus for use with the App Service. Continue following the rest of the steps in the Deployment Guide to deploy the new version of the Dynamics package and configure it to point to the new Service Bus and execution service. Once you've verified the new setup is working, you can delete the old Cloud Service resources.

Switching from Dynamics On-Prem to Online

While there are no universal steps for transitioning from an on-prem to online version of Dynamics, there are a few conceptual differences in the way InRule for Dynamics works in each that you should be aware of. In on-prem installations, communication with the rule execution service is typically handled directly via HTTP. In Dynamics Online, communication is handled via an Azure Service Bus resource that manages the WCF Relay. Depending on the specifics of how your migration takes place, you may need to re-install the Dynamics package entirely, or it may still be installed on the migrated instance, along with associated config records. Either way you will need to follow all the steps in Performing the Installation: In Azure for deploying to Azure, and ensuring that the required Service Bus information is updated in Dynamics. Once you complete all these steps, you will need to update any existing InRule Configuration records to replace the URI that was in the 'Service Endpoint Id (or Uri)' field with the appropriate value (d2e50ca1-ef4c-e611-80e9-6c3be5a82b30)

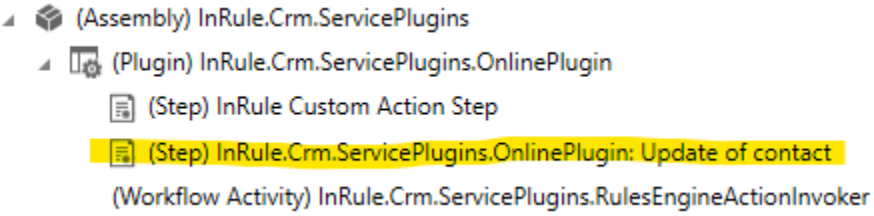
Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Using the new Entity-based Configuration

From version 5.0.28 and on, step registrations can be configured from within Dynamics. This new system uses Rules Configuration records and associates them to step registrations created in the Rules Configuration UI. Before you can use this new configuration, you will need to delete any custom registrations (registrations other than 'InRule Custom Action Step') under the OnlinePlugin:



Once you've deleted these registrations, you can create new ones from Dynamics using the steps in Appendix F: Rules Configuration and Settings

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Appendix K: Known Issues, Limitations and Troubleshooting

This portion of the document lists current known issues and limitations that may be encountered in usage of the Integration Framework. Most should only be encountered in limited edge cases.

- v5.3.0 of the Dynamics Integration Framework is only compatible with Dynamics v9.0 for both online and on-prem
- To deploy to a v8.2 Dynamics environment, view Section 4.3.4 of this deployment guide.
- In some cases, the rule helper can return nested entities from a query, which may cause performance issues.
- Status cannot be set as part of a rule app. The status field can be added to the entity schema in a rule app and can be read as part of executing a rule, but this value cannot be set by a rule.

Connections

Support for Connections is currently limited. You can bring in the Connection entity relationship through irX as well as write rules against the Connection entity, however, the ability to write rules against the Connected entity itself through the Connection relationship is not supported. For example, if you have an Account Connected to a Contact, you can write rules against the Connection entity itself and do things like change the Role, Role Type, or other fields on the Connection entity, however, you will not able to write rules against the Contact entity. The Connection entity sits between the two Connected entities (in this case Account and Contact) and contains all the Connection metadata describing the Connection.

On-Prem Execution Mode

Running the plugin in an On-Prem Dynamics environment requires running outside of Sandbox mode. Plugins in sandbox isolation mode run under partial trust, which prevents plugins from doing things like accessing the file system and registry. This prevents the execution service from logging to Dynamics' plugin trace logs. This also prevents reflection from being used, which is necessary for serializing Dynamics classes for communication over WCF. When using service endpoints to communicate with Azure Service Bus, Dynamics provides helper classes that handle this serialization, but no such classes are provided for On-Prem communication.

1-Minute Timeout

Because of Dynamics default configuration, a 1-Minute Timeout may occur when a request to the Azure Service Bus takes longer than a minute to respond. This can occur because of latency between Dynamics and the rule execution service, or anything else that causes the rule execution service to take more than 1 minute to respond.

2-Minute Timeout

Dynamics plugin execution is configured to timeout when the plugin runs for more than two minutes. In online implementations there is nothing that can be done about this behavior because the plugin cannot be taken out of sandbox mode. Since On-Prem implementations must be run outside of sandbox mode, this timeout period can be changed by updating the configuration.

S2S User Settings

If the 'InRule Integration' user is being used with the associated 'InRule Integration Administrator' role, the entity permissions are not updated after the initial creation. If a new entity type is created, it will not automatically be added to this security role and will have to be added manually. This user account can be assigned to the 'System Administrator' role to alleviate the stale permission issue, as it is automatically updated with permission when new entities are created.

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

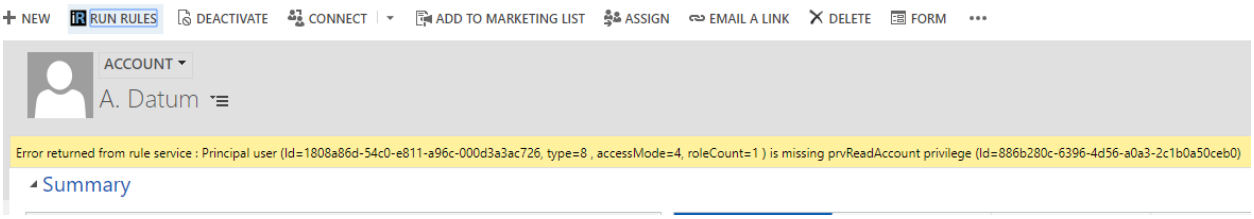
CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

Missing Entity Privilege Error

If, after attempting to run rules, you see an error like the one below, make sure that the 'InRule Integration' user has been assigned the 'InRule Integration Administrator' role, and that the role has the permissions needed to access any entities necessary.

Plugin Persistence Performance

When changes are made to entities as part of rule execution, these changes are bundled up and sent from the execution service back to the plugin for saving. Performance testing in this area indicates you can expect to be able to save around 10 changes a second, although other things can increase this time, such as other plugins registered to the entity being updated, and initial plugin start-up times. This performance constraint is specific to Dynamics overall (as compared to InRule) and has been verified with direct testing using XRM with simple entities.



Explicit Rule Sets Only

The Framework currently only supports executing Explicit Rulesets. The ability to call ApplyRules (Entity Rulesets with Fire Mode set to Auto) is currently not supported.

Common Troubleshooting Items

- When deploying the Azure WCF Relay (aka Azure Service Bus), you will need to ensure the Azure subscription you are deploying the Relay into has the Relay provider enabled. This can sometimes happen with older subscriptions that have not used relays before. For more information about this issue, refer to the link here
- When deploying plugins without isolation in an on-prem environment, dynamics requires that the user registering the plugin must be added as a Deployment Administrator from Deployment Manager. If the registering user lacks the proper permissions, when deploying the package Dynamics will return an error stating "**Assembly must be registered in isolation.**"

Copyright© 2019 by InRule Technology, Inc. All Rights Reserved.

CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.