

InRule for Salesforce

Deployment Guide

Document Updated against InRule v5.8.1 Document Updated against InRule for Salesforce v2.19.0

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 NDA 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 2024 InRule Technology, Inc.

Table of Contents

Tab	ole of C	ontents	.2
1	Introd	lucing InRule for Salesforce	.4
2	Unde	rstanding your options	. 5
2	2.1	Salesforce – InRule SaaS	. 5
2	2.2	Salesforce – Self-hosted in Microsoft Azure	.6
2	2.3	Salesforce - IIS On-Premises Rule Execution Service	.6
3	Perfo	rming the Installation	.7
3	8.1	Solution Overview	.7
3	8.2	Gathering prerequisites	. 8
3	8.3	Deploying and Configuring Components	12
	3.3.1	Catalog App Service	13
	3.3.2	Rule Execution App Service for Salesforce	17
	3.3.3	InRule for Salesforce App	25
Арр	oendix .	A: Additional Resources	37
I	nRule's	Support Website	37
I	nRule's	Support Team	37
I	nRule's	ROAD Team	37
I	nRule T	Fraining Services	37
Арр	pendix	B: Anatomy of a Request for Execution of Rules	38
Арр	pendix	C: irX General Integration Concepts	39
Арр	pendix	D: Accessing Salesforce Directly from Rule Helper	40
Арр	pendix	E: Methods for Executing Rules with Salesforce	48
1	Ac	Iding a Lightning Button	48
2	2 Ao	Iding a Classic UI Button	53
3	B Ex	cecuting Rules from Apex or Lightning Web Components	56
4	ι E>	cecuting Rules from Apex Triggers	59
5	5 Ex	ecuting Rules from Salesforce Flow	67
6	6 Ex	cecuting Rules from REST Endpoint	71
Арр	pendix	F: Azure App Service Plan & Application Insights Configuration	72
Арр	pendix	G: InRule SaaS Portal Configuration	76
Арр	pendix	H: Named Credential Configuration	79
	Settir	ng up the Named/External Credentials	79
	Provi	ding Named Credential Access via Permission Set	82
	Lega	cy Named Credential Configuration	84

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Appendix I: Salesforce and Rule Execution Service Event Logging	87
Appendix J: New Releases and Upgrading Versions	95
Appendix K: License Management	. 101
Appendix L: Known Issues, Limitations and Troubleshooting	. 104
Performance	. 107

1 Introducing InRule for Salesforce

InRule for Salesforce enables rich rule integration with Salesforce. The solution is comprised of:

- **irAuthor with irX for Salesforce** extension to irAuthor, InRule's premium desktop rule authoring tool
- InRule App for Salesforce available from the Salesforce AppExchange
- Rule Execution Services for Salesforce InRule SaaS, self-hosted in Azure, or other

This guide focuses on the deployment of the Rule Execution Services for Salesforce and corresponding InRule Salesforce App to your environment. The InRule Salesforce App may be deployed directly from the Salesforce AppExchange or from the Integration Framework zip file downloaded from the InRule Support Site.

Before beginning this guide, you may first want to familiarize yourself with the *irX for Salesforce* product by reading the *irX for Salesforce Help Documentation*. This irAuthor extension will allow you to author rules against Salesforce 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 Salesforce. At this point, this guide will become highly relevant for deploying the InRule solution and services to establish the selected integration patterns.

This document also provides an addendum, <u>Appendix E: Methods for Executing Rules from Salesforce</u> that discusses the different options available to you for running rules beyond what this Deployment Guide covers. It is a good next step 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 <u>Appendix A:</u> <u>Additional Resources</u> for support website detail. To find what's new, see <u>Appendix J: New Releases and</u> <u>Upgrading Versions</u>.

This guide assumes that the reader has basic familiarity with Salesforce, irAuthor and irX for Salesforce.



Important: Not all Salesforce environments support API access. However, InRule for Salesforce requires a Salesforce instance that supports the API. If you attempt to connect with an instance that does not support this access, you will receive an error when connecting that contains information such as "The REST API is not enabled for this Organization." The edition types that do NOT include API access are Contact Edition, Group Edition and Professional Edition. Please refer to official Salesforce documentation for up-to-date information as this list is subject to change.

2 Understanding your options

Salesforce is available only as an online SaaS offering, but the Rule Execution Service and Catalog Service components of InRule for Salesforce can be hosted anywhere an IIS site can be deployed. This guide assumes deployment in Azure, but an on-prem or VM environment can also be used if required.

The following sections describe which deployment scenario will be the most applicable depending on your needs:

- <u>2.1 Salesforce InRule SaaS</u>
- <u>2.2 Salesforce Self-hosted in Microsoft Azure</u>
- <u>2.3 Salesforce IIS On-Premises Rule Execution Service</u>

The primary consideration for InRule SaaS vs self-hosted Azure generally depends on if your organization is already setup to manage an Azure subscription and services.

- **If your organization does not have an Azure subscription**, this can be compelling rationale to go with InRule SaaS and forgo the overhead of Azure management and deployment.
- If your organization has an Azure subscription, it does not exclude you from choosing InRule SaaS, but it may indicate that your IT department has requirements or preferences for self-managed Azure solutions.

Either way, this guide will provide you with the information to help determine which InRule configuration will best suit your needs, including Government Cloud and other considerations.

2.1 Salesforce – InRule SaaS

InRule for Salesforce is now available via <u>InRule SaaS</u>, in which case many of the deployment steps in this document are not applicable and will be managed for you. This is the most stream-lined deployment available to both qualified Trial users and licensed customers.



Important: If you are new to InRule and do not have an InRule SaaS subscription, you can request a <u>free trial here</u> and specify that you would like to integrate it with your Salesforce instance.

With InRule SaaS, the deployment steps are significantly reduced by eliminating the need to install the InRule's App Services in Azure. This narrows the deployment focus down to irAuthor with irX for Salesforce and the InRule for Salesforce App from the Salesforce AppExchange. The main deployment steps in this scenario are summarized as follows:

Deployment Step	Installation Type
1. irAuthor with irX for Salesforce	InRule installer via the InRule Support Site
2. Rule Execution Services for Salesforce	*managed by InRule SaaS Support
3. InRule for Salesforce App	Salesforce AppExchange

All information to set up the connectivity between Rule Execution Services hosted by InRule SaaS and your organization's Salesforce instance can be managed through the InRule SaaS Portal. The following configuration is what will be managed through the Portal. More information can be found in <u>Appendix G:</u> <u>InRule SaaS Portal Configuration</u>.

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Entered by users in the SaaS Portal - configuration information for the Rule Execution App Services

- Service account credentials for Salesforce, including that account's API security token used to allow the InRule SaaS-hosted Rule Execution Service to connect to Salesforce.
- Salesforce Connected App Consumer Key and Consumer Secret information -- used to allow the InRule SaaS-hosted Rule Execution Service to connect to Salesforce.

Provided to users in the SaaS Portal – configuration information for the InRule Salesforce Package:

- Rule Execution App Service endpoint URL used for allowing Salesforce to make requests out to the Rule Execution App Service
- Rule Execution App Service access credentials -- used for allowing Salesforce to make requests out to the Rule Execution App Service

The net effect of an InRule SaaS deployment is that after the above pre-requisites are met, you get to *skip ahead* to <u>Section 3</u>: <u>Configuring the InRule for Salesforce App</u>. Beyond this initial setup the remainder of the guide will be beneficial for detailing <u>advanced rule execution methods</u> and troubleshooting.

2.2 Salesforce – Self-hosted in Microsoft Azure

The suggested setup is to install InRule using a Platform-As-A-Service (PaaS) model on Microsoft Azure. This document discusses the following three components:

- 1. Catalog App Service and Azure SQL Database
- 2. Rule Execution App Service for Salesforce
- 3. InRule Decision Client package for Salesforce

Section 3 of this document, <u>Performing the Installation: In Azure</u>, provides a complete walkthrough for setting up each of these components.

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

2.3 Salesforce - IIS On-Premises Rule Execution Service

If for any reason you cannot use the Azure App Service, the Rule Execution Service and Catalog Service can also be installed in an on-prem environment, or in any IaaS server configured with IIS. The following three components are required:

- 1. Catalog Web Service and Database
- 2. Rule Execution Web Service for Salesforce
- 3. InRule Decision Client package for Salesforce

Installation Documentation and InRule Installer, which will be used to install the Catalog Service, are available on <u>our support website's downloads section</u> and provide instructions for setting up the catalog.

Detailed installation steps are not included with this guide, but the InRule.Salesforce.WebService.zip package in the RuleExecutionAzureService folder can also be deployed in an on-prem environment. This is a Web Deploy package, so it can be deployed via MSDeploy or by copying the contents to an IIS Site. Alternatively, an MSI is also provided, located in the same file directory as the aforementioned Web

3 Performing the Installation

This section discusses the steps needed to integrate InRule with Salesforce. Using this scenario InRule components are hosted on Microsoft Azure.

3.1 Solution Overview

This guide will provide the instructions for setting up all 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.

Rule Execution App Service for Salesforce

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

InRule for Salesforce App

The InRule for Salesforce App contains Apex classes, as well as custom settings and objects. It is configured to communicate with the execution service over REST.

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.

Optional Resource Files

The following file can be downloaded from our support website's downloads section:

• InRule for Salesforce.zip

This zip file contains several resources that can be used in advanced deployments. A typical installation does not require the use of these resources, as all the required deployment assets are available online. However, this zip file provides an alternative means of accessing the InRule for Salesforce assets.

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:

```
InRule for Salesforce.zip

InRule for Salesforce

readme.txt

RuleApplications

SalesforceRules.ruleappx

RuleExecutionAzureService

azuredeploy.json

azuredeploy.parameters.json

InRule.Salesforce.WebService.zip

RuleHelperDeployment

InRule.Salesforce.RuleHelper.dll

...(many other supporting files)
```

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Rule Authoring Environment

A rule authoring environment is used to upload a Rule Application to your Catalog Service. A rule authoring environment is a machine or virtual machine where irAuthor has been installed with the irX for Salesforce extension. If you followed the instructions outlined in *irX for Salesforce 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 Salesforce license to activate the related components in the authoring environment. If you are a system administrator who does not intend to perform rule authoring activities 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

Salesforce Service Account

You will need an existing or new Salesforce account to use as a service account to connect to the Salesforce API. This account will be used by the rule execution service to load and save data from Salesforce required by rules. This account does not need to be a System Administrator account, but it will need permissions to use the Salesforce API and interact with entities used by rules.

Security Token

If the account does not already have an API token, then log into the Salesforce portal and go to the user Settings from the avatar menu in the top-right. On the sidebar that appears, navigate to My Personal Information \rightarrow Reset My Security Token. On the page that appears, click the 'Reset Security Token', and you should receive an email within a few minutes that contains an API token.

Important: Security tokens are, in most scenarios, required. They are only optional in scenarios when the account attempting to authenticate is connecting from a Trusted IP address. Trusted IPs can be set globally, meaning any user that connects from an IP in the trusted range will be regarded as trusted, or on the Profile level, meaning that only users assigned to the given Profile will be regarded as trusted when connecting from an IP that falls in the trusted range. But regardless of how Trusted IPs are configured, if the account in question does not fall in a configured Trusted IP range, it must have a security token set.

If trusted IP ranges are enabled for this account, you will not see the 'Reset My Security Token' button. If this is the case, simply leave this value blank where required later.



Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Sales Home Chatter	Opportunities 🗸 Leads 🗸 Tasks 🗸 Files 🗸 Accounts 🗸 Cor
Q Quick Find	Reset My Security Token
Advanced User Details Approver Settings	Reset Security Token
Authentication Settings for External Systems	When you access Salesforce from an IP address that isn't trusted for your company, and you us also reset.
Change My Password Connections	After you reset your token, you can't use your old token in API applications and des
Grant Account Login Access	Reset Security Token
Language & Time Zone	
Login History	
Personal Information	
Reset My Security Token	
Security Central	

Other Credentials

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

API Key for the Rule Service: The rule execution service is protected using an API key. For SaaS deployments, your API key can be found in the InRule Portal by navigating to "Provisioned Resources" and finding the "Execution" resource. For Self-hosted deployments, you can generate your own key, but it is important to save this key as it will be used when configuring Salesforce later.

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 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

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.

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
Rule Execution App Service Plan Name	MyAppInRuleDevRuleExecutionAppServicePlan
Rule Execution App Service Name	MyAppInRuleDevRuleExecutionAppService

Enabling OAuth Username-Password Flow

Starting with the Summer '23 Salesforce update, new Salesforce instances now have the Allow OAuth Username-Password Flows setting disabled. In order for InRule to authenticate back to Salesforce, you will need to verify that this setting is enabled.

To begin, navigate to the OAuth and OpenID Connect Settings page by using the quick find search box. Once there, find the Allow OAuth Username-Password Flows toggle and enable it.



3.3 Deploying and Configuring Components

There are two primary paths for deploying the required InRule components:

- App Stores: <u>Azure Marketplace</u> and <u>Salesforce AppExchange</u> (Recommended) Provides a straightforward, UI-driven process that simplifies deployment and eliminates the need to individually deploy both the Catalog App Service and the Rule Execution App Service. Both the Azure Marketplace (for InRule App Services) and the Salesforce AppExchange (for the InRule for Salesforce App) must be utilized to deploy their respective InRule Apps (see matrix below).
- 2. **Manual Deployment using ARM Templates and Scripts**: Manually deploy <u>Azure Resource</u> <u>Management (ARM) Templates</u> through either the Azure Portal, PowerShell, or Azure CLI; then deploy the InRule for Salesforce App via a versioned package link to your Salesforce environment.

InRule components by deployment path:

InRule Component	App Stores	Manual Deployment
Catalog App Service	Azure Marketplace	ARM Template
Rule Execution App Service for Salesforce		ARM Template
InRule for Salesforce App	Salesforce AppExchange	Versioned Package Link

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

While deploying from Azure Marketplace and AppExchange is generally recommended for initial environments; in advanced scenarios the manual deployment resources can be leveraged for multienvironment DevOps automation. Also note, choosing either path does not require you to stay on the same path across all components; as the deployment options can be mixed and matched as situations warrant.

The following sections are primarily focused on the manual deployment path for the Catalog App Service and Rule Execution App Service for Salesforce, while focusing on the AppExchange deployment path for the InRule for Salesforce App. However, there are still relevant config and testing steps which apply to both paths. For Marketplace deployments, you may skip the manual deployment steps and go directly to the relevant sub-sections for license management, configuration, and setup verification.

3.3.1 Catalog App Service

Installing the Catalog App Service

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

During this process, you will be creating:

- 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/blob/master/README.md#ircatalog-and-ircatalog-manager

Please ensure you are installing the Catalog service, not the irServer Rule Execution Service, which is found on the same page and is not compatible with Salesforce.

Testing the Catalog App Service

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. We'll now verify that we can appropriately connect to it.

To begin, open up irAuthor. By default, the Catalog App Service is deployed out with a test rule app already available on it. We can attempt to access this rule app to verify that everything is working correctly.

1: From the irAuthor launch screen, navigate to File -> Open -> Open from Catalog

FILE	FILE					
\bigtriangleup	Start Page		Open			
				Open from <u>Catalog</u> Open a rule application from an InRule Catalog		
	New			Onen from File		
	0			Open a rule application from the file system		
2	<u>O</u> pen	ŗ.,		Open Rule Trace File		
	<u>S</u> ave			application		
	Save <u>A</u> s	Þ				
	Save a Cop <u>v</u>					
	<u>C</u> lose					
?	<u>H</u> elp	Þ				
				Extensions 🖾 Optjons 🗙 Exit		

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

E Log in to catalog	×
🕂 Add Catalog 🗲	
myappinruledevcatalog	service.cloudapp.net Use this
Catalog name:	myappinruledevcatalogservice.cloudapp.net
Use Single Sign-On:	
Username:	admin
Password:	
Catalog URI:	https://myappinruledevcatalogservice.cloudapp.net/service.svc
	$\wedge \Psi \times$
Advanced options	\odot
	Close
Use versions:	

3: Select the SalesforceRules rule app and hit Open

Name	Revision	Used in	
SalesforceRules	98	N/A	
			Open Cancel

If the SalesforceRules rule app is there and opens without issue, then your catalog app service is good to go.

If the rule app does not appear as an option to open after connecting to the catalog, you can try the below steps as an alternative method of testing:

4: Download the <u>SalesforceRules sample rule app.</u>

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

📕 🛃 🗖 🖛 RuleApplications		
File Home Share View		
$\leftarrow \rightarrow \ \ \star \ \ \star \ \ \bullet \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	alesforce Platform >	RuleApplications
Name ^	Status	Date modified
R SalesforceRules.ruleappx	\odot	4/17/2019 5:17 PM

3: Save the Rule Application by choosing File \rightarrow Save As \rightarrow Save to Catalog

🖪 🔒 🕞 🎯 🖉 🦻	ē⊒ -	SalesforceRules - InRule irAuthor
FILE		DEVELOPER
Start Page	Save As	🕂 📩 👘 Move Up 🛨 Add -
× New	Save to <u>Catalog</u> Save the rule application to an InRule Catalog	Paste Add Duplicate Move Down S Insert - Categorize Rt Appli
	Save to <u>File</u> Save the rule application to the file system	rd Item Tags
		SalesforceRules
<u>≥</u> ave		Mixed •
Save <u>A</u> s		✓ Enabling/disabling of rule sets ③
Save a Copy		
Close		Edit Rule Templates and Classifications to create a custom vocabulary. Go to vocabulary
Pelp		1
	Extensions 🖂 Options 🗙 Exit	30000 (ms)
Pula Flaura	Maximum cycle cou	nt: 100000
Kule Flows		

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

E Log in to catalog	×
🕂 Add Catalog 🔫	
myappinruledevcatalog	service.cloudapp.net -> Use this
Catalog name:	myappinruledevcatalogservice.cloudapp.net
Use Single Sign-On:	
Username:	admin
Password:	
Catalog URI:	https://myappinruledevcatalogservice.cloudapp.net/service.svc 5
	$\uparrow \Psi \times$
Advanced options	· · · · · · · · · · · · · · · · · · ·
	Close
Use versions:	

7: Save with the name SalesforceRules

R Save			\times
Name:	SalesforceRu	les	
Action:	Oheck In	Leave checked out	
	O Save to the	admin' workspace	
Save as:	New		
	○ Existing:		*
✓ Label:	New:	LIVE	
	O Existing:		*
Comments:			
Catalog in	formation		
		OK Cano	el

Save the Rule Application to the Catalog using the name of *SalesforceRules*. You must also be sure to provide the label, which needs to match the label configured on the Catalog Service.

At this point, if you can click OK without an error, you have successfully saved the SalesforceRules Rule App to the new Catalog that you have created.

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

Upload License File

Next, you'll need to upload a license file to the web app service in order for the Rule Execution App Service to properly function. The simplest way to upload the license file is via the Azure App Service Editor. Alternatively, you can deploy the license file via FTP. Walkthroughs of both these approaches can be found in <u>Appendix K: License Management.</u>

3.3.2 Rule Execution App Service for Salesforce

Next, we will deploy the InRule Rule Execution service as an Azure App Service, along with all its Azure resource dependencies. To make this process easier, we will 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**.

Alternatively, while this document does not provide a walkthrough of it, the ARM template provided is configured to work with **Azure Portal** deployment. For an overview of how to leverage ARM deployment through the Azure Portal, reference Microsoft's documentation: <u>Deploy resources with ARM templates</u> and <u>Azure portal</u> and navigate to the section titled "Deploy resources from custom template".

You can navigate directly to the Azure Portal page for deploying an ARM template at this link: <u>Custom</u> <u>deployment - Microsoft Azure</u>.



1: Locate azuredeploy.parameters.json

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

The required *azuredeploy.json* and *azuredeploy.parameters.json* files can be downloaded here - AzureAppServices/Salesforce at master · InRule/AzureAppServices · GitHub.

Alternatively, they can be located within the *InRule for Salesforce.zip* file downloaded in <u>Section 3.2</u>: <u>Optional Resource Files</u>.

📕 🛃 🚍 🗢 RuleExecutionAzureServic	e		
File Home Share View			
$\leftarrow \rightarrow$ \checkmark \uparrow \blacksquare \ll InRule for the Sales	force Platf >	RuleExecutionAzureService	~
Name ^	Status	Date modified	Туре
🖵 azuredeploy.json	Ø	4/17/2019 5:17 PM	JSON
azuredeploy.parameters.json	\odot	4/17/2019 5:17 PM	JSON
Deploy-AzureResourceGroup.ps1	\odot	4/17/2019 5:17 PM	Wind
InRule.Salesforce.WebService.zip	Ø	4/17/2019 5:17 PM	Com

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

2: Update parameters

Before completing this section, make sure you have the following credentials:

- Salesforce Credentials
- Salesforce API Security Token
- Salesforce Connected App
- InRule Catalog credentials

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

```
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
"contentVersion": "1.0.0.0",
"parameters": {
 "appServiceName": {
    "value": ""
  "sfLoginUrl": {
   "value": "https://login.salesforce.com/services/oauth2/token"
  "sfUsername": {
   "value": ""
  },
 "sfPassword": {
   "value": ""
  "sfSecurityToken": {
   "value": ""
  },
  "sfConsumerKey": {
   "value": ""
 "sfConsumerSecret": {
   "value": ""
  },
  "catalogUri": {
   "value": ""
  "catalogUser": {
   "value": "admin"
 },
  "catalogPassword": {
   "value": "password"
  "ruleAppLabel": {
   "value": ""
 "executionServiceApiKey": {
   "value": ""
  "appServicePlanName": {
    "value": ""
  },
 "createOrUpdateAppServicePlan": {
   "value": true
  },
  "inRuleVersion": {
   "value": "5.8.0"
  },
  "appInsightsResourceName": {
    "value": ""
  },
  "appInsightsInstrumentationKey": {
   "value": ""
  },
  "appInsightsConnectionString": {
    "value": ""
  3
```

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved. CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

1. appServiceName	Provide a name for the Azure App Service that the Rule Execution Service will run on.
	Note, by default, an App Service Plan will be created by the ARM template. The App Service Plan will follow the naming convention of the App Service name you provide here, with "Plan" appended to the end (ex. appServiceNamePlan). 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 <u>Appendix F: Azure App Service Plan and Application Insights Configuration</u>
2. catalogUri	The URI for the Catalog Service that will be used Example: <u>https://myappinruledevcatalogservice.cloudapp.net/service.svc</u>
3. catalogUser	Username for Catalog Service, default value is 'admin'.
4. catalogPassword	Password for Catalog Service, default is 'password', please change this using the catalog manager!
5. ruleAppLabel	Optionally supply a default label used by the execution service to identify the ruleapp version to run.
6. sfLoginUrl	The default for this is set to https://login.salesforce.com/services/oauth2/token , but you can change this to the relevant URL for your instance if you are deploying to something like a sandbox instance
7. sfUsername	Salesforce username for the service account used to connect from the execution service to the Salesforce API
8. sfPassword	Password for the Salesforce service account
9. sfSecurityToken	API token for the Salesforce service account. Security tokens are required unless the service account will be connecting from an IP address that falls in a trusted IP range configured in Salesforce.
10. sfConsumerKey	OAuth consumer key for the <u>connected app</u>
11. sfConsumerSecret	OAuth consumer secret for the <u>connected app</u>
12. executionServiceApiKey	Api key used to secure the rule service. This same api key will need to be provided to Salesforce during configuration
13. inRuleVersion (To deploy most modern version, leave as default value)	This parameter allows the user to configure what version of the InRule Rule Execution Service they wish to deploy. By default, this parameter will be set to the most modern version.
14. 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 <u>Appendix F: Azure App Service Plan and Application Insights Configuration</u>
15. createOrUpdateAppServicePlan	By default, this value is set to true, and 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, set this value to false and reference <u>Appendix F: Azure App</u> <u>Service Plan and Application Insights Configuration</u> .
16. appInsightsResourceName (If you wish to create a new Al resource)	If you want to use Application Insights as a log sink in addition to the app service logging already enabled, but do not already have an Insights resource that you want to use, specify a name for a new resource here. Specifying a value for this parameter will create a new Application insights resource with the given name and populate the instrumentation key app setting on the app service with the key from this new resource. If you provide a value for this parameter, do not provide a value for applnsightsInstrumentationKey.
17. appInsightsInstrumentationKey (If you wish to use an existing Al resource)	Provide an Instrumentation Key if you have an existing App Insights resource you'd like to use for logging and telemetry. If you are configuring this in a nonstandard azure environment (such as Azure Government), please additionally provide an App Insights Connection String. Otherwise, leave this value blank and provide a value for the 'appInsightsResourceName' parameter, which will create the resource for you. For more information on the logging view <u>Rule Execution Service Event Log</u> .

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

18. appInsightsConnectionString (If you wish to use an existing Al resource in a non-standard Azure environment)	Only override the default value here if you have an existing App Insights resource you'd like to use, AND you need to use a non-standard connection string. If you need to supply your own connection string, be sure to set that value here, as well as providing your instrumentation key in the appInsightsInstrumentationKey parameter. If you want this template to manage the App Insights resource for you, or only need to provide an instrumentation key, leave this value as the default and provide values for appInsightsResourceName or appInsightsInstrumentationKey instead.
	Additionally, for any consideration about using app insights or setting it up in a non- standard Azure environment view <u>Appendix G: Azure Application Insights Configuration</u>

Once you have finished configuring your parameters, save the completed parameters file and keep a spare copy on hand for future upgrades or automation.

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

Command Prompt

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

Microsoft	
Sign in	
Email, phone, or Skype	
Can't access your account?	
No account? Create one!	

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:

:\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:\>az deployment group create -g ResourceGroupName --template-file .\azuredeploy.json --parameters .\azuredeploy.parmeters.json

Observe that the template deploys with no errors

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

🚬 Windows PowerShell

1.2 Navigate to the RuleExecutionAzureService folder

Windows PowerShell
PS C:\> cd .\RuleExecutionAzureService\
PS C:\RuleExecutionAzureService>

1.3 Enter "Connect-AzureRmAccount" to login into Azure

PS C:\RuleExecutionAzureService> Connect-AzureRmAccount

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:

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:

S 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

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

Observe that the template deploys with no errors

5: Verify Setup

Navigate to the Azure portal and locate the deployed App Service

1 of 2 items selected Show hidden types 👩	
NAME TU	TYPE ↑↓
✓ 🔇 armTestAppService	App Service

Ensure that all of the app settings are configured correctly for your setup

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved. CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

O Search (Ctrl+/)		
🛷 Tags	Application settings General settings Default d	documents Path mappings
X Diagnose and solve problems		
Security	Application settings	
Deployment	+ New application setting 💿 Show values 🖉	Advanced edit
🕰 Quickstart	Name	Value
Deployment slots	aspnet:UseTaskFriendlySynchronizationContext	Hidden value. Click show values button above t
G Deployment Center	inrule:logging:level	Hidden value. Click show values button above to
Configuration	inrule:repository:licensing:licenseFolder	Hidden value. Click show values button above t
Authentication / Authorizati	inrule:sf:auth:consumerKey	Hidden value. Click show values button above t
Application Insights	inrule:sf:auth:consumerSecret	Hidden value. Click show values button above t
Backups	inrule:sf:auth:loginUrl	Hidden value. Click show values button above t
Custom domains	inrule:sf:auth:password	Hidden value. Click show values button above t
SSL settings	ing desting the security Taken	Hidden value. Click chew values butten abeve t

6: Upload License File

Next, you'll need to upload a license file to the web app service in order for the Rule Execution App Service to properly function. The simplest way to upload the license file is via the Azure App Service Editor. Alternatively, you can deploy the license file via FTP. Walkthroughs of both these approaches can be found in <u>Appendix K: License Management.</u>

3.3.3 InRule for Salesforce App

At this point, all the Azure requirements are met. The execution service should be listening for incoming communication from Salesforce. We must now setup the InRule for Salesforce App, which is done with a managed Salesforce package. This can be done via either AppExchange or direct install link:

- Deploy <u>latest</u> version navigate to the <u>AppExchange</u>, select 'Get It Now'
- Deploy <u>specific</u> version deploy the <u>specific version</u> associated to this Deployment Guide (refer to cover page for version)

1: Install from AppExchange:

InRule for Salesforce can be installed from the Salesforce AppExchange marketplace. You can either search for 'InRule for Salesforce' or go directly to the listing here:

<u>https://login.salesforce.com/packaging/installPackage.apexp?p0=04tHu000003EZBV</u> When you get to the listing page, you'll need to select 'Get It Now' and choose the org you want to install the package in.



	Where do you want to install this package?	
	Install in a Production Environment	
Ir	nstall this package in the org where you or your users work, including Developer Edition orgs.	
*	Connected Salesforce Accounts 🕦	
	sfqa1@inrule.com	
	Install in Production	
	Install in Production	
	Install in Production Install in a Sandbox Test this package in a copy of a production org.	
	Install in Production Install in a Sandbox Test this package in a copy of a production org. Install in Sandbox	
	Install in Production Install in a Sandbox Test this package in a copy of a production org. Install in Sandbox	

Once you've selected an org and logged in, you'll see a screen prompting you to install the package. Select 'Install for all users' and hit ok. Installing this package installs all the components required for integrating with InRule, including a configuration app, Apex classes, and custom settings and objects. Once the installation is complete, you will need to configure the connection to the execution service.

2: Navigate to the InRule for Salesforce App

In Lightning view, select the Salesforce App Launcher button in the top right of the home screen and select "View All"

	Setup	Home	Object Manager	~
0	Search app	s and items		
App	os			
Rþ	Service			
0	Marketing			
ß	Communit	у		
4	Salesforce	Chatter		
	Content			
	Sales Cons	ole		
0	Service Cor	nsole		

Select the InRule App

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved. CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

All Ap	ps				
₽)	Service Manage customer service with accounts, contacts, cases, and more	1	0	Marketing Best-In-class on-demand marketing automation	
≁	Salesforce Chatter The Salesforce Chatter social network, including profiles and feeds	1	Þ	Content Salesforce CRM Content	
0	Service Console (Lightning Experience) Lets support agents work with multiple records across customer service channels on one screen	1	Ø	Sales Manage your sales process with accounts, leads, opportunities, and more	
(3)	Bolt Solutions Discover and manage business solutions designed for your		fR	InRule	1

This app provides useful information on how to get started using InRule for Salesforce, as well as quick links to important configuration pages and logging.

Q bean	🖈 🖬 🏩 ? 卒 🐥 🐧
InRule Getting Started Configuration Running Rules InRule Logs V	
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	
Configuring the InRule App	INRULE
Once the dependencies outlined in the Getting Started section are successfully deployed, the InRule App can now be configured within Salesforce. When the Connecte configuration is complete, test the Rule Execution Service connection by clicking the following button:	d App and Named Credential
Test Connectivity	
Configuration includes the following three steps:	
Setup a Connected App	
Establish the framework for the Rule Execution Service to communicate with Salesforce	
> Connected App Details	
Setup Named Credential	
Establish the credentials for Salesforce to authenticate with the Rule Execution Service	
> Named Credential Details	
> Named Credential Access Details	
Setup Custom Settings	
Manage Rule App Name, logging, and caching configurations	
> Custom Settings Details	

# 3: Configure the InRule App

#### **Connected App**

First, create a new Connected App in Salesforce that is configured for OAuth authentication. Navigate to Setup  $\rightarrow$  App Manager -> New Connected App

Lightning Ex	sperience App Manager	New Lightning App New Connected Ap
ip Home		
ice Setup Assistant		
ti-Factor Authentication stant		
ase Updates		
tning Experience Transition stant		
/ Salesforce Mobile App ckStart		
tning Usage		
mizer		
INISTRATION		
iers		
ata		
mait		
TFORM TOOLS		
pps		
App Manager		
AppExchange Marketplace		
Connected Apps		
Lightning Bolt		
Mobile Apps		

Fill out the Basic Information and configure OAuth authentication. The Rule Execution Service uses the Username-Password authentication flow, so there is no callback URL, but because Salesforce requires this field to be populated, enter any properly formatted URL; it will not be used. Also, be sure to add "Full access" to the Selected OAuth Scopes. Any other values can be left as defaults.

Require Secret for Web Server Flow®

Basic Information	
Connected App Name	InRule Rule Execution
API Name	InRule_Rule_Execution
Contact Email	person@inrule.com
Contact Phone	
Logo image OKL	Upload loop image or Choose one of our sample loops
Icon URL®	
	Choose one of our sample logos
Info URL	
Description	
<ul> <li>API (Enable OAuth Settings)</li> </ul>	
Enable OAuth Settings	
Enable for Device Flow	
Callback URL	https://login.salesforce.com/service/oauth2/callback
Use digital signatures	
Selected OAuth Scopes	Available OAuth Scopes Selected OAuth Scopes
	Access and manage your Wave data (wave api)
	Access and manage your data (api)
	Access custom permissions (custom permissions)
	Access your basic information (id, profile, email, address, phone)
	Allow access to Lightning applications (lightning)
	Allow access to content resources (content)
	Allow access to your unique identifier (openid) Remove
	Perform requests on your behalf at any time (refresh token, offline access)

After saving, select the newly created Connected App and inspect the OAuth settings. Record the Consumer Key and the Consumer Secret.

Provide access to custom applications (visualforce) Provide access to your data via the Web (web)

Connected App Name Help							
« Back to List: Cust	iom Apps	Edit	Delete Manage				
		Version	1.0				
		API Name	irX_for_Salesforce				
		Created Date	5/10/2016 8:45 AM				
			By These Available				
		Contact Email	inside and the second second				
		Contact Phone					
		Last Modified Date	9/13/2016 9:22 AM				
			By.				
	1	Description					
		Info URL	•				
▼ API (Enable O	Auth Settings)						
Consumer Key	Charles PP 2	197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197		Consumer Secret	Click to reveal		
Selected Fu OAuth Scopes	II access (full)			Callback URL	https://ap1.salesforce.com /services/oauth2/token		

If your Salesforce instance was created **after** the Summer '23 update, verify that you have enabled the 'Allow OAuth Username-Password Flows' toggle on the OAuth and OpenId Connect Settings page.

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved. CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.



Once you have completed these steps, you should have the token, consumer key and consumer secret, all of which are required for authentication with Salesforce in addition to the username and password.

#### Named Credential

Next, we need to configure the Named Credential in Salesforce. This will be used by the Apex code to make secure HTTP requests to the Rule Execution Service. See <u>Appendix H: Named Credential</u> <u>Configuration</u> for instructions on how to set up your Named Credential.

#### **Custom Setting**

Once you have set up the Named Credential, navigate back to the InRule App and return to the configuration tab. Under the "Custom Settings" header, click the "Custom Settings" shortcut link to configure your default rule app name and logging level.

✓ Custom Settings Details
Go to Custom Settings
The InRule Custom Settings object is installed as a part of the InRule App but needs to be configured to be used. To begin, click the link above and select InRule $ ightarrow$ Manage $ ightarrow$ Edit
When configuring the Custom Setting object, provide the following settings:
• Rule App Name: The name of the rule app that will be loaded by the Rule Execution Service when running rules unless otherwise specified by the Lightning component configuration.
• App Domain Cache: An integer that defines the amount of time in seconds that the InRule RuleHelper will hold entities and collections in cache after querying for them. The following button can be used to clear the app domain cache for all RuleHelper queries.

#### Click "Manage" next to the InRule label

SETUP Custom Setti	nas					
						Halo for this f
tom Settings						
stom Settings						
stom Settings istom settings to create an force. Apex, and the Web !	d manage custom data at t Services API.	the organization, profile, and use	ar levels. Custom settings data is stored in th	e application cache. This means you can acc	ass it efficiently, without the cost of repeated queries. Custom settings	data can be used by formula fields
stom Settings istom settings to create an force. Apex, and the Web t	d manage custom data at t Services API.	the organization, profile, and use	r levels. Custom settings data is stored in th	e application cache. This means you can acc Get Usage	ass it efficiently, without the cost of repeated queries. Custom settings	data can be used by formula fields
stom Settings istom settings to create ar force. Apex, and the Web ! #: AIV Scette New View	d manage custom data at t Services API.	the organization, profile, and use	or levels. Custom settings data is stored in th	e application cache. This means you can acc	ass if efficiently, without the cast of repeated queries. Custom settings	data can be used by formula fields
stom Settings actom settings to create an force. Apex, and the Web ! w: (AB ) Scente.New Web !	d manage custom data at t Services API.	the organization, profile, and use	r lovels. Custom settings data is stored in th	e application cache. This means you can acc Get Usage	as it efficiently, without the cost of repeated queries. Custom settings $\mathbf{x} \mid \mathbf{z} \mid \mathbf{C} \mid \mathbf{D} \mid \mathbf{z} \mid \mathbf{r} \mid \mathbf{C} \mid \mathbf{H} \mid 1 \mid \mathbf{J} \mid \mathbf{K} \mid \mathbf{L} \mid \mathbf{M} \mid \mathbf{N} \mid \mathbf{O} \mid \mathbf{P} \mid$	data can be used by formula fields $a \mid \pi \mid s \mid \tau \mid u \mid v \mid w \mid x \mid v \mid z \mid o$
stom Settings ustom settings to create an force. Apex, and the Web 1 W: Ale Coaste Jacober B: Label 1	d manage custom data at i Bervices API. Vietesty	the organization, profile, and use	r lovels. Custom settings data is stored in th	e application cache. This means you can acc Ger Urage New Description	as if efficiently, without the cast of repeated queries. Curtom settings $A \mid B \mid C \mid D \mid C \mid P \mid C \mid H \mid i \mid J \mid K \mid L \mid M \mid N \mid O \mid P \mid$ Record See. Number of Records	c data can be used by formula fields □   R   S   T   U   V   W   X   Y   Z   O Total Size

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Click "New". Be sure to select the button above "Default Organization Level Value", not the one below it. The "New" button in the grid below will create a setting that is only applicable to a subset of users, but the org level value will apply to all users. Scoped values can be used if desired, but be sure to at least define an org level value.

Custom Setting InRule	Heap for this Page 🥹
If the custom setting is a list, click New to add a new set of data. For example, if your application had a setting for	country codes, each set might include the country's name and dialing code.
If the custom setting is a hierarchy, you can add data for the user, profile, or organization level. For example, you n	nay wani different values to display depending on whother a specific user is running the app, a specific profile, or just a general user
Edit Delese	
▼ Default Organization Level Value	
Location InRule	AppOemain Caobers 0
Log Levels 0	Rula App Name (5 SalestonocRules
View: All Casada New Man	
	A B C D E F B H I J K L M N D F Q F S T U V W X Y Z Other MI
	New
Setup Owner 1	Location
No records to display.	

Edit InRule			Save
InRule Information			
	Location AppDomain Cache () Log Level () Rule App Name ()	InRule 3600 3 SalesforceRules	

Configuration Form Fields	
Rule App Name	The name of the rule app that will be loaded by the Rule Execution Service when running rules
App Domain Cache	An integer that defines the amount of time in seconds that the InRule RuleHelper will hold entities and collections in cache after querying for them. You can use the "Clear Cache" button on the InRule Configuration page to clear the cache at any time. Additionally, re-setting the configuration value to 0 will clear the cache.
Logging Level	An integer that denotes the amount of information to log after rule execution completes. This is a default value that will be used by all invocations of the DecisionClient, but can be overridden in the calling script. Results are written to the InRule_Logc object. Values can be 0, 1, 2, or 3. *These log levels are different than the log levels in the execution service which must be configured independently*:
	<ul> <li>0 – disable all logging</li> <li>1 – Logs errors and a minimal amount of information on successful requests</li> <li>2 – Logs errors, rule engine notifications, and rule engine validations</li> <li>3 – Logs the same information as 2, but also includes JSON from the HTTP request and response payloads, and additional trace</li> </ul>

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Once configured, press Save.

## 4: Verify a successful deployment

Now that all of our resources are properly deployed and configured, we can test to ensure they're all working as they should be. Navigate back to the InRule App and then the Configuration tab once again. Find the "Test Connectivity" button and press it. This button will make a mock request out to your Rule Execution Service to verify that the InRule App and all related Azure resources are deployed and configured properly.

InRule Getting Started Configuration Running Rules InRule Logs 🗸	
<u> </u>	#QQQ///A(`\##!
Configuring the InRule App	INRULE
Once the dependencies outlined in the Getting Started section are successfully deployed, the InRule App can now be configured within Salesforce. When th configuration is complete, test the Rule Execution Service connection by clicking the following button:	he Connected App and Named Credential
Test Connectivity Configuration includes the following three steps:	
Setup a Connected App	
Establish the framework for the Rule Execution Service to communicate with Salesforce	
> Connected App Details	
Setup Named Credential	
Establish the credentials for Salesforce to authenticate with the Rule Execution Service	
> Named Credential Details	
> Named Credential Access Details	
Setup Custom Settings	
Manage Rule App Name, logging, and caching configurations	
> Custom Settings Details	

If successful, you should get 3 green notifications like below:



# 5: Add the Run Rules button

As a part of the deployment of the InRule for Salesforce App, a new "Run Rules" Lightning component has been installed. To add and configure this component to an entity or entities, you can follow the steps below. This example uses the Account entity and the rule app you uploaded in the <u>Testing the Catalog</u> App step, but this process should work for any entity, default or custom.

Alternatively, a sample Run Rules JavaScript button has now been installed for the account and contact entities in the Classic (non-Lightning) UI. For guidance on how to add a Run Rules button in the Classic UI, refer to <u>Appendix E: Methods for Executing Rules from Salesforce.</u>

To begin, navigate to the entity you wish to add the button to. Note that you will have to add the button to each entity individually.

Once on the desired entity page, find the settings icon in the top right corner and select Edit Page.

	* 🖬 ? 🌣 🐥 🐻
(///	Setup Developer Console
•	Edit Page
	Edit Object Edit Page

On the left-hand side of the page in the Lightning Components section scroll to the bottom of the page until you find the InRule_RunRules in the Custom-Managed Section

### Custom (0)

No components available.



InRule RunRules

**Note:** If you have not setup your Salesforce domain, you will see this message in the Custom-Managed Section.



components here.

Simply follow the link to setup your domain. Once it has been set and you have logged in with the new domain come back to edit page and you should now see the InRule_RunRules Lightning Component.

Drag the InRule_RunRules Lightning component to the desired location on the page view. Once you have placed the component, click on it and a menu bar on the right will appear with two available configuration values.

#### Page > InRule_RunRules

Rule Application Name - leave blank to use the default configured rule application

RuleSet Name(s) - comma-delimited list or leave blank to derive RuleSet as (entityName)DefaultRules

The first configuration field, Rule Application Name, allows you to define a Rule Application to use for this specific lightning component. Placing a value in this field will override the Rule Application you have defined in your Custom Setting as the default. Leave this field blank to use the default configured Rule Application.

The "Rule Set Name(s)" field accepts a comma delimited list of Rule Set names. Adding multiple rule sets here will create multiple Run Rules buttons for each defined Rule Set on the page. Leaving this field blank will leave only the singular "Run Rules" button, and this button will execute the Default Rule Set for the given entity. The Default Rule Set is defined as the entity's name + "DefaultRules." For example, if you are adding the Lightning component to the Account entity, the Rule Set name it will default to if this field is left blank is "AccountDefaultRules."

If no Rule Sets are defined, it will create 1 button that uses the default Rule Set for the entity. This button will be generically titled "Run Rules":

+ Follow	Edit	New Contact	New Case	•	RuleSet Name(s) - comma-delimited list or leave blank to derive RuleSet as (entityName)DefaultRules
InRule Pup Pulos		0		<b>⊕ 亩</b>	<ul> <li>Set Component Visibility</li> <li>Filters</li> <li>+ Add Filter</li> </ul>

If one Rule Set is defined, will create 1 button that will use that defined Rule Set and will be titled with that Rule Set's name:

InRule Rule 1		07/72	Υ <i>214</i>	11.15	617
Activity	Chatter				

If 2+ Rule Sets are defined, will create a button per Rule Set that will use the respective Rule Sets, each button titled with the rule set they map to:



Once you have configured your Rule Sets, click save in the top right corner. It will prompt you to activate this page to make it visible to your users, click activate.

	Activity	×
	Page Saved	
is		.0
-	Activate this page to make it visible to your users.	
	Activate the page now, or do it later using the Activation button in the App Builder toolbar.	
	Don't show me this message again Not Yet Activate	
T	New No next step	os. To get

Salesforce will now ask for the scope to activate the record page. Select the desired scope and click Assign and then save on the subsequent prompt.

Custom reco	rd pages can be assigned at different levels:
S The org de	fault record page displays for an object unless more specific assignments are made.
S Apr	default page assignment, if specified, overrides the org default.
	App, record type, profile assignments override org and app defaults.
Learn more abo	ut Lightning page assignment.
ORG DEFAUL	APP DEFAULT APP, RECORD TYPE, AND PROFILE
et this page as the efined.	s org default to display it for all Account records, except when app default of app, record type, or prome-specific assignments an
In standard default pag page as the	Salesforce console apps, some objects have a system app default record page. For those objects, if you assign a custom org e, it doesn't display to users. To enable a custom org default page to show up in the console for those objects, assign a custom app default. Check your assignments.
Assign as Org D	efault

Navigate back to the main entity page and the component should appear.

	大王 田 ? 卒事 🐻
Account       Test       Type       Phone       Website       Account Site       Industry	+ Follow New Contact New Case New Note •
Related Details News	InRule
We found no potential duplicates of this account.      Contacts (0)      New	Activity Chatter
Opportunities (0)	New Task Log a Call New Event Email Ovade a task
Cases (0) New	Filters: All time - All activities - All types 🔻 Berlieth Expend All
ی upread Files Or drop Files	No not story: To get things moving, add a task or set up a meeting. Past Activities No past activities No past activities
	Tract Wars Fact Articities 🔹
# Appendix A: Additional Resources

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

#### InRule's Support Website

InRule's support website can be found at <u>http://support.inrule.com</u>. 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 <u>support@inrule.com</u>.

#### 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 <u>support@inrule.com</u>.

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 <u>ROADServices@InRule.com</u>

#### 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 <u>Training@InRule.com</u>.

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

The steps below help to give a top-level understanding of how InRule is integrated with Salesforce. Please note these steps are 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. The DecisionClient in Salesforce is called by a button or event. This generates a call to the Rule Execution App Service.
- 2. The Rule Execution App Service makes a request to the Catalog Service, asking for a copy of the requested RuleApp.
- 3. The Catalog Service Queries its SQL Server based database for a copy of the requested RuleApp.
- 4. The SQL Server responds with the RuleApp.
- 5. The catalog service responds to the Rule Execution App Service with the RuleApp.
- 6. The RuleApp executes inside the Rule Execution App Service.
- 7. Optionally, the RuleApp has an opportunity to query Salesforce for additional data needed to execute rules.
- 8. The RuleApp completed execution
- 9. The Rule Execution App Service relays the response to Salesforce, where the InRule for Salesforce App synchronizes changes

# Appendix C: irX General Integration Concepts

### **Runtime Mapping across Nested Relationships**

Much like Salesforce, 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 Salesforce Entities to InRule Entities, and vice versa.

# Controlling irVerify Behavior with Load, Save and Inactive Record Settings

When working with an Entity Schema composed of many related Salesforce 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 two buttons that give the rule author the control to denote if a relationship should be automatically loaded or saved excluded.



**Note:** Automatic loading and saving is enabled by default for all relationships that are imported from Salesforce. 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.

# Appendix D: Accessing Salesforce 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. InRule for Salesforce 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 Salesforce 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 Salesforce
- The rules need to compare many combinations of unrelated Salesforce Entities and produce results about best possible matches or scores
- A custom filter is required when loading data for 1:N relationships

# Working with Disconnected Fields when Loading and Saving Data

One of the most important integration concepts when loading Salesforce 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 Salesforce.

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 Salesforce.





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

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

FILE	HOME	CATALOG J	AVASCRIPT	DYNAN	AICS SALESFO	DEVE	LOPER	
D Manage	Disconnect Item	🔁 Auto Load	Test Re	egression Test	Update Regression Test	Configure	License Information	<b>?</b> Help
Schema	ltem		Testing		Environments	Help		

**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 Salesforce. 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 Salesforce.

FILE	HOME	CATALOG J	AVASCRIPT	DYNA	MICS SALESFO	DEVEL	OPER	
Manage	Disconnect Item	🔒 Auto Load	Test	Regression Test	Update Regression Test	Configure	License Information	<b>?</b> Help
Schema	Item			Testir	ng	Environments	Help	

# **Integrating the Rule Helper Component**

InRule provides a sample rule application (SalesforceRules) that is already configured for RuleHelper usage. You can simply edit this rule app, or, if you wish to integrate the Rule Helper into an existing rule app, you can copy both the UDF Library "RuleHelper" and End Point 'SalesforceHelper" from the SalesforceRules 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.Salesforce.RuleHelper.dll assembly. Select the SalesforceDriver class and then select the methods that should be callable from rules. Edit the name of the end point to "SalesforceLib" or similar. Select the methods from the SalesforceDriver that are needed for the Rule Application. You do not need to select all the methods—only import the methods that will actually be used by rules. Additional methods can always be imported later be revisiting the endpoint screen and reloading the assembly.

Add - 🕺	c	SalesforceLik	)					
SalesforceLib	NFT Assemt	NFT Assembly Function Library						
	Assembly	InRule Salesf	orce RuleHelper	eload				
	Classes	marcibarcs	oreander leiper	cioad				
	Classes							
	Filter:		-					
	Show all:	✓ Classes ✓	Methods					
	Include	Full Name		Alias	Use Intrin	sic		
		nRule.Salesforce.RuleHelper.B	Environment.SalesforceConnector	Salesforc	eConnector			
		nRule.Salesforce.RuleHelper.B	Environment.SettingsManager	Settings	Nanager	1		
		nRule.Salesforce.RuleHelper.(	OrderBy	ClauseBuilder	1			
		nRule.Salesforce.RuleHelper.S	GalesforceDriver	Salesforc	eDriver			
Rule Flows		nRule.Salesforce.RuleHelper.\	WhereClauseBuilder	WhereCl	auseBuilder	1		
Rules	Class me	ethods						
Vocabulary	Include	Name	Туре	Static	Parameters			
Entities	E	Equals	System.Boolean		Object obj	~		
Linuues		GetHashCode	System.Int32					
User Defined Functions		GetType	System.Type					
Data		LoadMappedChildCollection	System.Void	~	IRuleExecutionContext context, E	int		
End Points		LoadMappedChildCollection	System.Void	$\checkmark$	IRuleExecutionContext context, E	int		
		LoadMappedChildEntity	System.Void	$\checkmark$	IRuleExecutionContext context, E	int 🗸		

- 3. Add a User Defined Function library and set the name to "SalesforceHelpers" or similar. This library will contain functions that the rules will call to query Salesforce.
- 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 SalesforceDriver. Fill out the UDF with script that will call a method on the SalesforceDriver.

**Note:** The methods on the SalesforceDriver 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 Salesforce 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 a reference to the Entity against which the current Rule Set is executing. The Context and its child properties are passed to the SalesforceDriver so it has enough information to form calls to Salesforce 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.SalesforceDriver.QueryCollection(Context, Context.Entity, collectionName, filter, orderBy, connectionString);

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



Rules	A LoadContacts
IoadCrmData     IoadContacts     Independent Rule Sets	RuleHelper.QueryCollection(CollectionName = "Contacts", Filter = concatenate "AccountId = ", RuleHelper.GetAccountId(account = this Account), "", [values], OrderBy = "")

Important: The Target Collection in the sample rule is called "FamilyMembers". This is a Field that either does not exist in Salesforce (only for use in rules), or has been imported and then "disconnected" from Salesforce 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 in this example.

# Filtering Queries using the Where Clause Builder

When loading data from Salesforce 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 SalesforceDriver class.

During execution of the SalesforceDriver, the filter clause is parsed into an Abstract Syntax Tree (AST) and then translated into SOQL (Salesforce Object Query Language) so it can be executed against the Salesforce data. 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 SalesforceDriver 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:

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.



- All the Field names that are used are the Salesforce friendly names (Salesforce Field Label) that are used in the Rule Application. These names are mapped back to the Salesforce 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 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 and fields** queried in the relationship chain be imported into irAuthor.

Rules	
🕂 Add Rule Set 👻 🖅 Collapse All 👻	A Call_QueryCollection
⊿ 🧀 Account	Language Rule
AccountDefaultRules	
∡ → RuleHelperQueryRules	RuleHelper.QueryCollection(CollectionName = "QueriedContacts", Filter = "Cases.WebName = 'Rogers'")
QueryAllContacts	
QueryFilteredContacts	
▲ Call_QueryCollection	

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 a Web Name of "Rogers." 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:

Rules	
🕂 Add Rule Set 👻 📴 Collapse All 🔹	A Call_QueryCollection
🔺 🚔 Account	Language Rule
AccountDefaultRules	
▲	RuleHelper.QueryCollection(CollectionName = "QueriedAccounts", Filter = "Contacts.Cases.WebName = 'Rogers'")
QueryAllContacts	
▲ QueryFilteredContacts	
Call_QueryCollection	
· · · · · · · · · · · · · · · · · · ·	1

This example would return Accounts that have related Contacts with Cases that have a Web Name of "Rogers." The filter expression can support querying in this manner up to 5 "layers" deep, not including the initially queried entity. Put another way, you can have up to 6 total different related entities in a single filter expression.

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

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."

Rules	
+ Add Rule Set 👻 🖂 Collapse All 🔹	A Call_QueryCollection
🔺 📻 Account	Language Rule
AccountDefaultRules	
a 📻 RuleHelperQueryRules	RuleHelper.QueryCollection(CollectionName = "QueriedContacts", Filter = "Cases.WebName = 'Rogers'" & "OriginatingLead.Name = 'Jones'")
QueryAllContacts	
QueryFilteredContacts	
Call_QueryCollection	

# Ordering Query Results with the OrderByClauseBuilder

The SalesforceDriver class also supports the ability to control the order of the results returned from Salesforce 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:

Rules	
🕈 Add Rule Set 👻 🕮 Collapse All 👻	A LoadContacts
Account  CodCrmData  CodContacts  Independent Rule Sets	Language Rule RuleHelper.QueryCollection(CollectionName = "Contacts", Filter = concatenate "AccountId = "", RuleHelper.GetAccountId(account = this Account), """, [values], OrderBy = "FullName")

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

Rules + Add Rule Set * = Collapse All *	LoadContacts
LoadCrmData      LoadContacts      Independent Rule Sets	Language Kule         RuleHelper.QueryCollection(CollectionName = "Contacts", Filter = concatenate "AccountId = '", RuleHelper.GetAccountId(account = this Account), "", [values], OrderBy = "FullName DESC")



*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.

# Methods Available in the Rule Helper

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

Method Name	Description
LoadMappedChildCollection	Populates a child Entity Collection based on an existing 1:N
	relationship in Salesforce. The Collection is populated based on
	existing parent-child relationship data in Salesforce.
LoadMappedChildEntity	Populates a child Entity Field based on an existing 1:1 relationship in
	Salesforce. The Field is populated based on existing parent-child
	relationship data in Salesforce.
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 Salesforce.
QueryEntity	Populates an Entity Field or variable based on a query to Salesforce.
	An optional filter clause (where clause) can be used to define

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Method Name	Description
	selection criteria for the Entity. The Field does not need to correspond to a 1:1 relationship in Salesforce. If more than one Entity is returned from the query to Salesforce, then the first Entity in the set is used.
QueryField	Populates a primitive Field or variable based on a query to Salesforce. 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 Salesforce, then the Field value from the first Entity in the matching set is used.

# 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 Salesforce. 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 Salesforce Entities should be reused from the InRule entity cache, or if new entity instances should be created. If false, the original entity data will be requested from Salesforce, and a copy of the Entity is created. The Instance ID is not set to the ID of the Salesforce Entity, which will also prevent changes to this entity from being written back to Salesforce. This functionality can, for example, be used to load the original values for an entity persisted in Salesforce when rules are run on update and compare the original and updated values. The default value is true.
overwriteIfLoaded	Denotes if a previously loaded Salesforce Entity should be repopulated with the latest values in Salesforce. This behavior will overwrite Field values stored in the cache. The default value is false.
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.



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

# Using the Rule Helper with the Native REST Execution Service

While using the Salesforce rule execution service documented in this guide is the suggested way to interact with Salesforce via rules, you can also use the rule helper from the native REST execution service, which does not connect to Salesforce out of the box.

To do this, you will first need to copy the Salesforce rule helper assemblies to the bin directory of your execution service. These assemblies can be found in the `RuleHelperDeployment` folder of the framework zip you download from the support site. The method for copying these assemblies will differ

based on your exact hosting setup, but if your execution service is hosted in Azure you can copy these files over via the App Service Editor or FTP.

Once you have copied the assemblies to the bin directory, you will need to add the required app settings for authenticating to Salesforce. These are the same parameters provided in the <u>Update parameters</u> step of the template deployment, but you will need to provide the exact app setting names instead of the template parameter names. These are the settings you will need to provide, along with their mapping to the template parameters from the section linked above:

App Setting Name	Template Parameter Name
Inrule:sf:api:loginUrl	sfLoginUrl
inrule:sf:api:username	sfUsername
inrule:sf:api:password	sfPassword
inrule:sf:api:securityToken	sfSecurityToken
inrule:sf:api:consumerKey	sfConsumerKey
inrule:sf:api:consumerSecret	sfConsumerSecret

Once you have set these app settings, you should be able to use the rule helper to manually load and save data to Salesforce from rules, just like you would from the Salesforce rule execution service.

# Appendix E: Methods for Executing Rules with Salesforce

Once the Salesforce components are installed, end-to-end connectivity can be tested by adding a call into a Page Layout. Many possible approaches can be applied to call the DecisionClient Apex class—six approaches are documented here, adding a button to the Lightning interface, adding a JavaScript button to the Classic UI, executing rules from Apex, executing rules from Apex triggers, executing rules from Salesforce Flow, and executing rules from the Salesforce REST Endpoint

### 1 Adding a Lightning Button

Navigate to the entity you wish to add the button to, note that you will have to add the button to each entity individually.

Once on the desired entity page find the settings icon in the top right corner and select Edit Page.

	*• 🖬 ? 🌣 🐥 🐻
- / //	Setup Developer Console
:	Edit Page
	Edit Object Edit Page

On the left-hand side of the page in the Lightning Components section scroll to the bottom of the page until you find the InRule_RunRules in the Custom-Managed Section



If you have not setup your Salesforce domain, you will see this message in the Custom-Managed Section.



Copyright© 2024 by InRule Technology, Inc. All Rights Reserved. CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent. Simply follow the link to setup your domain. Once it has been set and you have logged in with the new domain come back to edit page and you should now see the InRule_RunRules Lightning Component.

Drag the window to the desired location on the page view. Once you have placed the component, click on it and a menu bar on the right will appear three two available configuration values.

Rule Application Name - leave blank to use the default configured rule application

RuleSet Name(s) - comma-delimited list or leave blank to derive RuleSet as (entityName)DefaultRules

Rule Application Label(s) - comma-delimited list of Rule Application Labels. If populated with more than 1 label, will display a dropdown menu for user selection of label. If populated with 1 item, will hide dropdown and automatically use that label. If left blank, rule execution will use the label configured on the Rule Execution Service.

The first configuration field, Rule Application Name, allows you to define a Rule Application to use for this specific lightning component. Placing a value in this field will override the Rule Application you have defined in your Custom Setting as the default. Leave this field blank to use the default configured Rule Application.

The "Rule Set Name(s)" field accepts a comma delimited list of Rule Set names. Adding multiple rule sets here will create multiple Run Rules buttons for each defined Rule Set on the page. Leaving this field blank will leave only the singular "Run Rules" button, and this button will execute the Default Rule Set for the given entity. The Default Rule Set is defined as the entity's name + "DefaultRules." For example, if you are adding the Lightning component to the Account entity, the Rule Set name it will default to if this field is left blank is "AccountDefaultRules."

If no Rule Sets are defined, 1 button will be displayed that uses the default Rule Set for the entity:

+ Follow	Edit New Contact	New Case Rul lea	eSet Name(s) - comma-delimited list or ve blank to derive RuleSet as ıtityName)DefaultRules
InRule Run Rules	0	<b>+ </b> [⊕] Fi	Set Component Visibility Iters + Add Filter

If one Rule Set is defined, 1 button will be displayed that will use that defined Rule Set:

+ Follow	Edit	New Contact	New Case	Ŧ	RuleSet Name(s) - comma-delimited list or leave blank to derive RuleSet as (entityName)DefaultRules
					Rule1
	1110				✓ Set Component Visibility
nRule		·	1	<b>⊕</b>	Filters
Run Rules					+ Add Filter

If 2+ Rule Sets are defined, a button per Rule Set will be displayed that will use the defined Rule Sets.

+ Follow	Edit	New Contact	New Case	•	RuleSet Name(s) - comma-delimited list or leave blank to derive RuleSet as (entityName)DefaultRules
					Rule1, Rule2
		•			✓ Set Component Visibility
InRule			1	+ <b>•</b>	Filters
Rule1 Rule2		0			+ Add Filter

The last configuration field accepts a comma-delimited list of Rule Application Labels. This allows for overriding the Rule Application Label defined on the Rule Execution Service and providing users the ability to change what label to run rules against from the defined list of options.

When no Rule Application Labels are defined, the Run Rules button appears as normal and rules will be executed against the label configured on the Rule Execution Service:



If one Rule Application Label is defined, the Run Rules button appears as normal and rules will be executed against that defined label:

	Rule Application Label(s) - comma-delimited
	list of Rule Application Labels. If populated
	with more than 1 label, will display a dropdown
	menu for user selection of label. If populated
	with 1 item, will hide dropdown and
	automatically use that label. If left blank, rule
0	execution will use the label configured on the
InRule 🕂 🖷	Rule Execution Service.
Run Rules	Label1
0	

If more than one label is defined, a dropdown box will appear underneath the Run Rules button that allows users to select what label to run rules against. All defined Rule Sets and their associated buttons will execute against the selected label:

Rule Application Label(s) - comma-delimited

	list of Rule Application Labels. If populated with more than 1 label, will display a dropdown
TinRule 🕂 🗇 🏦	with 1 item, will hide dropdown and automatically use that label. If left blank, rule execution will use the label configured on the Dule Execution Service
*Rule Application Label Label1	Label1, Label2

Once you have configured these fields, click save in the top right corner. It will prompt you to activate this page to make it visible to your users, click activate.

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved. CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent.

			Activity	×
	Page Saved			
is.				.0
Ad	ctivate this page to make it visible to your users.	-		
Ac Ap	tivate the page now, or do it later using the Activation button in the p Builder toolbar.	_	~	
Dor	't show me this message again	Not Yet	Activate	
1	New	No	next steps.	To get

Salesforce will now ask for the scope to activate the record page. Select the desired scope and click Assign and then save on the subsequent prompt.

Custom record pa	iges can be assigned	at different levels:
3 The org default	record page displays for a	an object unless more specific assignments are made.
👆 🗲 App defa	ult page assignment, if sp	pecified, overrides the org default.
	o, record type, profile as	signments override org and app defaults.
Learn more about Lig	htning page assignment.	
ORG DEFAULT	APP DEFAULT	APP, RECORD TYPE, AND PROFILE
et this page as the org o efined.	default to display it for all	Account records, except when app default or app, record type, or profile-specific assignments are
<ul> <li>In standard Sales default page, it d page as the app of</li> </ul>	force console apps, som oesn't display to users. To default. Check your assig	e objects have a system app default record page. For those objects, if you assign a custom org o enable a custom org default page to show up in the console for those objects, assign a custom ments.
Assign as Org Default		

Navigate back to the main entity page and the component should appear.

All v Q, Search Salesdoxce	🖈 🖬 ? 🌣 🐥 👸
Account Account Account Account Ste Industry	+ Fallow New Contact New Case New Note ¥
Related Details News	InRule
We found no potential duplicates of this account.      Contacts (0)      Kerr	Activity Chatter
C Opportunities (0) New	New Task Log a Call New Event Email
Cases (0)	Filters All time - All activities - All types 🔻 Refersh Expend All
Proces & Assectments to)	Next Steps User lifes No next steps. To get trings moving, add a task or set up a meeting. Past Activities No past activity Past meetings and tasks marked as done show up here.
	Inser thus Real Articles 🔍

### 2 Adding a Classic UI Button

Classic UI buttons are not included directly in the InRule for Salesforce app, but we do provide a helper library and sample code here to make it easier to add a button to a classic form for running rules. This button will run rules, show notifications, and refresh field changes when clicked.

To add a Classic UI button, navigate to Setup  $\rightarrow$  Customize or Create, add a new Button. Set the Behavior to Execute JavaScript and then the Content Source to OnClick JavaScript.

Custom Button or Link Edit	Save	Quick Save Prev	iew Cancel
Label	Run Rules		
Name	Run_Rules		
Description			//
Display Type	Detail Page Link <u>View example</u>		
	Detail Page Button <u>View example</u>		
	List Button <u>View example</u>		
Behavior	Execute JavaScript	•	View Behavior Options
Content Source	OnClick JavaScript ▼		

Within the script window, copy and paste over the sample JavaScript found below:

```
{!REQUIRESCRIPT("/soap/ajax/33.0/connection.js")}
{!REQUIRESCRIPT("/soap/ajax/33.0/apex.js")}
{!REQUIRESCRIPT('/resource/' & LEFT(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(TEXT(NOW()),':',''),'
',''),10) & '000/inrule_DecisionClientHelper')}
var config = {
entityId : '{!Account.Id}',
```

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

```
entityType : "Account",
ruleSetName : "AccountDefaultRules"
};
var response = executeRules(config);
displayNotifications(response);
```

window.location.reload();

Your script window should now look similar to this:

```
{!REQUIRESCRIPT("/soap/ajax/33.0/connection.js")}
{!REQUIRESCRIPT("/soap/ajax/33.0/apex.js")}
{!REQUIRESCRIPT('/resource/' &
LEFT(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(TEXT(NOW()),':',''),'',''),10) &
'000/inrule_DecisionClientHelper')}
var config = {
entityId : '{!Account.Id}',
entityType : "Account",
ruleSetName : "AccountDefaultRules"
};
var response = executeRules(config);
displayNotifications(response);
window.location.reload();
```

To customize this button for the specific entity type you intend to use this button for, you can alter the "config" object in the JavaScript by editing the values of the properties to match the particulars you want.

The table below includes a list of the properties that can be configured on the "config" object:

Parameter Name	Description
entityId	The unique Salesforce identifier for the root object in the request. This follows the naming convention of: "entityName.Id"
objectType	The Salesforce object type of the root object in the request
ruleSetName (optional)	Optional. The name of an Explicit Rule Set to call as the entry point for rule execution. If no value is supplied, then the Rule Sets configured as 'Auto' for that entity will be executed.
loggingLevel (optional, must be manually added to config object)	<ul> <li>Optional. An integer that denotes the amount of information to log after rule execution completes. Results are written to the lnRule_Logc object. Values can be 0, 1, 2, or 3:</li> <li>0 – disable all logging</li> <li>1 – Logs errors and a minimal amount of information on successful requests</li> <li>2 – Logs errors, request information, rule engine notifications, and rule engine validations</li> </ul>
	<ul> <li>3 – Logs the same information as 2, but also includes JSON from the HTTP request and response payloads</li> </ul>

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

	Defining a logging level here will override the logging level universally defined for the InRule for Salesforce App in the custom object settings for a specific button.
useEntityPrefix (optional, must be manually added to config object)	Optional. If set to true, the DecisionClient will append the entity label to the supplied rule set name.
ruleAppName (optional, must be manually added to config object)	Optional. Defining and passing a RuleAppName here allows you to override the default Rule App Name defined in your Custom Setting created during initial configuration for this specific button.

Once your config object has been appropriately configured, simply save your button and add it to the entity page layout for use. The button will leverage the DecisionClientHelper static resource to handle the communication between the JavaScript button and the DecisionClient, as well as handling any notifications returned from the rules.

To verify everything is setup correctly, open up a record for the configured object and click the 'Run Rules' button you added. What will happen next is dependent on your configuring rule set, but if you are running the included 'AccountDefaultRules', you should see the description of the account updated with the current date and time.

« Back to List: Remote S	Site Settings		
Contacts [0]	Opportunities [0]   Cases [0]   Op	an Activities [0]   Activity History [0]	Notes & Attachments [0]   Partners [0]
Account Detail	Edit Delete	Include Offline Run Rules	
Account Owner	Liser User [Change]	Rating	
Account Name	Test Account [View Hierarchy]	Phone	
Parent Account		Fax	
Account Number		Website	
Account Site		Ticker Symbol	
Туре	Account	Ownership	
Industry		Employees	
Annual Revenue		SIC Code	
Billing Address		Shipping Address	
Created By	User User, 4/29/2019 2:21 PM	Last Modified By	User User, 4/29/2019 2:21 PM
Description	Updated via rules on 4/29/2019 4	:21:43 PM	
	Edit Delete	Include Offline Run Rules	
Contacts	New Contact	Merge Contacts	Contacts He
No records to display			

# 3 Executing Rules from Apex or Lightning Web Components

Rule Execution is handled on the Salesforce side by an Apex class installed with the InRule for Salesforce App called the DecisionClient. The DecisionClient is what accepts the run rules requests from the Run Rules button and trigger requests to handle sending it along to the Rule Execution Service, but it also accepts calls from other Apex classes or Lightning Web Components (LWCs). This section will document how to run rules from your own Apex classes or LWCs by calling the DecisionClient, as well as detailing what manner of response it returns.

To run rules from another Apex class, simply invoke the executeRules method in the DecisionClient with the following call:

# inrule.DecisionClient.executeRules(String eventType, String id, String objectType, String ruleSetName, Boolean useEntityPrefix, String ruleAppName, String ruleAppLabel, string entityImage, Boolean persistChanges)

Alternatively, setting up a call to the DecisionClient from a LWC requires setting up an action. First, you must reference the DecisionClient as your LWC's controller in your .cmp file:

#### <aura:component controller="inrule.DecisionClient">

To create the run rules action to hit the DecisionClient in your component controller, new up your action with by setting it as below:

#### var action = component.get('c.executeRules');

From there, the action can be setup and enqueued as with any other action, with the following parameters to set:

action.setParams({ eventType: string id: string objectType: string ruleSetName: string, useEntityPrefix: boolean, ruleAppName: string, ruleAppLabel: string, entityImage: string, persistChanges: boolean });

Regardless of whether you are calling from Apex or a LWC, the arguments will need to be defined with the following values:

Parameter Name	Description
Event Type (String)	The event type for invoking rules. This will always need to be either "insert," "update" or "delete," depending on what your rule is doing
entityId (String)	The unique Salesforce identifier for the root object in the request. This is a property available on all Salesforce objects and be accessed with: "entityName.Id"
objectType (String)	The Salesforce object type of the root object in the request. For example, if running rules against an Account entity, this will need to be set to a string value of "Account".
ruleSetName (String)	The name of an explicit Rule Set to call as the entry point for rule execution.
useEntityPrefix (Boolean)	If set to true, the DecisionClient will append the entity label to the supplied rule set name. For example, if you pass in a ruleSetName of "DefaultRules" and set useEntityPrefix to true, the effective ruleSetName name would be "AccountDefaultRules"
ruleAppName (String) (Optional)	Optional. Defining and passing a RuleAppName here allows you to override the default Rule App Name defined in your Custom Setting created during initial configuration for this specific button. If you do not wish to override your Custom Setting, pass null here.
ruleAppLabel (String) (Optional)	Optional. Defining and passing a rule app label here allows you to override the rule app label configured on the execution service. If you do not wish to override your execution service's configuration, pass null here.
entityImage (String) (Optional)	Optional. This allows you to pass in the serialized JSON string of an entity image as it exists at the time of calling the DecisionClient, which will have rules execute against the entity image passed in, not the entity image as it exists in Salesforce when the execution process reaches the execution service. If you do not wish to pass this in, pass null instead.
persistChanges (Boolean) (Optional)	Optional. This allows you to define whether to persist data changes made during rule execution to Salesforce. Pass in null to default this value to true.

The DecisionClient will always return a JSON string; a serialized version of the DecisionClientResponse object. Once you have received it back as a string, you will need to deserialize it to access its properties.

Property Name	Description
IsSuccess (Boolean)	Denotes whether or not rules successfully ran with no errors.
Notifications (List <notificationmessage>)</notificationmessage>	Provides a list of all notification messages returned by rule execution. Each NotifactionMessage has 2 properties on it:
	• <b>Type (Integer):</b> The notification type is an integer that maps to a Salesforce notification type. 0 maps to Informational, 1 to Success, 2 to Warning, and 3 to Error.
	<ul> <li>Message (String): The notification text</li> </ul>
Errors (List <errormessage>)</errormessage>	Provides a list of all errors encountered during rule execution. These differ from errors thrown by the rule application itself, which are instead added to the Notifications list as NotificationMessages of type Error. Errors added into the Errors list are strictly errors encountered during rule execution runtime.

The available properties on the DecisionClientResponse are:

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Each ErrorMessage has 2 properties:
<ul> <li>Source (String): At what point during runtime the error was</li> </ul>
thrown
Message (String): The error text

# 4 Executing Rules from Apex Triggers

As of v5.5.1, the InRule for Salesforce App supports the execution of rules from **Update** triggers with some limitations. For a full list of these limitations, references the <u>Known Issues and Limitations of</u> <u>Executing Rules from Triggers</u> section below.

To emphasize, only **After Update** and **After Insert** triggers are supported. Due to Salesforce's enforcement of all external service methods running asynchronously, trigger execution and persistence to the Salesforce database will always complete before rule execution has had time to finish, thus preventing the ability to execute rules on the "in-progress" entity image before it has been persisted. As an extension of this limitation, no entity image rollbacks can occur as a result of any validation done during or after the rule execution process.

As an alternative to Apex triggers, <u>Salesforce Flows</u> are a no-code alternative that can also be used to call InRule. For an overview of the differences between these two options, refer to the following Salesforce documentation: <u>Record-Triggered Automation</u>

#### Adding a Rule Execution Trigger

To define a rule-executing trigger, create a new trigger on the entity of choice. The following example will use Account:

# Apex Trigger

#### Apex Trigger Edit

Apex Trigger	Version Settings	
Is Active 🗹		
Q 🔺	🄄 🎓 🗛 А	
1 trigg	ger example on Account (after update)	
3		
ц.,р		

Next, all rule executing triggers must contain the boilerplate wrapper around their code defined below. This is required to prevent infinite loops that may occur if the invoked rule edits the same entity.

# Apex Trigger

#### Apex Trigger Edit

Is Active A A	Apex Trig	gger Version Settings
Q → A A 1 trigger example on Account (after update) 2 {     if (!System.isFuture())     4     {         5	Is Active	
<pre>1 trigger example on Account (after update) 2 { 3     if (!System.isFuture()) 4     { 5         //Trigger code goes here 6     } 7     else</pre>	Q.	🔸   🦣 🎓   A 🗛
<pre>2 { 3 if (!System.isFuture()) 4 { 5</pre>	1	trigger example on Account (after update)
<pre>5 11 (:System:Isruture()) 4 { 5      //Trigger code goes here 6 } 7 else</pre>	2	( if (ISustam isEutupa())
<pre>5 //Trigger code goes here 6 } 7 else</pre>	4	{
6 } 7 else	5	//Trigger code goes here
7 else	6	}
	7	else r
9 return:	9	i return:
10 }	10	}
11 }	11	}

All operations, including calls to the rule execution service, will be contained within the **If** (!System.isFuture()) block.

Based on whether you're using an insert or update trigger and whether you want to pass the original or new entity values to the rules, use either Trigger.new or Trigger.old to access the entity image:

# Apex Trigger

#### Apex Trigger Edit

Apex Trigger Version Settings Is Active 🗹 Q 🚽 | 1 Α A 1 trigger example on Account (after update) 2 ł З if (!System.isFuture()) 4 { 5 for (Account a : Trigger.new) 6 { 7 8 3 9 } 10 else 11 { 12 return; 13 } 14 }

At this point, the only remaining required component is to call the rule execution service. To call the rule execution service from a trigger, invoke the inrule.DecisionClient.executeRulesFromTrigger method and pass the appropriate parameters:

Is Active	
Q	→   🖘 🅐   A 🗛
1	trigger example on Account (after update)
2	i if (!System.isFuture())
4	{     for (Account a : Trigger.new)
6	
7	inrule.DecisionClient.executeRulesFromTrigger('update', a.id, 'Account', 'AccountDefaultRules', false, null);
9	
10	else
11	
12	return;
14	}

The parameters for calling executeRulesFromTrigger, in the order they're passed into the signature:

Parameter Name	Description
eventType	The event type for invoking rules. This will always be 'update'.
entityId	The unique Salesforce identifier for the root object in the request. This follows the naming convention of: "entityName.Id"
objectType	The Salesforce object type of the root object in the request
ruleSetName	The name of an explicit Rule Set to call as the entry point for rule execution.
useEntityPrefix	If set to true, the DecisionClient will append the entity label to the supplied rule set name. For example, if you pass in a ruleSetName of "DefaultRules" and set useEntityPrefix to true, the effective ruleSetName name would be "AccountDefaultRules"
ruleAppName (optional)	Optional. Defining and passing a RuleAppName here allows you to override the default Rule App Name defined in your Custom Setting created during initial configuration for this specific button. If you do not wish to override your Custom Setting, pass "null" here, as displayed in the example above.
entityImage (optional)	Optional. Defining and passing an entity image allows you to capture the entity image at time of trigger execution to ensure its state for rule execution, whereas leaving it out will require the execution to query Salesforce to get the current entity state. This state may differ at that point from its state when trigger execution began due to the asynchronous nature of trigger execution. Passing an entity image requires you first to serialize it into a JSON
	string. An example of this can be found in the screenshot below.

An example of serializing and passing an entity image:



Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

At this point, all required components are set. Any operations you wish to include before/after the invocation of the rule execution service can be added at the points denoted below:

Is Active	e 🗹	
Q	->   👆 (	🕈   А А
1	trigger ex	ample on Account (after update)
3	i if (!s	ystem.isFuture())
5 6 7 8 9 10 11	for { } }	<pre>(Account a : Trigger.new) //Custom pre-rule execution code here inrule.DecisionClient.executeRulesFromTrigger('update', a.id, 'Account', 'AccountDefaultRules', false, null); //Custom post-rule execution code here</pre>
12	else	
13 14 15 16	{ ret } }	urn;

It's important to note that any custom code written after the rule execution service has been called **must not** have any dependencies on outcomes of rule execution, as rule execution will be operating asynchronously, and trigger execution will continue and likely finish well before rule execution does.

#### Known Issues and Limitations of Executing Rules from Triggers

Currently, there are a number of known issues and limitations with executing rules from triggers:

- Currently, only **After Update** and **After Insert** triggers are supported. Due to Salesforce's enforcement of all external service methods running asynchronously, trigger execution and persistence to the Salesforce database will always complete before rule execution has had time to finish, thus preventing the ability to execute rules on the "in-progress" entity image before it has been persisted. As an extension of this limitation, no entity image rollbacks can occur as a result of any validation done during or after the rule execution process.
- Due to the asynchronous nature of trigger execution, the trigger will not wait on a response from the rule execution service before continuing with any additional code contained within it. Therefore, you cannot rely on having a rule response at any point during trigger execution, even if your code relying on it comes after your call to the execution service. It is not recommended to have any code in your trigger that relies on any data contained in the rule response.
- As a result of triggers' asynchronous nature, automatic data refresh on a record page as a result of rule execution is not supported in Classic view, since the page has no way of knowing when rule execution has completed. Notifications will still display once rule execution completes, but the page will have to be manually refreshed by a user for them to be able to see any updates to the record itself.
- Automatic data refresh on record pages is supported in Lightning, however, it requires giving all
  users that will be initiating triggers that execute rules Read permissions to the InRule_Event
  platform event. This can be done in two ways, both of which are explained in the below section
  Adding Permissions to InRule Platform Event

• Due to the fact that triggers execute asynchronously and Formula fields on entities are calculated and persisted asynchronously as well, any rules you attempt to execute from a trigger that make use of a Formula field will be caught in a race condition between the Formula field being calculated and persisted to the database and the rule execution service querying Salesforce to fetch the currently persisted value of that field. If persistence for the Formula field has not yet completed by the time the rule execution service reaches that point, it will pull down an out of date value for that field and your rules will execute using the wrong value. Therefore, it is strongly recommended that you **do not** execute rules that act on Formula fields from triggers. While there is a possibility for it to work correctly, there is a high chance for unexpected behavior to occur and there is no means of controlling whether it will work properly or not on any given execution of that Rule Set from a trigger.

# **Adding Permissions to InRule Platform Event**

Displaying notifications and refreshing the entity form after rules are run from a trigger relies on a Salesforce Platform Event that is installed as part of the InRule package. However, by default, most Salesforce default user profiles do not have read access to Platform Events. In order for notifications to be displayed and data to be properly refreshed on a record page after rule execution via trigger, you must give your users read access to the InRule_Event platform event.

There are 2 methods of doing this, both with their own advantages and disadvantages. Which approach is best is ultimately dependent on the circumstances of your organization.

#### Add Platform Event Permissions via Editing Profiles

The first way of giving the needed permissions to your users is by editing their Profiles. In Salesforce, all users have an assigned Profile that defines their permissions in the environment. Granting the needed permissions to your users through this method can be as simple as editing the profile permissions of the user types that will be setting off your trigger(s). This approach has some pros and cons.

#### Pros:

- Makes changes to entire profiles rather than specific users, meaning once it is done, all that will need to be done for future users is to assign them to a profile with these permissions already granted and they'll be set
- If you are already using predominately custom/cloned Salesforce profiles in your environment, making this change is very quick, easy, and doesn't require any future overhead.

#### Cons:

- Cannot be accomplished with default profiles, which mandates cloning all of your profiles if your users are using predominately default profiles.
- Migrating all users from default profiles to new profile types can be highly time-intensive depending on the size of your org.

To do this approach, a user with system admin permissions must login to Salesforce and go to Setup.

From there, search for Profiles:

Se	tup	Home	Ob
Q Profile	s		
✓ Users	es		
Didn't find v	vhat you'	re looking	for?

Try using Global Search.

Once you've navigated to the Profiles page, select the profile you wish to edit. For this example, we'll be granting the necessary permission to Standard Users

New Profile		
Action	Profile Name *	User License
Edit   Clone	Silver Partner User	Silver Partner
Edit   Clone	Solution Manager	Salesforce
Edit   Clone	Standard Platform User	Salesforce Platform
Edit   Clone	Standard User	Salesforce
Edit   Clone	System Administrator	Salesforce

Once on that profile's page, select Edit:

Profile Standard User	
Users with this profile have the permissions and page	ge layouts listed below. Administrators can change a user's profile by editi
If your organization uses Record Types, use the Edit	t links in the Record Type Settings section below to make one or more re-
Login IP Ranges [0]   Enabled Apex Class Access [0]	0]   Enabled Visualforce Page Access [0]   Enabled External Data Source Access [0] Enabled Flow Access [0]
Profile Detail	Edit Clone View Users
Name	Standard User
User License	Salesforce
Created By	InRule Developer, 11/20/2019 11:47 AM

Scroll down until you see the "Platform Event Permissions" header:

Platform Event Permissions			
		Basic Access Read	Create
	InRule_Events		

Check the "Read" checkbox, then scroll to the bottom of the page and press Save.

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved. CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent. Repeat this process as needed for all profile types that need to be able to initiate rule executing triggers.

If the Read checkbox is greyed out for you and you can't select it, this means you're trying to edit a default Salesforce profile. Salesforce has several "default" profile types that are commonly used; the Standard User profile used in the example above is one such default profile. Unfortunately, Salesforce default profiles **cannot** be edited. This means that if you have users that will be setting off your trigger(s) that use default profile types, you will have to clone that profile, move your users from the default Salesforce profile over to the clone of it, and add the permission on the new profile clone.

To clone a profile, navigate to that profile's detail page as above, but instead of clicking "Edit," click "Clone."

Profile Standard User	
Users with this profile have the permissions and page	e layouts listed below. Administrators can change a user's profile by editi
If your organization uses Record Types, use the Edit	links in the Record Type Settings section below to make one or more re-
Login IP Ranges [0]   Enabled Apex Class Access [0]	Enabled Visualforce Page Access [0]   Enabled External Data Source Access [0] Enabled Flow Access [0]
Profile Detail	Edit Clone View Users
User License	Salesforce
Created By	InRule Developer, 11/20/2019 11:47 AM

Name your new profile appropriately:

You must select an existing profile to clone from	n.
Existing Profile User License Profile Name	Standard User Salesforce Standard User_clone
	Save

Hit save. Your new profile clone has been created. You should now be able to follow the steps above to add the appropriate Platform Event permission.

Once that's done, you need to move your users from the original profile over to the clone.

You will need to repeat this process for every profile type in your organization that needs UI updates when triggers run

#### Add Platform Event Permissions via InRule User Permission Set

The alternative approach to granting the necessary platform event to your users is to apply the InRule User Permission Set that is installed as a part of the InRule Salesforce package. This approach has its own pros and cons:

Pros:

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved. CONFIDENTIAL – Not to be distributed beyond the party to which this document was originally sent. • The InRule User Permission Set comes pre-installed and pre-configured with the InRule Salesforce package, making the only required step being to assign the permission set to the appropriate users

#### Cons:

• Permission Sets must be assigned to specific users, rather than profiles. This means that anytime a new user is created, the permission set must be independently added to that user, creating user management overhead. Managing this may not be viable within a large organization.

To add users to the InRule User Permission Set, go to Setup and search for Permission Set:

***	Setup	Home	Ob
QP	ermission Sets	i	
✓ User	5		
I	Permission Set	s	

Find and select InRule_User_Permissions:

Permission Sets			
On this page yo	u can create, view, and manage permission sets.		
All Permissio	on Sets ✔ Edit   Delete   Create New View		
New			
Action	Permission Set Label 1		
Clone	Buyer		
Clone	Buyer Manager		
Clone	C360 High Scale Flow Integration User		
Clone	CRM User		
Clone	Commerce Admin		
Clone	Data Cloud Home Org Integration User		
Clone	DeliveryEstimationServicePermSet		
Clone	FieldServiceMobileStandardPermSet		
Clone	InRule User Permissions		
Clone	Marketing Cloud Reporting C2C Perm		
Clone	Merchandiser		

Select "Manage Assignments"

InRule_User_Permissions	
Q Find Settings 8 Clone Ma	nage Assignments View Summary
Permission Set Overview	Create Dand completions to the InDule Scient stations and allowing for the dimension of each of
Description	data and notifications from rule execution
License	data and notifications from rule execution.
License Session Activation Required	data and notifications from rule execution.

#### Select "Add Assignments"

-> SETUP > PERMISSION SE InRule_User_Perm	T INRULE_US nissions	ER_PERMISSIONS'		r - Norsense - Haller	r- 1118-201		t ooren aller a	2005-5414	6446-53118317.	
Current Assignments									× 1	Add Assignment
Full Name †	~	Active	Role	~~	Profile	~	User License	~	Expires On	~

You'll be presented with a list of all users in your environment. You can select all users that will need the ability to initiate rule executing triggers. Once you've selected all the relevant users, you can select "Assign," and the permission set will be assigned to all of those users.

These users will now be able to see UI updates when rules are run from triggers.

#### 5 Executing Rules from Salesforce Flow

If you need to integrate rules with more complex process automation, rules can also be run from Salesforce Flow. The InRule integration with Flow lets you run rules against a particular entity record and receive back the status and notifications from rule execution.

There are currently 5 types of Flows in Salesforce:

- Screen Flow With <u>Screen Flow</u> you can create a custom UI (user interface) and guide users through a business process that can be launched from Lightning Pages, Experience Cloud (previously known as Community Cloud), quick actions and more.
- **Record-Trigger Flow** <u>Record-Triggered Flow</u> launches when a record is created, updated, or deleted. So far, we have used Apex triggers for this automation, some of which can now be done using Flows.
- **Scheduled-Triggered Flow** This flow launches at the specified time and frequency for each record in a batch. Traditionally, we have met this kind of requirement by using Apex batch jobs.
- **Platform Event Flow** <u>Platform event flow</u> Launches when a platform event message is received. For example, you can pump the data from an external system in Platform Events and then use Flows to split and save the records in different objects.
- Autolaunched Flow <u>Auto Launches flow</u> launch when invoked by Apex, Process Builder or even REST API.

For more information about these Flows, you can visit the Introduction to Salesforce Flows.

To run rules from a flow, you will need to add an 'Action' element to your flow. You can find the InRule action by searching for 'Run Rules' in the search box. The display name will be 'Run Rules' and the ID will be apex-inrule__DecisionClient

4 Action	×
Search Actions	
Q Run Rules	
Actions	
Run Rules apex-inrule_DecisionClient	

When adding the action, you will need to fill in the following required parameters

Parameter Name	Description
Entity ID (String)	The unique Salesforce identifier for the root object in the request. If
	running a flow from an object page, you can configure this value to be
	passed into an input variable
Object Type (String)	The Salesforce object type of the root object in the request. For
	example, if running rules against an Account entity, this will need to be
	set to a string value of "Account".
Persist Changes (Boolean)	Enable or disable saving changes made to entities during rule execution
Rule App Label (String)	Set to use a specific catalog version label when executing rules,
	otherwise leave blank to always use the latest version of the Rule App
Rule App Name (String)	The name of the InRule Rule App that contains the rule set to run
Rule Set Name (String)	The name of an explicit Rule Set to call as the entry point for rule
	execution.
Use Entity Prefix (Boolean)	Typically set to false. If set to true, the DecisionClient will append the
	entity label to the supplied rule set name. For example, if you pass in a
	ruleSetName of "DefaultRules" and set useEntityPrefix to true, the
	effective ruleSetName name would be "AccountDefaultRules"

Output Name	Description
Notifications (Apex-Defined	Includes notifications fired from rules during execution. This output can
collection)	be assigned to a resource variable of the same type for use in
	subsequent steps. The Apex class for this collection is
	'inruleNotificationMessage'. This class has the following properties:
	<ul> <li>Type: integer value representing notification type. 0 for Info, 1</li> </ul>
	for Warning, and 2 for Error
	<ul> <li>Message: string value containing the notification text</li> </ul>
Errors (Apex-Defined	Includes any errors that prevent rule execution from completing. This
collection)	output can be assigned to a resource variable of the same type for use
	in subsequent steps. The Apex class for this collection is
	'inruleNotificationMessage'. This class has the following properties:
	<ul> <li>Message: string value containing text of the error</li> </ul>
	<ul> <li>Source: string value containing the error source</li> </ul>
Is Success (Boolean)	If rules are not able to be executed for any reason, this value will be set
	to true. If true, additional error information will be available in the 'Errors'
	output collection

The following is an example of what a Flow using rules might look like. You can do other things with the output from the Apex action, but this demonstrates a common use case, displaying rule notifications to the user. The flow starts with Apex action, assigns the output collections to variables, loops over the notifications to build a single string containing all notifications, and then outputs the value to the screen



When looping through the notifications, a decision checks the notification type integer to determine what text to prefix the message with.

😝 Decision			×		
* Label		*API Name 🕚			
Choose Notification Type		Choose_Notification_Type			
Description			la la		
Outcomes For each par	th the flow can take, create an outcome. For each outcome,	specify the conditions that must be met for the flow t	to take that path.		
OUTCOME ORDER 0 +	OUTCOME DETAILS		Delete Outcome		
∐ Info	*Label	*Outcome API Name			
II Warning	Info	Info			
Error	Condition Requirements to Execute Outcome All Conditions Are Met (AND)				
Default Outcome	Resource O (#Loop Notification Loop > Type X) (+ Add Condition	erator Value Equals	Q		
E Assignment			×		
* Label		* API Name 🚯			
Assign Info Prefix		Assign_Info_Prefix	]		
Description					
			là		
Set Variable Value	es				
Each variable is modifi	ed by the operator and value combination.				

Variable	Operator	Value	
Aa NotificationsText ×	Add	Info:	Ê
+ Add Assignment			

The combined notification text value is then displayed on the screen

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Edit Screen	
Run Rules	← Display Text "*
Display Text	* API Name
{!NotificationsText}	Body
Pause Previous Finish	Insert a resource Q
	{!NotificationsText}
	Salesforce Sans 🔻 12 💌
	> Set Component Visibility

#### 6 Executing Rules from REST Endpoint

As of v5.7.3 rules may also be executed by interacting with the Decision Client over REST. This easily enables external workflows, such as Power Automate, to execute rules from the Decision Client. A POST endpoint is available on the Decision Client that will accept a request to execute rules and return a response containing information on the entity changes or information about errors encountered.

# **Authentication with Salesforce**

Before sending your request to execute rules, an access token needs to be retrieved. This will be used in the rule execution request to authenticate with the Salesforce REST API. Below is an example request body that might be used in a POST request to your login URL to receive an access token. More information about REST API authorization can be found in the <u>Salesforce REST API Developer Guide</u>.

```
{
    "grant_type": "password",
    "client_id": "{{clientId}}",
    "client_secret": "{{clientSecret}}",
    "username": "{{username}}",
    "password": "{{password}}{{secretToken}}"
}
```

# **Rule Execution Request**

The URI for Decision Client rule execution REST endpoint will use the subdomain for your Salesforce org.

https://{{DomainName}}.my.salesforce.com/services/apexrest/inrule/executerules

The access token from the authorization request will then need to be provided as a bearer token in your authorization headers.

Finally, you will need to supply the body for your salesforce request. Below is information on the request parameters that can be sent as part of your Rule Execution Request, as well as an example request body.

Parameter Name	Description
ld (String)	The unique Salesforce identifier for the root object in the request. This is a property available on all Salesforce objects and be accessed with: "entityName.Id"
ObjectType (String)	The Salesforce object type of the root object in the request. For example, if running rules against an Account entity, this will need to be set to a string value of "Account".
RuleSetName (String)	The name of an explicit Rule Set to call as the entry point for rule execution.
RuleAppName (String) (Optional)	Optional. Defining and passing a RuleAppName here allows you to override the default Rule App Name defined in your Custom Setting created during initial configuration for this specific button. If you do not wish to override your Custom Setting, pass null here.
RuleAppLabel (String) (Optional)	Optional. Defining and passing a rule app label here allows you to override the rule app label configured on the execution service. If you do not wish to override your execution service's configuration, pass null here.
EntityImage (String) (Optional)	Optional. This allows you to pass in the serialized JSON string of an entity image as it exists at the time of calling the DecisionClient, which will have rules execute against the entity image passed in, not the entity image as it exists in Salesforce when the execution process reaches the execution service. If you do not wish to pass this in, pass null instead.
PersistChanges (Boolean) (Optional)	Optional. This allows you to define whether to persist data changes made during rule execution to Salesforce. Pass in null to default this value to true.

```
I
"Id": "0016g000015JnOnAAK",
"ObjectType": "Account",
"RuleSetName": "AccountDefaultRules",
"RuleAppName": "SalesforceRules",
"PersistChanges": false
}
```

```
Appendix F: Azure App Service Plan & Application Insights
Configuration
```

# **Azure App Service Plan Overview**

The Salesforce Rule Execution Azure App Service runs on an Azure App Service Plan. The ARM Template deployment process outlined in <u>Section 3.3.2: Rule Execution App Service for Salesforce</u> 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 "F1" (Free) pricing tier, but this can be increased based on your scaling and configuration needs.

Important: If you leave the ARM template configured to re-deploy the App Service Plan, updating an existing one, the pricing tier of that App Service Plan will be set to the default pricing tier, regardless of what it may currently be set to. If you do not wish the pricing tier to be reverted to default, it is recommended you follow the steps below.

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 parameters file are necessary.

## Configuring the ARM Template to Use an Existing Azure App Service Plan

#### 1: Locate azuredeploy.parameters.json

The ARM template parameters file is located in the *RuleExecutionAzureService* folder as, defined in <u>Section 3.3.2: Rule Execution App Service for Salesforce</u>



#### 2: Populate "appServicePlanName" parameter

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



#### 3: Set "createOrUpdateAppServicePlan" parameter

Next, just below the "appServicePlanName" parameter be sure to set the parameter called "createOrUpdateAppServicePlan" to false



#### 4: [Optional] Create "servicePlanResourceGroupName" parameter

In the event the App Service Plan you intend to use is located in a different resource group than the one you are deploying the ARM template against, you 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.** 



#### 5: Save azuredeploy.parameters.json and continue deployment

Save and close the file. You can now proceed with the deployment process outlined in <u>Section 3.3.2</u>: <u>Rule Execution App Service for Salesforce</u> as normal; your rule execution app service will now deploy to the App Service Plan you defined in the steps above

#### **Azure App Insights Overview**

For improved logging capabilities, the Salesforce Rule Execution Service is configured to use an Azure App Insights resource as a logging sink in addition to the logging the App Service itself already has. ARM Template deployment process outlined in <u>Section 3.3.2: Rule Execution App Service for Salesforce</u> will, by default, automatically deploy an App Insights resource for you.

The app insights resource will aggregate all the logs generated from the execution service. These logs can be tremendously useful for debugging any issues encountered with the Rule Execution Service. Depending on the level of event logging configured for the <u>Rule Execution Service Event Log</u>, these logs can add insight into rule executions, entity loading, overall execution timing, and any errors encountered during rule execution.

## App Insights in non-Standard Azure Instances (Government Cloud)

If this is your first time deploying the arm template and you would like for the template to create the app insights resource for you, then no other configuration is required.

However, if you would like to use a pre-existing app insights resource then you need to set the **appInsightsInstrumentationKey** and the **appInsightsConnectionString** for that resource in the azuredeploy.parameters.json. Then proceed with the rest of the deployment steps outlines in <u>Section</u> <u>3.3.2: Rule Execution App Service for Salesforce</u>

## Appendix G: InRule SaaS Portal Configuration

InRule SaaS offers customers the most streamlined deployment process, eliminating the need to install and manage the InRule App Services in Azure, as InRule will manage the deployment of the Rule Execution Service for Salesforce for you. SaaS customers are also able to use their SaaS portal to access the information needed for configuring the InRule App for Salesforce and provide Service Account credentials for the SaaS-hosted Rule Execution Service to connect to a Salesforce environment.

## **Access Solution Configuration Information**

SaaS Customers can find the information needed for configuring the InRule App for Salesforce through the Configuration page of their SaaS portal. This page will have a section for each Rule Execution Service instance hosted for the customer by InRule. Please note that only the Salesforce Runtime Settings are currently used by the Salesforce Execution Service. In a future release, settings from the Execution Server Settings section will also contribute to the Salesforce Execution Service. In the interim, if you need any of those settings changed, please reach out to InRule Support.

ÎNRULE Configuration	Decisioning -	Machine Learning 🝷	Process Automation -	Analytics 🝷	trial-b2v12a-451 🔹 🔕 🌲 🧰
Configuration					
Dynamics 365 Runtime Settings					<b>~</b>
Execution Runtime Overrides					<b>v</b>
Execution Server Settings					<b>*</b>
General Settings					¥.
Salesforce Runtime Settings					<b>~</b>

Additional configuration information can be located on the Provision Resources page available from the cogwheel icon in the top right corner of the SaaS portal.



The Provision Resources screen is where you can find read-only configuration information such as the Rule Execution Service URL and the Rule Execution Service API Key.

## **Providing Connection Information**

Clicking the drop-down arrow for a Salesforce section will allow you to provide connection information that will allow the SaaS-hosted Rule Execution Service to interact with your Salesforce environment. You must configure authentication information for both Salesforce and for your Rule Execution Service

#### **Salesforce** Authentication

In order to authenticate to Salesforce, you must configure the appropriate login URL for your Salesforce environment, the service account's username, password, and security token, and the consumer key and

consumer secret associated with the Connected App you created as a part of <u>Section 3.3.3</u>: <u>Configure</u> <u>the InRule App</u>.

Salesforce Authentication			Show Values     Test Connection
Inherited From &	Setting e	Value 😫	
Tenant: trial-b2v12a-451	Consumer Key	(Value Hidden)	
Tenant: trial-b2v12a-451	Consumer Secret	(Value Hidden)	
Tenant: trial-b2v12a-451	Login Url	(Value Hidden)	
Tenant: trial-b2v12a-451	Password	(Value Hidden)	
Tenant: trial-b2v12a-451	Security Token	(Value Hidden)	
Tenant: trial-b2v12a-451	Username	(Value Hidden)	

#### **Rule Execution Service Authentication**

This configuration section defines the authentication type that will be used by Salesforce to authenticate to the Rule Execution Service. Currently, the supported authentication types are API Key and a legacy username/password basic authorization. Support for the basic auth configuration will be discontinued at an unspecified date in the future. If using the basic auth configuration, you will additionally need to supply a value for the username and password settings.

On the Salesforce side, the same credentials will need to be configured on the Named Credential described in <u>Section 3.3.3: Configure the InRule App</u>.

Rule Execution Service Authentication		
Inherited From 🖨	Setting 🖨	Value 🖨
Tenant: di-dev	Auth Type	API Key

#### **Endpoint Override Configuration (Optional)**

As of version 5.5, InRule for Salesforce now supports Overriding Endpoint Configuration via Azure App Service App Settings. This allows for the overriding of various endpoint settings configured on a rule app, such as REST API URLs or Database Connection Strings, by setting App Settings on your execution service App Service.

To set an endpoint override on your app service, simply navigate to your rule execution app service, go to the Configuration view, and click the drop-down arrow for Execution Runtime Overrides.

#### Configuration

Execution Runtime Overri	des					÷
API URL			Primary API Key	Secondary API Key		
Inherited From \$	Type 8	Property 8	Name 8	Value 🖨	Show Values     Add New Override	
Tenant: trial-anglgi-132	RestOperation	Body	AuthenticateProcess	(Value Hidden)		
Tenant: trial-anglgl-132	RestService	RestServiceRootUrl	ConcertListingsEndpoint	(Value Hidden)		
Tenant: trial-ahgigi-132	InlineValueList	ValueListitems	SampleinlineValueList	(Value Hiddien)		

#### Click the Add New Override button:

Inherited From 🖨	Type ⊜	Property 😝	Name 😝	Value 🗧	Show Values     Add New Override
Tenant: trial-anglgI-132	RestOperation	Body	AuthenticateProcess	(Value Hidden)	
Tenant: trial-anglgl-132	RestService	RestServiceRootUrl	ConcertListingsEndpoint	(Value Hidden)	
Tenant: trial-anglgl-132	InlineValueLIst	ValueListitems	SampleInlineValueList	(Value Hidden)	
Tenant: trial-ahglgl-132	- 0	• •	0		Save   Cancel

Click the drop-down arrow under the Type column and select the corresponding type for your override. The Property column should auto-populate for you. Note that there is a drop-down arrow on the Property column as well – some Types have multiple properties for you to choose from.

Inherited From 😝	Туре 🖯	Property 🖨	Name 🖨	Value 🖨	Show Values     Add New Override
Tenant: trial-b2v12a-451	<b>*</b>	0 🔽	•	•	Save Cancel

Your endpoint name should match the name of the endpoint in the rule app you wish to override.

The value of the override would then be set to whatever value you wish to override with.

Below is what a properly configured end-result would look like, using an InlineValueList override as an example:

Inherited From 🖨	Type ⊜	Property 🖯	Name 🕀	Value 🖯	Show Values	⊕ Add New Override
Tenant: trial-b2v12a-451	RestService 🔻	RestServiceRootUrl 🔻	SamploEndpoint	https://www.sampleovorrido.com		Save   Cancel

Once your override is set, simply save the changes to your app service. Upon the next execution of rules, the specified endpoint type will be overridden with the supplied value.

## Appendix H: Named Credential Configuration

Named Credentials are used by the Apex code to make secure HTTP requests to the Rule Execution Service. Currently, there are two types of Named Credentials available:

- Named/External Credentials (new) These are created with the newly named credential process introduced when Salesforce renewed named credentials in 2023. This new process provides additional authentication options, more granular access control, and will be supported for the foreseeable future.
- Legacy Named Credentials (old) These are named credentials that were created prior to Salesforce's renewal of the named credential process. While it is still possible to create legacy named credentials today, it is not recommended – Salesforce has stated they will be discontinuing them at an unspecified date in the future. For additional information regarding setting up a Legacy Named Credential, you can find those instructions <u>here</u>.

**Important**: If you have a legacy named credential set up from prior to the InRule 5.8.1 release, you will need to delete your legacy named credential before you will be able to take advantage of the new Named/External credentials. A warning message will display to make you aware of this when testing your connectivity on the InRule application's configuration page.



The instructions below will walk you through the process for setting up a Named/External Credential.

## Setting up the Named/External Credentials

The InRule managed package includes pre-configured named and external credentials for your convenience. Before you can make use of them, you will need to follow the steps below to finish the setup using your user-specific information.

To begin, navigate to the Named Credentials settings page and locate the named credential called InRule ApiKey Credential.

Named Crede	ntials				~			
Named Credentials External	Credentials							
1 Items · Sorted by Label								New 🔻
Label	~	Туре	~	URL	~	External Credential	~	Actions
InRule ApiKey Credential		Secured Endpoint		https://yoururlgoeshere.com		ApiKey Credential		

Click on the label to be taken to the settings page for this named credential. Click on the edit button in the top right corner.

SETUP > NAMED CREDENTIALS InRule ApiKey Credential		Edit Delete
Label InRule ApiKey Credential	Name inrule_Rule_Execution_Service	
URL https://yoururlgoeshere.com		
Enabled for Callouts		
Authentication		
External Credential ApiKey Credential		
Client Certificate		

URL for the rule execution service.
<ul> <li>For self-hosted tenants, this can be found in the Azure portal in the</li> </ul>
overview section for the app service created earlier. For example,
https://sampleservice.azurewebsites.net
<ul> <li>For SaaS tenants, this can be found in the Provisioned Resources page available from the cogwheel icon in the top right corner of the <u>InRule</u></li> <li>SaaS Portal</li> </ul>

Replace the value in the **URL** field with the URL for the rule execution service. Save your changes when done.

Edit InRule A	piKey Credential
*Label InRule ApiKey Credential *URL https://yoururlgoeshere.com Enabled for Callouts	* Name Rule_Execution_Service
Authentication *External Credential ApiKey Credential Client Certificate	
Search Certificates	Q
Callout Options	
Generate Authorization Header	
Allow Formulas in HTTP Header 🕚	
Allow Formulas in HTTP Body (	
	Cancel

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.



Once there, click on the External Credential tab and find the external credential named **ApiKey Credential**. Click on its' label to be taken to its' overview page.

Named Credentials		
Named Credentials           External Credentials           1 items - Sorted by Label		New
Label	Authentication Protocol 🗸	Actions
ApiKey Credential	Custom	

Locate the Principals section and find the principal named **InRule Service Principal**. Click the drop-down arrow on the right side of the screen and click the edit button.

SETUP > NAMED CREDENTIALS ApiKey Credential					Edit Delete
Label ApiKey Credential			Name inrule_ApiKey_Credential		
Authentication Protocol Custom					
Managed Package Access					
Created By Namespace					
Related Named Credentials	~	Name	v	URL	
nRule ApiKey Credential		inrule_Rule_Execution_Service		https://yoururigoeshere.com	
					9
Principals					New
Principals Sequence V Parameter Name			✓ Authentica ✓	Authentication Status	→ Actions

Click the **Add** button next to the **Authentication Parameters** section to add a new authorization parameter. The **Name** field **must be set to ApiKey**.

API key	<ul> <li>A unique key that allows users to verify their identity when interacting with the rule execution service. Your key can be located in one of two locations:</li> <li>For self-hosted tenants, if you have not done so already, you will need to update your parameters as shown <u>here</u> – the api key configured will be the same one you use in Salesforce.</li> <li>For SaaS tenants, your key will be located on the Provision Resources page available through the cogwheel icon in the top right corner of the InRule SaaS Portal.</li> </ul>
---------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Parameter Name			*Sequence Number	f.	
InRule Service Principal			1		
Identity Type					
Named Principal		•			
Authentication Paramete	ers				Add
Parameter 1					Delete
* Name					
АріКеу					
*Value					
•••••					
Principal Access					
Name	~	Entity Type	~	ID	~
InRule_User_Permissions		PermissionSet		0PS8N000001iZkDWAU	

The Value field will need to be set to your API key. Save your changes when done.

The named credential should now be configured correctly. Before your users will be able to make use of the named/external credentials, you will have to give them access. This can be accomplished either by assigning a permission set, or by granting the user's profile access.

## **Providing Named Credential Access via Permission Set**

In order to grant access via a permission set, you will need to assign the **InRule_User_Permissions** permission set to any user(s) you want to enable the named credential for.

To begin, navigate to the Permission Sets setup page. Once there, locate the **InRule_User_Permissions** permission set in the list and click on its' label.

L P	ermission Sets		
Permis	sion Sets		Help for this Page 😡 着
On this page	you can create, view, and manage permission sets.		-
New		A   B   C   D   E   F   G   H	I J K L M N O P Q R S T U V W X Y Z Other All
Action	Permission Set Label +	Description	License
Clone	Buver	Allows access to the store. Lets users see products and categories,	B2B Buyer Permission Set One Seat
Clone	Buyer Manager	Includes all Buyer capabilities, and allows access to manage carts a	B2B Buyer Manager Permission Set One Seat
Clone	C360 High Scale Flow Integration User	Allows integration user to access features specific to C360 High Scal	Cloud Integration User
Clone	CRM User	Denotes that the user is a Sales Cloud or Service Cloud user.	CRM User
Clone	Commerce Admin	Allow access to commerce admin features.	Commerce Admin Permission Set License Seat
Clone	FieldServiceMobileStandardPermSet	Give your mobile workforce access to the Field Service mobile app	Field Service Mobile
Clone	InRule_User_Permissions	Grants Read permissions to the InRule_Event platform event allowing f.	
Clone	Merchandiser	Allow access to commerce merchandising features.	Commerce Merchandiser User Permission Set License Seat

Once on the permission set's overview page, click on the Manage Assignments button above the permission set's description.

SETUP Permission Sets				
Permission Set InRule_User_Permission	าร		1	/ideo Tutorial   Help for this Page 🥑
Q, Find Settings	Clone Manage Assignments View Summary (Beta)			
Permission Set Overview				
Description	Grants Read permissions to the InRule_Event platform event allowing for the dynamic refresh of data and notifications from rule execution.	API Name	InRule_User_Permissions	
License		Namespace Prefix	inrule	
Session Activation Required		Created By	InRule Developer, 2/1/2024 12:44 PM	
Permission Set Groups Added To	0	Last Modified By	InRule Developer, 2/1/2024 12:44 PM	

On the following screen, click the Add Assignment button on the right side.



From here, select any user(s) you wish to grant access to the named/external credential to. Note: You can click the checkbox next to the Full Name column to select all users displayed.

InRule_U	ser_Permiss	ions	MANAGE ASSIGNMENT EXPIRATION					
elect Users to A	Assign							
All Users	me • Filtered by All	users • Updated	a few seconds ago		Γ	Q. Search this	list	\$ • C
Full Name 1	~	Alias 🗸	Username	~	Role V	Active 🗸	Profile	~

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

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

Page 83 of 111

Once done, select the **Next** button in the bottom right corner of the screen. You will be taken to the assignment expiration management screen. From here, you can optionally provide an expiration date for the permission set assignment. Once done, click the **Assignment button** in the bottom right corner.

elect an Expiration Option For Assigned Users   No expiration date   Specify the expiration date   1 Day   1 Week   30 Days   60 Days   Custom Date   Select a time zone  Selected Users	: V Expires On V
No expiration date       Image: Time Zone         1 Day       1 Week       30 Days       60 Days       Custom Date         Select a time zone	• Expires On • •
1 Day     1 Week     30 Days     60 Days     Custom Date	÷ <ul> <li>Expires On</li> </ul>
Selected Lisers	V Expires On V
	✓ Expires On ✓
ull Name $\checkmark$ Role $\checkmark$ Profile $\checkmark$ Active User License $\checkmark$ Expires On	
IRule Developer System Administrator 🗸 Salesforce Never Expires	Never Expires

Your user(s) should now have access to the named/external credentials and should now have access to make authenticated callouts.

## Legacy Named Credential Configuration

Legacy named credentials offer an older, less robust way of setting up authenticated callouts. They are used by the Apex code to make secure HTTP requests to the Rule Execution Service.

In the InRule App, select the "Configuration" tab



Under the "Named Credentials" header, click the "Named Credentials" link

✓ Named Credential Details
Named Credentials are used by the DecisionClient deployed as part of the Salesforce application to make secure HTTP requests to the Rule Execution Service.
Before you can use the named credential included in the InRule Package, InRule ApiKey Credential, you must do the following:
Navigate to the Named Credentials settings page.
• Find the InRule ApiKey Credential listed under the Named Credentials tab and click on it. Click the edit button in the top right corner.

Press the drop-down arrow on the "New" button and select "New Legacy".

	SETUP Named Credentials				
Named C	Credentials External Credentials				
1 Items · S	Sorted by Label				New 👻
Label	~	Туре	✓ URL	✓ External Credential	New Legacy
AII S	Named Creder	Save Cancel	Rule Execution Ser	VICE	
	Name 🥥 URL	InRule_Rule_Execution_Ser	ites.net		
rt	▼ Authentication		-		
	Certificate		9		
	Authentication Protocol	Named Principal			
	Username Password	BasicUserName			
	▼ Callout Options				
	Generate Authorization Header 🥝				
	Allow Merge Fields in HTTP Header 📀				
	Allow Merge Fields in HTTP Body 🥝				
		Save			
Con	figuration				

Configuration	
Form Fields	
Label	This should be set to InRule Rule Execution Service. This value is not required to match exactly, but if you choose something else, make sure to replace the autogenerated Name with the correct one
Name	This value must be set to <b>InRule_Rule_Execution_Service</b> . The name field is how the Named Credential is retrieved in the Apex code, so this needs to match exactly
URL	URL for the rule execution service in Azure. This can be found in the Azure portal in the overview section for the app service created earlier. For example, https://sampleservice.azurewebsites.net
Identity Type	Named Principal

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Authentication Protocol	Password Authentication
Username	Username for accessing the rule execution service. This should be the same value chosen earlier for the 'ruleServiceUsername' parameter when setting up the service in Azure or established via the <u>InRule SaaS Portal</u>
Password	Password for accessing the rule execution service. This should be the same value chosen earlier for the 'ruleServicePassword' parameter when setting up the service in Azure or established via the <u>InRule SaaS Portal</u>
Callout Options	Make sure 'Generate Authorization Header' is checked

#### **InRule Salesforce Logging**

This section will highlight how to configure and view event logging for the InRule for Salesforce App.

#### Enable InRule for Salesforce App Logging

To enable logging from the InRule for Salesforce App, first navigate to **Setup > Develop > Custom Settings**.

Build	
Develop	
Custom Settings	

Once you're in the Custom Settings page, locate the InRule custom settings object and select "Manage"

Action	Label 1	Visibility	Settings Type
Manage 📥	InRule	Public	Hierarchy

Select Edit. Here, you can configure the LoggingLevel custom setting that will define whether or not the InRule for Salesforce App logs to Salesforce and how verbose of logs it will create. The field can be configured with values from 0-3, with each signifying the following:

- 0 Disables all logging
- 1 Logs errors and a minimal amount of information on successful requests
- 2 Logs errors, request information, rule engine notifications, and rule engine validations
- 3 -- Logs the same information as 2, but also includes JSON from the HTTP request and response payloads

		Edit Delete		
<ul> <li>Default Organization Level Value</li> </ul>				
Location			Authentication Password	
Authentication Username			ServiceUrl 🥥	
LoggingLevel	3			

Once you have configured the LoggingLevel, select "Save."

#### Viewing InRule for Salesforce App Log

After installing the InRule for Salesforce App, a new custom tab to access the InRule for Salesforce App logs will have been added. To add it to your navigation bar, click the "+" at the end of the navigation bar.

Home Chatter Campaigns Leads Accounts Contacts Opportunities Forecasts Contracts Orders Cases Solutions Products Reports Dashboards 💽

A list of objects will appear. Select "InRule Logs"

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

ANT CAN Take

V Accounts	💡 Ideas
Analytics	R) Images
Rep Launcher	InRule Loos
Assets	🗶 Leads
Sampaigns Tell me more!	Libraries
Cases	List Emails
Chatter	🗲 Macros
Console	Qoportunities
Consumption Schedules	i Orders
Contact Requests	💄 People
Contacts	Price Books
Content	Products
Contracts	💄 Profile
📩 Contribute	2 Profile Feed
& D&B Companies	2 Profile Overview
S Dashboards	Recommendations
T Data.com	C Reports
2 Documents	5 Scorecards
Duplicate Record Sets	Site.com
Elles	le Solutions
Corecasts	3 Streaming Channels
W Groups	Subscriptions
e Home	👤 User Provisioning Requests

You will be navigated to a new page displaying a list of any recent logs that may have been created. To view a list of all logs, select "All" from the dropdown list at the top of the page and select "Go!"

View: All • Got Edit   Create New View	
Recent InRule Logs	New
InRule Log Id	
a011U00000LGwHi	
a011U00000lkrgU	
a011U00000LGvxS	
a011U00000IklaH	

You can now view all logs created by the InRule for Salesforce App. To view more information about an individual log, you can click on the log ID to view the log details

InRule Log Detail	Edit Delete Clone		
InRule Log Id	a011U00000LW/Tae	Owner	Sigao Devs (Change)
Was Successful	×.	Request	{"Attributes":{{"Name":"type","Value":"Contact"},{"Name":"Id","Value":"0031U00000K1hIV"}, {"Name":"Id","Value":"0031U00000K1hIV"}},"EntityName":"Contact","Id":"0031U00000K1hIV"}
Response	[Entity-null_EntityChangesResponse" [EntityChanges" [[1]" "0301/00000K1hiVQA2": ChangeType":0_Entity"null_EntityImageContainer": (Attrubuse" ["Change" attrubuse" ["None" "Entity" ("None" "Tit"] ("None	Rule App	SalesforceRuleApp
Notifications	{"Notifications":{{"Type":0,"Message":"Contact Email is now user5978@domain.com"}}	Rule Set	AnonymizeContactEmail
Validations	{"Validations":[]}		
Error			
Object Name	Contact	Object Id	0031U00000K1hIV
Event Type	update	Time of Request	4/15/2019 8:56 AM
Created By	Sigao Devs, 4/15/2019 8:56 AM	Last Modified By	Sigao Devs. 4/15/2019 8:56 AM
	Edit Delete Clone		

What fields are and aren't populated is determined by what logging level you configured.

## **Rule Execution Service Event Log**

#### **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 3.3.2: Rule Execution App Service for the Salesforce.

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



#### Select Diagnostic Tools



#### 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.

Level	Date Time 🔺	Event Id	Source	Computer
Level	Date Tim	Event I	Source	Computer
1 Info	01/21/2019 20:12:13	2000	HttpPlatformHandler	RD00155D70EBC2
1 Info	01/21/2019 20:12:13	2000	HttpPlatformHandler	RD00155D70EBC2
1 Info	01/21/2019	2000	HttpPlatformHandler	RD00155D70EBC2

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:

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 provides 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 <u>Section 3.3.3: Rule Execution App Service for the</u> <u>Salesforce</u>.

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

	«	Z Browse	Stop	<b>⊮</b> _≇ Swap	💟 Restart	📋 Delete	🛓 Get	t publish pr	ofile 🛛 🗘 Reset publish pr	ofile		
📀 Overview	A R	esource group	o (change) Im							URL https://crmirint-	webjob.	azurewebsites.net
Activity log	S	tatus (unning								App Service Plan	n Plan (Ba	sic: 1 Small)
<ul> <li>Access control (IAM)</li> <li>Tags</li> </ul>	L	ocation Jorth Central l	JS							FTP/deployment No FTP/deploym	t userna nent use	ne r set
<ul> <li>Diagnose and solve problems</li> </ul>	S	ubscription (d ngineering-De	hange) evTest							FTP hostname ftp://waws-prod	-ch1-03	3.ftp.azurewebsites.windows.net
Deployment	S	ubscription ID f3242e8-7c98-	-4b60-a71	4-3be7b91c	5df1					FTPS hostname ftps://waws-proc	d-ch1-0	33.ftp.azurewebsites.windows.net
🗳 Quickstart	т	ags (change)										
<ul> <li>Deployment slots (Preview)</li> </ul>	11	Billable : Upda	ateMe	CreatedBy :	YourName	CreationDat	te : YYYY-I	MM-DD	Department : Engineering	Project : Upda	ateMe	
🏦 Deployment slots	1								*			
Deployment options (Classic)	1.5											
🐔 Deployment Center		Diagr	nose and	solve probl	ems bleshooting expe	rience	T	Application	on Insights hights helps you detect and diagon	se quality		App Service Advisor App Service Advisor provides insights for improving app
Settings		helps you identify and resolve issues with your web app.			pp.	issues in your apps, and helps you understand what your actually do with it.			nat your users	your users experience on the App Service platform. Recommenda are sorted by freshness, priority and impact to your ap		
Application settings												

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

Name	Value
inrule:logging:level	Hidden value. Click show values button above to view
inrule:repository:licensing:licenseFolder	Hidden value. Click show values button above to view
inruleisftapi:consumerKey	Hidden value. Click show values button above to view
inrule:sfiapi:consumerSecret	Hidden value. Click show values button above to view
inrule:sftapi:loginUrl	Hidden value. Click show values button above to view
inrule:sftapi:password	Hidden value. Click show values button above to view
inrule:sftapi:securityToken	Hidden value. Click show values button above to view
inrule:sftapitusername	Hidden value. Click show values button above to view
inrule:sf:catalog:label	Hidden value. Click show values button above to view
inrule:sf:catalog:password	Hidden value. Click show values button above to view
inrule:sf:catalog:ruleAppDirectory	Hidden value. Click show values button above to view
inrule:sf:catalog:sso	Hidden value. Click show values button above to view
inrule:sf:catalog:uri	Hidden value. Click show values button above to view
inrule:sf:catalog:useInRuleCatalog	Hidden value. Click show values button above to view
inrule:sf:catalog:user	Hidden value. Click show values button above to view
inrule:sf:ruleService:basicPwd	Hidden value. Click show values button above to view
inrule:sf:ruleService:basicRealm	Hidden value. Click show values button above to view
inrule:sf:ruleService:basicUserName	Hidden value. Click show values button above to view

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

inrule:logging:level	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

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

🗟 Save 🗙 Discard	
or Click here to upgrade to a higher SKU and enable additional f	eatures.
+ New application setting 🐵 Show values 🖉 Advance	d edit 🛛 🔽 Filter
Name	Value
inrule:logging:level	Hidden value. Click show values button above to view

#### **Entity Loading Log**

Enabling Info level logging on the Execution Service can provide more detailed performance data. The plugin trace log will include some summary data about the entire rule execution run, but the execution service can log specifics about entity data loading and rule execution from the service. To enable the info level logging set the "inrule:logging:level" app setting to "Info". For all installations this can be done by editing the web.config, but Azure-hosted services can also edit this setting via the Azure portal.

Setting this to "Info" will enable two sets of logs, the standard InRule SDK logs that appear for all InRule products, and the data loading logs that measure entity loading specific to the Salesforce integration. You can find all the details about the Event Logs on the InRule support site <u>here</u>. There are a variety of different events, but the 'ExecuteRuleSet' entry will contain some of the most important measurements, such as compile, execution and external call times.

However, these measurements only capture actual rule execution, and before rule execution happens the entity tree is loaded from Salesforce. The data loading log captures this information. When executing rules from Salesforce, only the root entity is sent from the plugin to the execution service, and the execution service will then use the rule app metadata to determine which relationships and entities need to be loaded. The execution service batches these requests for greater efficiency, but very large entity trees or network latency can still lead to long load times. The entity loading log breaks down the total loading time into each 'batch' of loading requests made, and lists the total load time and all entities loaded in the batch. This information can be used to identify where large numbers of entities are being loaded and how long particular batches take to retrieve from Salesforce.

Taken together, these logs can be used with the plugin trace log to get an end-to-end picture of performance, and help identify which steps consume the most time. An extract of a sample of the data loading log is shown below.

Entity Loading:
Total Loading Time: 911.5114ms
Total Org Service Time: 701ms
Root Entity: inr_projectprofile
Related Entities: inr_projectprofile -> account lookup <inr_account_inr_projectprofile> (1),</inr_account_inr_projectprofile>
/inf_projectprofile -> inf_agreement lookup
in projectorofilegroup lookup (in in projectorofilegroup in profile) (1).
<pre>inr_projectprofile -&gt; inr_agreement collection <inr_inr_projectprofile_inr_agreement> (2),</inr_inr_projectprofile_inr_agreement></pre>
inr_projectprofile -> annotation collection <inr_projectprofile_annotations> (33),</inr_projectprofile_annotations>
inr_projectprofile -> inr_businessapplicationtype collection
<pre><inr_inr_businessapplicationtype_inr_projectpr> (1), inr_projectprofile -&gt;</inr_inr_businessapplicationtype_inr_projectpr></pre>
<pre>inr_integrationdetail collection <inr_inr_integrationdetail_inr_projectprofile> (5),</inr_inr_integrationdetail_inr_projectprofile></pre>

inr_projectprofile -> inr_runtimeapplicationtype collection
<inr_inr_runtimeapplicationtype_inr_projectpro> (4), Time: 180.1637ms
Related Entities: account -> incident collection <incident_customer_accounts> (43),
inr_agreement -> serviceappointment collection <inr_agreement_ServiceAppointments> (69), Time:
128.366ms
Related Entities: incident -> subject lookup <subject_incidents> (8), serviceappointment ->
service lookup <service_service_appointments> (5), Time: 109.0635ms

In addition to always being logged on the execution service when Info logging is enabled, this log can also be returned in the response and logged to the plugin trace log for easier access.

## Appendix J: New Releases and Upgrading Versions

In most cases, updating InRule for Salesforce is a relatively straight forward process. You will effectively follow the same order of operations of the initial installation steps outlined in this project.

#### The components that will need to be upgraded are:

- irAuthor
- Azure Rule Execution Service
- InRule for Salesforce App

# All 3 of these components must be upgraded in all upgrade scenarios, and all three must be upgraded to the same version.

You can review the InRule Version Release Notes <u>here</u>. If you're upgrading a version that's not the latest, keep in mind that you'll need to use a versioned package link for the InRule for Salesforce App – App Exchange only offers the latest version. This distinction is discussed in more detail in <u>Section 3.3.3</u>: <u>InRule for Salesforce App</u>. The link provided in this section is for the version matching the version of this document. If you need an older version than this document's version, **do not** use the link located in Section 3.3.3. You'll need the Deployment Guide for that version.

This appendix discusses some special cases and considerations to be aware of when upgrading.

InRul	e for Sa	alesforce	Version Matrix
Version	Status	Release date	Description
v5.8.1	UNRELEASED	May 31, 2024	New auth type allowing API key-based named credentials, irAuthor Salesforce login fixes, RuleHelper fixes
v5.8.0	RELEASED	Nov 7, 2022	Maintenance Release
v5.7.3	RELEASED	Apr 7, 2022	Azure Marketplace listing, DecisionClient callable from REST, RuleHelper query and caching enhancements
v5.7.2	RELEASED	Jun 30, 2021	InRule for Salesforce configuration in SaaS Portal. Salesforce API limit fixes
v5.7.1	RELEASED		Maintenance Release
v5.7.0	RELEASED	Feb 17, 2021	Expanded execution options for Decisions and Version Labels, data loading performance improvements, Call InRule from Lightning Flow, advanced irX OAuth login flow
v5.6.0	RELEASED	Jun 23, 2020	InRule App on the AppExchange, InRule App content and configuration pages, 'Test Connectivity' button, DecisionClient enhancements
v5.5.1	RELEASED	Feb 18, 2020	Maintenance Release
v5.5.0	RELEASED	Dec 10, 2019	AppInsights logging, Execution App Service endpoint overrides
v5.4.1	RELEASED	Jun 18, 2019	Maintenance Release
v5.4.0	RELEASED	May 27, 2019	ARM Template, hosted InRule App package deployment
v5.3.0	RELEASED	Feb 26, 2019	Maintenance Release
v5.2.0	RELEASED	Oct 25, 2018	Rule Execution Azure App Service for Salesforce

## **Release History**

## **Version Compatibility Considerations**

The below list of items for upgrade consideration includes specific fixes or changes that may require migration or configuration action. Issue number and title are included here, but you can refer to the

release notes section of the support site for further information: <u>Decision Platform Release Notes – InRule</u> <u>Technology</u>. This page includes release notes for all InRule products, but Salesforce release notes will be prefixed with "SF"



*Important:* Again, when upgrading to a new version, irAuthor, InRule for Salesforce App, and the Azure Rule Execution Service should always be upgraded together

Issue Number	Version	Description
SFC-279	5.8.1	A new configuration setting, Auth Type, has been added. For existing users upgrading to 5.8.1, this setting will be set to basic username/password authorization. Authorization credentials <u>configured in</u> <u>the Named Credential</u> must now match the Auth Type configured for the Rule Execution Service.
SFC-278	5.8.1	Added support for the new Named/External Credentials introduced by recent Salesforce updates. Named Credentials that existed prior to these updates are now referred to as Legacy Named Credentials and should be deleted once the new credentials have been configured.
SFC-272	5.8.1	Added support for Webview2 to the irAuthor Salesforce login, resolving an issue with lightning-based login URLs such as https://your- username.lightning.force.com. Webview2 will now be used to render the login page if the runtime is present. If the runtime is not present, the legacy implementation will be used. A new warning log was added when the runtime is not detected.
SFC-261	5.7.3	Added ruleAppLabel parameter to execution service arm template. By default this label is empty. No ruleAppLabel will result in the most recent version of a rule app being used. Customers relying on a label to denote the production ready version of a rule app, such as the LIVE, will want to add the production label to the arm template prior to upgrade deployment.
SFC-48	5.7.0	Added improvements to the InRule Log Entity. Views will need to be manually updated to take advantage of these updates.
SFC-126	5.7.0	Added support for interactive login to irX, allowing login without requiring the Salesforce API token and the Consumer Secret/Key. This also enables support for MFA authentication.

## SaaS Upgrades

For SaaS customers, the upgrade process will be communicated to you by InRule via email. Keep an eye out for these communications, as they will be the primary source of guidance around what to do in upgrade scenarios. Relevant information from these communications includes when an update will be occurring, the new features coming with the update, the maintenance times required to apply the update, and mandatory steps you'll need to take immediately following the update. It is recommended to update via the Power Platform Admin.

## **Execution Service App Settings**

When running an ARM template deployment to upgrade an existing app service, the new deployment will remove all app settings that currently exist and replace them with the settings during the new deployment. Because of this, it is important to save your parameters file when you do a deployment. If you do not have your previous parameters file you can find it in the deployments section of the Azure Portal. This also means that any settings that were changed or added manually, such as logging level or endpoint

overrides, will be removed.

Alternatively, current app setting values can be manually set in the ARM template parameter file prior to deployment to have the template deploy using those values. Should you go this route, be thorough during the transfer process, it is common for values to have been manually changed over time. For a thorough breakdown of the relevant parameters, reference <u>Section 3.3.2: Rule Execution App Service for</u> <u>Salesforce</u>

Lastly, redeploying or upgrading via the ARM template will remove your InRule license from the resulting app service. The license file will need to be manually added back; for a walkthrough on how to do this, reference <u>Appendix K: License Management.</u>

## irX Salesforce Login Updates

As of v5.7, the irX Salesforce plugin now uses web login to Salesforce, rather than the former form-based authentication configuration. This allows for simply signing into Salesforce through Salesforce's native login experience, rather than needing to configure security tokens and consumer secrets/keys within irX. Additionally, this allows for login from accounts with Multi-Factor Authentication (MFA) required, whereas MFA-required accounts were not compatible before (Note: MFA support only applies to irX. Executive service accounts with MFA required are still not supported).

If upgrading from an older version, old usernames and environments will be preserved, but users will need to re-login.

## Default Rule App Label Change

As of v5.7.3, the default Rule App Label configured on the execution service has been changed from "LIVE" to not being set with a value. If a default label is not configured on the execution service, it will now default to using the most recent version of the rule app being run. This has the potential to create an issue during upgrade scenarios where users are relying on the LIVE label to denote their production-ready rule. If using the most recent version of the given rule app is not the desired outcome, the default label on the execution service will need to be set back to LIVE (or whatever the appropriate production label for your existing rule apps may be). This can be done in the ARM template prior to upgrade deployment.

## **InRule Log Entity**

As of v5.7, updates have been made to improve the information displayed in InRule Logs. This includes updates made to the list views. Salesforce does not allow list views to be updated with a package update, so list view updates will need to be performed manually.

To update the "All" list view:

- 1. Navigate to the InRule Logs tab
- 2. Select the "All" list view
- 3. Select the List View Controls
- 4. Select the "Select Fields to Display" option

	InRu	Ie Getting Started Co	onfigurati	on Running Rules InRule I	Logs 🗸	1. Navigate	e to InRule	Logs	11182-1	0.0597/1	5-7-(II)	311-78	3857//
<b>P</b>	All	elogs ▼ 2 Select All	List V	liew			Soloct Lie	t View Controls	Chang	e Owne	P	rintable	e View
29 iter	ns • Sor	rted by Created Date - Filtered by	All inrule	logs • Updated 3 minutes ago		-	Q Search	this list	愈,		G	/	6 7
		InRule Log Id	~	Created Date 4	~	Was Successful 🗸	Rule App	LIST VIEW CONT	ROLS			~	
1		a006g00000Fx6M9		2/17/2021, 12:30 PM		0	SalesforceRules	New		ules			T
2		a006g00000Fx6M4		2/17/2021, 12:29 PM			SalesforceRules	Clone		ules			-
3		a006g00000Fx6Lb		2/17/2021, 12:28 PM			SalesforceRules	Rename		ules			•
4		a006g00000Fx6Lz		2/17/2021, 12:28 PM			SalesforceRules	Charles Cattings		ules			-
5		a006g00000Fx6Ln		2/17/2021, 12:28 PM			SalesforceRules	Sharing Settings		ules			×
6		a006g00000Fx6Lm		2/17/2021, 12:28 PM			SalesforceRules	Edit List Filters	12	ules			Ŧ
7		a006g00000Fx6Lu		2/17/2021, 12:28 PM			SalesforceRules	Select Fields to Disp	ilay	iles			¥
8		a006g00000Fx6L2		2/17/2021, 12:25 PM			SalesforceRules	Delete	ialde	ules	licel	-	V
9		a006g00000Fx6LI		2/17/2021, 12:24 PM		0	SalesforceRules	Reset Column Widt	leids	ules	Jispi	ay	•
10		a006g00000Fx6Lp		2/17/2021, 12:24 PM									×

5. Use the "Select Fields to Display" window to add and order fields for the All view. Below is the recommended configuration for the "All" list view.



The "Recently Viewed" list view is controlled by the Search Layout. As such updating the "Recently Viewed" list view is done by updating the Search Layout for the InRule Log object in the Object Manager.

To update the "Recently Viewed" list view:

- 1. Select "Search Layouts" from the InRule Log Object Manager Screen
- 2. Use the dropdown arrow for the Default Layout to select the option to Edit the Default Layout

		2 11	a aluand
oetails	Search Layouts 39 Items, Sorted by Profile	2. Use to arrow to	ne aropa o select Ec
ields & Relationships	PROFILE	<ul> <li>COLUMNS DISPLAYED</li> </ul>	
age Layouts	Default Layout	InRule Log Id, Created Date, Was Successful, Rule App, Rule Set	
ightning Record Pages uttons, Links, and Actions	Analytics Cloud Integration User	Default Layout	Edit
compact Layouts	Analytics Cloud Security User	Default Layout	
eld Sets	Authenticated Website	Defauit Layout	
object Limits ecord Types	Authenticated Website	Default Layout	
elated Lookup Fliters	Chatter External User	Default Layout	
earch Layouts	1. Select Search Layouts	Default I search	

 Use the "Edit Search Layout" page to add and order fields for the InRule Log Search Layout. Once the changes to the Search Layout are saved, it will update the "Recently Viewed" list view. Below is the recommended configuration for the InRule Log Search Layout.

SETUP > OBJECT MANAGER	
Details	Edit Search Layout InRule Log Search Results
Fields & Relationships	Select the fields to include in this search layout. Note that your choices only determine the display of search results and do not affect the fields that are actually searched. Your choices are determined the field that are available to users when successing their energy to search results columns. Please refer to the columns the field that are available to users when successing their energy to search results columns.
Page Layouts	Selections also determine the news that are available to users when customicany their search results columns. These relet to the online help for <u>more thromation of search re</u>
Linterio Decent Decen	Available Fields Selected Fields
Lightning Record Pages	Record ID  InRule Log Id  From Created Date
Buttons, Links, and Actions	Event Type Was Successful
Compact Lavouts	Execution Time And Rule App App App App App App App App App Ap
	Object Id
Field Sets	Object Name Remove Down
Object Limits	Response Time of Request
Record Types	Trace Details +
Related Lookup Filters	Override the search result column customizations for all users
Fearsh Lavarite	Standard Buttons
Search Layouts	There are no customizable standard buttons for this view.
Search Layouts for Salesforce	Custom Buttons
Classic	Click here to create a new custom list button
Triggers	
Validation Rules	Save

## Named Credential Changes

Salesforce has revamped their Named Credential process – existing Named Credentials are now referred to as Legacy Named Credentials and will be discontinued at an unspecified date in the future. Starting with InRule version 5.8.1, we have added support for the new Named Credential process to the InRule managed package. We recommend following the configuration steps <u>here</u> and then deleting any InRule specific named credentials that existed prior to upgrading to version 5.8.1.

**Important:** Failure to remove the existing legacy credential will result in continued usage of that credential, regardless of whether you have configured the new credential included in the package upgrade.

**Important:** Failure to configure the new credential puts you at risk of a breaking change being introduced by Salesforce when they deprecate support for Legacy Named Credentials.

## **Deleting the InRule Package**

In some situations, it may be necessary to delete the InRule package from your organization to resolve an issue. Before deleting the package, ensure you have backed up any config or log data you want saved. Then you will need to remove all references to the InRule package from any customizations you have made, including things like forms and flows. If you attempt to delete the package without removing these customizations, Salesforce will list any remaining dependencies you need to remove.

## Appendix K: License Management

Licenses are required for both the Catalog Service and the Rule Execution Service. Whether you're deploying an Online or On-Prem service 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 can uploaded either through Azure's App Service Editor tool or via FTP. Walkthroughs of both means are provided below.

For On-Prem services, you will need to leverage the InRule License Activation Utility. A walkthrough of this process can be found <u>here</u>.

## Uploading a License File

#### **Azure App Service Editor**

To upload the InRule license file to your execution service or your catalog service, navigate to the Azure portal and to the appropriate app service. On the left-hand nav-bar, scroll down until you find the App Service Editor option, under the Development Tools header:

Development Tools



On the resulting page, press "Go"



App Service Editor (Preview)

App Service Editor provides an in-browser editing experience for your App code. Learn more



From here, you'll need to navigate to the appropriate file location, which is different depending on if you're adding it for the Execution Service or the Catalog Service:

Execution Service	Catalog Service
-------------------	-----------------

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

#### Right-click on the bin folder on the left-hand side of the Right-click anywhere in the top level of the screen, then select Upload File: wwwroot directory, then select Upload File: **EXPLORE** EXPLORE **▲ WORKING FILES** WORKING FILES **▲** WWWROOT WWWROOT ⊿ bin ▷ hin New File ▶ de Service.svc ▶ es New Folder web.config ⊳ fr Upload Files ⊳ it ▶ ja Find in Folder New File ▶ ko New Folder ▶ ru Copy Ctrl+C ▶ zh Upload Files ▶ zh In Rename F2 Find in Folder In Delete Delete rs.dll Find your InRuleLicense.xml file wherever you have it stored and select Open to finish. Find your InRuleLicense.xml file wherever you have it stored and select Open to finish.

With that, your license file should be uploaded and usable by your Rule Execution or Catalog Service.

#### 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 <u>az webapp deployment list-</u> <u>publishing-profiles</u> 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 + "/bin/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 + "/bin/InRuleLicense.xml");$client.UploadFile($uri,
"LICENSE_FILE_ABSOLUTE_PATH")
```

## Appendix L: 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.

In general, when beginning to troubleshoot a given issue, it is critical that you verify that you are running matching versions of irAuthor, the Rule Execution Service, and the InRule for Salesforce App. Mismatched versioning can be the root of a plethora of unpredictable problems. For more information on this and upgrading InRule components, refer to <u>Appendix J: New Releases and Upgrading Versions</u>.

## **API Authentication to Salesforce**

An authentication error response when attempting to login to Salesforce during rule execution is generally the result of a bad username, password, or security token. All three of these values must be properly configured to be able to authenticate.

You can test your ability to authenticate to Salesforce via API in a number of different ways, depending on your specific scenario:

Within Salesforce, you can navigate to the InRule Configuration page and leverage the Test Connectivity button. This button will verify your execution service's ability to authenticate to Salesforce and provide more specific error handling in the event of an issue.

InRule Getting started configuration Running Rules inRule Logs V	
9// ANNO 19// ANNO 1	NC/22/2016.11.01110/10
Configuring the InRule App	ÎNRULE
Once the dependencies outlined in the Getting Started section are successfully deployed, the InRule App can now be configured within Salesforce. When the configuration is complete, test the Rule Execution Service connection by clicking the following button:	Connected App and Named Credential
Test Connectivity Configuration includes the following three steps:	
Setup a Connected App	
Connected App Details	
Setup Named Credential Fstablish the credentials for Salesforce to authenticate with the Rule Execution Service	
> Named Credential Details	
> Named Credential Access Details	
Setup Custom Settings	
Manage Rule App Name, logging, and caching configurations	
> Custom Settings Details	

For SaaS customers, a similar Test Connectivity button can be found in the SaaS portal's Salesforce configuration section

ULE Configuration	E Configuration			Analytics 🔻		di-dev	0
C	Configuration						
	Dynamics 365 Runtime Settings				¥		
	Execution Runtime Overrides				Ŷ		
	Execution Server Settings				~		
	General Settings				v		
	Salesforce Runtime Settings				Ŷ		
	Salesforce Authentication				Show Values     Test Connection		
	Inherited From @	Setting &	Value @				
	Tenant: d-dev	Consumer Key	(Value Hidd	lent			
	Tenant: di-dev	Consumer Secret	(Veloe Hidde	len)			
	Tenent: di-dev	Login Url	(Value Hidd	lenj			
	Tenant: di-dev	Password	(Value Hidd	lenj			
	Tenant: di-dev	Security Token	(Velue Hidd	lenj			

Alternatively, you can leverage Salesforce's available collections to test authentication via Postman. These collections can be found <u>here.</u>

#### **Security Tokens**

A common cause of authentication issues is confusion around how Salesforce handles API security tokens. **In most scenarios, security tokens are required.** Security tokens are only optional in scenarios when the account attempting to authenticate is connecting from a Trusted IP address. Trusted IPs can be set globally, meaning any user that connects from an IP in the trusted range will be regarded as trusted, or on the Profile level, meaning that only users assigned to the given Profile will be regarded as trusted when connecting from an IP that falls in the trusted range. If the account in question does not fall in a configured Trusted IP range, it **must** have a security token configured.

#### **Multi-Factor Authentication**

Currently, service accounts configured to require multi-factor authentication (MFA) are not supported by Salesforce's authentication endpoint. As a result, MFA cannot be set to "required" for your service accounts, or authentication will fail.

## **Apex Callout Timeout**

When calling out to an external system from Apex via HTTP, the maximum total communication time limit set by Salesforce is 120 seconds. Executing rules involves a single HTTP callout, and the framework is configured to use all 120 seconds for the timeout. If the timeout is reached, you will receive a "Read Timeout Error" when running rules. This can occur because of latency between Salesforce and the rule execution service, or anything else that causes the rule execution service to take more than 2 minutes to respond.

## **Salesforce API Limits**

In order to maintain performance and availability, Salesforce has various limits on the number of requests that can be made to its API. Full documentation of these limitations can be found <u>here</u>, but as an example, in a Developer Edition instance, you are limited to 15,000 calls per 24-hour period. The rule

execution service makes requests via the API to Salesforce to load and save data, and these will count against any limits, but these requests are batched to be as efficient as possible.

## Multiple Collections with the Same Name

There is a known issue that causes a null reference exception to be thrown when loading entities that have multiple collections of the same name on them, with the only difference being capitalization. An example of this scenario can be seen below:

4 🌄 Quote	
CopportunityID	
📴 QuotelD	
📴 QuoteName	
📴 QuoteNumber	
CuoteForms	
📴 QuoteLineltems	
📴 Occupancy Type	
e quoteForms	
📴 PrimaryState	
E HiscoxAttributes	

This is currently not a supported scenario; all collections on a single given entity need to have wholly unique names; differences in capitalization do not adequately distinguish the two collections as unique from each other.

## **Application Insights Location Error**

Application Insights resources are not available in every region, the list of supported regions can be found in <u>Microsoft's Product Availability</u>. By default, the ARM template will attempt to deploy the App Insights resource in the resource group specified for the template deployment. If this resource group is in one of the unsupported regions you will get the following error:



To fix this error we will have to choose a specific region for the Application Insights resource in the ARM template parameters file.

- Locate InRule.Salesforce.Service.parameters.json
   The ARM template parameters file is located in the RuleExecutionAzureService folder as, defined
   in <u>Rule Execution App Service for Salesforce</u>
- Create an "appInsightsLocation" parameter
   Open the file in your text editor of choice. First, create the appInsightsLocation" parameter at
   the bottom of the parameters file. Set the value equal to a region where Application Insights
   resources are offered.



3. Save InRule.Salesforce.Service.parameters.json and continue deployment Save and close the file. You can now proceed with the deployment process outlined in <u>Rule</u> <u>Execution App Service for Salesforce</u> as normal; your rule execution app service will now deploy to the App Service Plan you defined in the steps above

#### Performance

#### **Azure Platform**

The Rule Execution App Service for Salesforce can use either a 64-bit or 32-bit platform in Azure. As of v5.7.3, the configuring the platform will be managed in the Arm Template and will be based on the app service plan. Using the 64-bit platform requires a Basic or higher app service plan, so when a Basic or higher app service plan is used in the Arm Template, the Rule Execution App Service will be deployed with the 64-bit platform. Otherwise 32-bit is used.

#### **Apex Trace Log**

When troubleshooting performance issues, the InRule Log entity included with the package provides detailed metrics to help analyze the various components of rule execution performance. At a high level, the logs provide run times for the various execution steps. The 'Information' section of the log form includes a field for 'Execution Time', which is the total run time for the Apex rules method. In addition, the InRule log also provides a more detailed breakdown of steps in the trace text. These are broken into Execution Service and Save Time.

Execution Service time will typically be longer, as this includes communication latency over HTTP, loading of additional entity data, and execution of rules. Save time will not always show up in trace logs, but if any entity changes come back from rule execution the time needed to save them will be reflected here. The trace log will also include a message noting the number of entities returned from the execution service for saving. This saving should not take much time for a handful of entities but can take several seconds for large change sets.

For a further breakdown of the data loading and execution time from the rule execution service, the 'Info' level event logs for the App Service can be accessed through the Azure Portal.

Included below is a sample trace log that includes these features.

#### Starting rule execution

Copyright© 2024 by InRule Technology, Inc. All Rights Reserved.

Sending request to rule execution service at
callout:InRule_Rule_Execution_Service/SfRuleExecution.svc/ExecuteRulesFromPlugin?eventType=upd
ate&id=0018A00000cPk1CQAS&entityName=Account&ruleAppName=SalesforceRules&ruleSetName=RuleSet1&
ruleAppLabel=&appDomainCache=0.0
Received response from rule execution service
Execution Service Time: 3,221ms
Saving 4 changes to Salesforce
Processing change Insert on entity ID 73c95b94-df07-44c6-8895-fcbc3c422a3c type Contact
Processing change Insert on entity ID 01737977-eb66-46c9-89d2-2d1c94756e7d type Account
Processing change Insert on entity ID 6ed24d2a-77dd-497b-ac7c-4b692f9e8f6c type Case
Processing change Insert on entity ID 3cb009a2-a3e8-46d6-a368-e3892511b39a type
WorkOrder
Save Time: 676ms
Eviting rule evention

### **Request and Response Message Size Limitations**

Salesforce includes a variety of different governors to track and enforce limits on resources used by Apex. One of these is request and response size limitations to HTTP callouts. Salesforce sets this limit at 12 MB for both requests and responses. These limits should not commonly be exceeded, but if you get this error during rule execution, try to reduce the number of notifications, validations, or entity changes you are making in a rule set. These are all returned in the response for processing, and reducing these can reduce the response message size.

## **Enable Background Compilation**

This setting controls whether the download, compilation, and caching of updates to rule applications from the catalog is handled on a background task. When enabled, rule execution will not be blocked while updating the rule app and will instead run immediately against the currently cached version. Self-hosted tenants should ensure this value is set to true for their deployment to avoid unnecessary catalog overhead. For additional information, please view the documentation available on the <u>Configuration</u> <u>Overview</u> page.

## **Disabling State Refresh**

Disabling the state refresh option could improve performance for applications with many RuleHelper calls. State refresh is configured for both .Net Assembly Function Libraries, and User-Defined Functions.

#### .Net Function Assembly Libraries

To disable state refresh for .Net Assembly Function Libraries, begin by navigating to the Endpoints tab in irAuthor. Once there, select the function library containing the function you wish to disable state refresh for.
End Points + Add - 2↓	0	CrmHelper			
🔎 🛋 CrmHelper	.NET Assembly Function Library				
	Assembly:	InRule.Salesforce.RuleHelper Reload			
	Classes				
	Filter:				
	Show all:	✓ Classes ✓ Methods			
	Include Full name		Alias	Use intrinsic	¥
	InRule.Salesford	e.RuleHelper.SalesforceDriver	SalesforceDriver		ß

Select the function being updated from the list and then click the refresh icon on the right.

( Class methods								
Includ	e Name	Туре	Static	Parameters		<b>\$</b>		
✓	LoadMappedChildCollection	System.Void		IRuleExecutionContext context, Entity root,	^	B		
✓	LoadMappedChildCollection	System.Void	$\checkmark$	IRuleExecutionContext context, Entity root,		₫ Re	fresh option	ns
$\checkmark$	LoadMappedChildEntity	System.Void	$\checkmark$	IRuleExecutionContext context, Entity root,				
$\checkmark$	LoadMappedChildEntity	System.Void	$\checkmark$	IRuleExecutionContext context, Entity root,				
$\checkmark$	QueryCollection	System.Void	$\checkmark$	IRuleExecutionContext context, Entity root,				
$\checkmark$	QueryCollection	System.Void	$\checkmark$	IRuleExecutionContext context, Entity root,				

A Refresh Options pop-up should appear – locate the "Refresh all bound objects" checkbox and uncheck it. Click OK.

Refresh Options		×
Refresh all bound objects		
Path	Enable rec	ursion 🛨
		×
	ОК	Cancel

State refresh should now be disabled for this function.

## **User-Defined Functions**

To disable state refresh for User-Defined Functions, begin by navigating to the User-Defined Functions tab of your app in irAuthor. Once there, select the function you wish to disable state refresh for and click the "Modify state refresh options" link under the State Refresh Options section.

User Defined Function				
Return type:	None	•		
Entity:		• • ()		
Parameters:	Name	Data type	Description	+
	FieldName	Text		$\boldsymbol{\times}$
State refresh options				
Manage options for refreshing state affected by this function. Modify state refresh options				

A State Refresh Options pop-up should appear. Deactivate the "Enable full refresh of all bound objects" checkbox and lick Ok.

TR State Refresh Option	ns X
Enable full refresh of	all bound objects
Path	Enable Recursion
	OK Cancel

State refresh should now be disabled for this function.