Powered by Universal Speech Solutions LLC

UniMRCP

Google Dialogflow Plugin

Administrator Guide

Revision: 3

Distribution: Red Hat / Cent OS

Created: December 27, 2017

Last updated: January 31, 2021

Author: Arsen Chaloyan

Table of Contents

1 Overview

This guide describes how to obtain and install binary packages for the Google Dialogflow (GDF) plugin to the UniMRCP server on Red Hat-based Linux distributions. The document is intended for system administrators and developers.

1.1 Applicable Versions

Instructions provided in this guide are applicable to the following versions.

UniMRCP 1.5.0 and above

UniMRCP GDF Plugin 1.0.0 and above

1.2 Supported Distributions

UniMRCP binary packages are currently available only for x86_64 (64-bit) architecture.

Operating System	32-bit	64-bit
Red Hat / Cent OS 7		✓
Red Hat / Cent OS 8		✓

Note: packages for other distributions can be made available upon request. For more information, contact services@unimrcp.org.

1.3 Authentication

UniMRCP binary packages are available to authenticated users only. In order to register a free account with UniMRCP, please visit the following page.

https://www.unimrcp.org/profile-registration

Note: a new account needs to be verified and activated prior further proceeding.

2 Installing RPMs Using YUM

Using the Yellowdog Updater, Modifier (yum), a command-line package management utility for Red Hat-based distributions, is recommended for installation of UniMRCP binary packages.

2.1 Repository Configuration

The content of a typical yum configuration file, to be placed in /etc/yum.repos.d/unimrcp.repo, is provided below.

```
[unimrcp]
name=UniMRCP Packages for Red Hat / Cent OS-$releasever $basearch
baseurl=https://username:password@unimrcp.org/repo/yum/main/rhel$releasever/$basearch/
enabled=1
sslverify=1
gpgcheck=1
gpgkey=https://unimrcp.org/keys/unimrcp-gpg-key.public

[unimrcp-noarch]
name=UniMRCP Packages for Red Hat / Cent OS-$releasever noarch
baseurl=https://username:password@unimrcp.org/repo/yum/main/rhel$releasever/noarch/
enabled=1
sslverify=1
gpgcheck=1
gpgkey=https://unimrcp.org/keys/unimrcp-gpg-key.public
```

The username and password fields included in the HTTPS URI must be replaced with the corresponding account credentials.

2.2 Repository Verification

In order to verify that yum can properly connect and access the UniMRCP repository, the following command can be used.

```
yum repolist unimrcp
yum repolist unimrcp-noarch
```

where *unimrcp* and *unimrcp-noarch* are names of the sections set in the yum configuration file above.

In order to retrieve a list of packages the UniMRCP repository provides, the following command can be used.

```
yum --disablerepo="*" --enablerepo="unimrcp" list available
```

```
yum --disablerepo="*" --enablerepo="unimrcp-noarch" list available
```

2.3 GDF Plugin Installation

In order to install the GDF plugin, including all the dependencies, use the following command.

```
yum install unimrcp-gdf
```

In order to install the additional data files for the sample client application *umc*, the following command can be used.

```
yum install umc-addons
```

Note: this package is optional and provides additional data which can be used for validation of basic setup.

3 Installing RPMs Manually

UniMRCP RPM packages can be installed manually using the *rpm* utility. Note, however, that the system administrator should take care of package dependencies and install all the packages in appropriate order.

The RPM packages have the following naming convention:

```
$packagename-$universion-$packageversion.el$rhelversion.$arch.rpm
```

where

- *packagename* is the name of a package
- *universion* is the UniMRCP version
- *packageversion* is the RPM release version
- *rhelversion* is the Red Hat version
- *arch* is the architecture (x86_64, i686, …)

3.1 Package List

The following is a list of UniMRCP RPM packages required for the installation of the GDF plugin.

Package Name	Description
unimrcp-gdf	GDF plugin to the server.
unigrpc	UniMRCP edition of the gRPC library.
umc-addons	Sample en-US data files used with umc. [Optional]
unilicnodegen	Node information retrieval tool, required for license deployment.
unimrcp-server	Shared library and application of the server.
unimrcp-client	Shared libraries and sample applications of the client. [Optional]
unimrcp-demo-plugins	Set of demo plugins to the server. [Optional]
unimrcp-common	Data common for the client and the server.
uniapr	UniMRCP edition of the Apache Portable Runtime (APR) library.

uniapr-util	UniMRCP edition of the Apache Portable Runtime Utility (APR-Util) library.
unisofia-sip	UniMRCP edition of the Sofia SIP library.

3.2 Package Installation Order

Note that all the RPM packages provided by UniMRCP are signed by a GNU Privacy Guard (GPG) key. Before starting the installation, you may need to import the public key in order to allow the *rpm* utility to verify the packages.

```
rpm --import https://unimrcp.org/keys/unimrcp-gpg-key.public
```

Packages for the APR, APR-Util and Sofia-SIP libraries must be installed first.

```
rpm -ivh uniapr-$aprversion-$packageversion.el$rhelversion.$arch.rpm
rpm -ivh uniapr-util-$apuversion-$packageversion.el$rhelversion.$arch.rpm
rpm -ivh unisofia-sip-$sofiaversion-$packageversion.el$rhelversion.$arch.rpm
```

Then, a package containing common data for the client and the server, and a package for the server should follow.

```
rpm -ivh unimrcp-common-$universion-$packageversion.el$rhelversion.$arch.rpm
rpm -ivh unimrcp-server-$universion-$packageversion.el$rhelversion.$arch.rpm
```

Next, a package containing the utility tool *unilicnodegen*, required for license deployment.

```
rpm -ivh unilicnodegen-$toolversion-$packageversion.el$rhelversion.$arch.rpm
```

Next, a package containing the gRPC library.

```
rpm -ivh unigrpc-$grpcversion-$packageversion.el$rhelversion.$arch.rpm
```

Finally, a package containing the GDF plugin should follow.

```
rpm -ivh unimrcp-gdf-$universion-$packageversion.el$rhelversion.noarch.rpm
```

4 Obtaining License

The GDF plugin to the UniMRCP server is a commercial product, which requires a license file to be installed.

4.1 License Type

The following license types are available:

- Trial
- Production
- Test and Development

4.2 Node Information

The license files are bound to a node the product is installed on. In order to obtain a license, the corresponding node information needs to be retrieved and submitted for generation of a license file.

Use the installed tool *unilicnodegen* to retrieve the node information.

```
/opt/unimrcp/bin/unilicnodegen
```

As a result, a text file *uninode.info* will be saved in the current directory. Submit the file *uninode.info* for license generation to services@unimrcp.org by mentioning the product name in the subject.

4.3 License Installation

The license file needs to be placed into the directory */opt/unimrcp/data*.

```
cp umsgdf_*.lic /opt/unimrcp/data
```

5 Obtaining Service Credentials

In order to utilize the Google Cloud Dialogflow API, a corresponding service account credentials file needs to be retrieved from the Google Cloud Platform Console and further installed to the UniMRCP server.

5.1 Project Creation

Create a project in the Google Cloud Platform Console, if not already created. Projects allow to manage all Google Cloud Platform resources, including deployment, access control, billing, and services.

1. Open the Cloud Platform Console.

   https://console.cloud.google.com

2. In the drop-down menu at the top, select a project *My First Project* created by default or create a new project.

5.2 Project Billing

Enable billing for your project, if not already enabled. Enabling billing allows the application to consume billable resources such as Dialogflow API calls. See Cloud Platform Console Help for more information about billing settings.

5.3 Dialogflow API

In the Google Cloud Platform Console, navigate to API & Services and enable the Dialogflow API.

5.4 Credentials Retrieval

Download a service account credentials file.

1. In the Google Cloud Platform Console, navigate to API & Services > Credentials > Create credentials > Service account key
2. Under **Service account**, select *New service account*.
3. Under **Service account name**, enter a service account name of your choice. For example, *accessor*.
4. Under **Role**, select Project > Owner.

   To better understand the Cloud IAM roles that you can grant to your service account to access Cloud Platform resources, check out the following page.

   https://cloud.google.com/iam/docs/understanding-roles

5. Under **Key type**, leave JSON selected.
6. Click **Create** to create a new service account and download the json credentials file.

5.5 Credentials Installation

The downloaded json credentials file needs to be placed into the directory */opt/unimrcp/data*.

```
cp *.json /opt/unimrcp/data
```

6 Configuring Server and Plugin

6.1 Plugin Factory Configuration

In order to load the GDF plugin into the UniMRCP server, open the file *unimrcpserver.xml*, located in the directory */opt/unimrcp/conf*, and add the following entry under the XML element <*plugin-factory*>. Disable other recognition plugins, if available. The remaining demo plugins might also be disabled, if not installed.

```
<!-- Factory of plugins (MRCP engines) -->
<plugin-factory>
  <engine id="Demo-Synth-1" name="demosynth" enable="true"/>
  <engine id="Demo-Recog-1" name="demorecog" enable="false"/>
  <engine id="Demo-Verifier-1" name="demoverifier" enable="true"/>
  <engine id="Recorder-1" name="mrcprecorder" enable="true"/>
  <engine id="GDF-1" name="umsgdf" enable="true"/>
</plugin-factory>
```

6.2 Logger Configuration

In order to enable log output from the plugin and set filtering rules, open the configuration file *logger.xml*, located in the directory */opt/unimrcp/conf*, and add the following entry under the element <*sources*>.

```
<source name="GDF-PLUGIN" priority="INFO" masking="NONE"/>
```

6.3 GDF Plugin Configuration

The configuration file of the plugin is located in */opt/unimrcp/conf/umsgdf.xml*. Default settings should be sufficient for generic use.

Refer to the *Usage Guide* for more information.

7 Validating Setup

Validate your setup by using the sample UniMRCP client and server applications on the same host. The default configuration and data files should be sufficient for a basic test.

7.1 Setting up Sample Dialogflow Agent

Follow the instructions to import a sample Dialogflow room reservation agent.

In order to identify the Dialogflow agent, an associated project ID must be specified in the configuration file of the plugin, located in */opt/unimrcp/conf/umsgdf.xml*.

```
<streaming-recognition
  interim-results="false"
  language="en-US"
  max-alternatives="1"
  project-id="abcdefgh-ijklmn-123456"
/>
```

7.2 Launching Server

Launch the UniMRCP server application.

```
cd /opt/unimrcp/bin
./unimrcpserver
```

In the server log output, check whether the plugin is normally loaded.

```
[INFO]   Load Plugin [GDF-1] [/opt/unimrcp/plugin/umsgdf.so]
```

Next, check for the license information.

```
[NOTICE] UniMRCP GDF License

-product name:    umsgdf
-product version: 1.0.0
-license owner:   Name
-license type:    trial
-issue date:      2017-12-28
-exp date:        2018-01-27
-channel count:   2
```

```
-feature set:    0
```

Next, check for the service account credentials.

```
[NOTICE] Set Google App Credentials /opt/unimrcp/data/My First Project-a78…c15.json
```

7.3 Launching Client

Note: the optional package *umc-addons* must be installed for this test to work.

Launch the sample UniMRCP client application *umc*.

```
cd /opt/unimrcp/bin
./umc
```

Run a typical speech recognition scenario by issuing the command *run gdf1* from the console of the *umc* client application.

```
run gdf1
```

This command sends a RECOGNIZE request to the server and then starts streaming a sample audio input file *bookroom.pcm* to recognize.

Check for the NLSML results to be returned as expected.

```
<?xml version="1.0"?>
<result>
  <interpretation grammar="builtin:speech/transcribe" confidence="1">
    <instance>
      <query_text>book a room</query_text>
      <action>room.reservation</action>
      <parameters>
        <guests></guests>
        <duration></duration>
        <location></location>
        <time></time>
        <date></date>
      </parameters>
      <fulfillment_text>I can help with that. Where would you like to reserve a
          room?</fulfillment_text>
      <fulfillment_messages>
        <text>
```

```
      <text>I can help with that. Where would you like to reserve a room?</text>
    </text>
    <platform>FACEBOOK</platform>
  </fulfillment_messages>
  <fulfillment_messages>
    <text>
      <text>I can help with that. Where would you like to reserve a room?</text>
    </text>
  </fulfillment_messages>
  <output_contexts>
    <name>projects/abcdefgh-ijklmn-
      123456/agent/sessions/ee86f76015ee5943/contexts/e8f6a63e-73da-4a1a-8bfc-
      857183f71228_id_dialog_context</name>
    <lifespan_count>2</lifespan_count>
    <parameters>
      <duration></duration>
      <guests></guests>
      <location></location>
      <time></time>
      <date></date>
      <duration_original></duration_original>
      <time_original></time_original>
      <guests_original></guests_original>
      <location_original></location_original>
      <date_original></date_original>
    </parameters>
  </output_contexts>
  <output_contexts>
    <name>projects/abcdefgh-ijklmn-
      123456/agent/sessions/ee86f76015ee5943/contexts/room_reservation_dialog_params_
      location</name>
    <lifespan_count>1</lifespan_count>
    <parameters>
      <guests></guests>
      <duration></duration>
      <location></location>
      <time></time>
      <date></date>
      <duration_original></duration_original>
      <time_original></time_original>
      <guests_original></guests_original>
      <location_original></location_original>
      <date_original></date_original>
    </parameters>
  </output_contexts>
  <output_contexts>
    <name>projects/abcdefgh-ijklmn-
      123456/agent/sessions/ee86f76015ee5943/contexts/room_reservation_dialog_context<
      /name>
    <lifespan_count>2</lifespan_count>
    <parameters>
```

```
            <duration></duration>
            <guests></guests>
            <location></location>
            <time></time>
            <date></date>
            <duration_original></duration_original>
            <time_original></time_original>
            <guests_original></guests_original>
            <location_original></location_original>
            <date_original></date_original>
          </parameters>
        </output_contexts>
        <intent>
          <name>projects/abcdefgh-ijklmn-123456/agent/intents/e8f6a63e-73da-4a1a-8bfc-
              857183f71228</name>
          <display_name>room.reservation</display_name>
        </intent>
        <intent_detection_confidence>1</intent_detection_confidence>
        <diagnostic_info>
        </diagnostic_info>
        <language_code>en-us</language_code>
      </instance>
      <input mode="speech">book a room</input>
    </interpretation>
  </result>
```

Visually inspect the log output for any possible warnings or errors.

Note that utterances are stored in the *var* directory, if the corresponding parameter is enabled in the configuration file *umsgdf.xml* and/or requested by the client.