Enghouse Interactive

Contact Center as a Service

Provisioning Portal Deployment Guide

Contents

1: Intro

This document is for implementation engineers. It describes the procedures for preparing the CCaaS platform to support the Provisioning Portal and the procedures for implementing Provisioning Portal on CCaaS.

What's new

- Appendix G - Tenant configuration — the latest version of the Provisioning Portal introduces addons configuration, which allows landlords or administrators to customize and configure a set of CCSP addon modules directly through the Provisioning Portal interface.
- You can now configure Working Hours for custom chat across different time zones, and this feature is also compatible with the Social Connector.
- Added a new Change Audit feature that enables landlords or tenant administrators to review detailed information about recent SQL changes. This includes execution details of stored procedures and their parameters, all accessible through the Provisioning Portal.
- Enable auto SQL update — enhanced the PS.TscSvc service to ensure that the Tenant creation service automatically performs the necessary Provisioning Portal SQL updates upon startup.
- Finalize Provisioning Portal deployment — you can now configure Working Hours for Custom Chat across different time zones, and this feature is also compatible with the Social Connector.
- Appendix J - SSO configuration — Provisioning Portal now supports enhanced Single Sign-On (SSO) functionality.

Legal disclaimer

This document is governed by the terms of the software license agreement and applicable contract (including addendums) entered into with Enghouse.

Support

To submit comments or questions about the information in this guide, please open a case with Enghouse Interactive Support.

2: Prepare the CCSP platform for Provisioning Portal

This chapter contains the following information:

> **Note**
> Microsoft .NET 4.8 or a newer .NET standard framework is required to be installed prior to the deployment of Provisioning Portal web applications and/or tenant creation Windows Services.

Prepare the Designer scripts template folder

This folder holds a collection of Designer scripts template subfolders by source tenant ID. Subfolder 17 referenced below is a sample of the Designer scripts template that may be used for source tenant provisioning.

> **Note**
>
> - A synchronized copy of the folder should be found on at least two host instances for load-balancing and N+1. Select an existing host or provision a new host as the **source script host**. The host instances should reside on the IPC network that is responsible for providing xCS scripts access through Web Server (IIS) running on the host instances. The folder on the source host should be shared as a network folder for Provisioning Portal to read (reload) and to write (publish) scripts. Set up the folder synchronization process to replicate the source folder on to other host instances in the same synchronization group.
> - You can provision one network folder for Provisioning Portal to read (reload) scripts, and another network folder for Provisioning Portal to write (publish) scripts. Consult with ASG Project Manager to find out more.

> **Caution**
> The synchronization process must be monitored for errors. Automatic validation process must be in place to ensure source content is replicated correctly to the destinations. Source and destination folders must be monitored for disk usage.

To prepare the Designer scripts template folder:

1. Copy *TemplateScript* folder from Provisioning Portal package to *C:* as *C:\IVRScripts\TemplateScript* on the source script host. This folder may reside on a different drive and path. This document uses *C:\IVRScripts*.

2. Create VD in IIS called *IVRScripts* with physical path pointing to *C:\IVRScripts*.
3. Open Designer workspace *C:\IVRScripts\TemplateScript* with Designer tool.

> **Note**
>
> To use Designer with Provisioning Portal modified files, you need to perform the following steps on each IIS server where Designer is installed.
>
> 1. Back up
>   *C:\Program Files (x86)\CosmoCom\Server Components\CosmoDesigner\CosmoDesignerOCX.js*
> 2. Modify the above file:
>   From:
>    ```
>    ' <param name="Enforcement" value="1">' +
>    ```
>   To:
>    ```
>    ' <param name="Enforcement" value="0">' +
>    ```

4. Replace *FQDN* in Path block in all scripts with FQDN of the new VD.

> **Note**
>
> For example, if your new VD is *scripts.yourdomain.loc*, change the Path in the Path block
> From:
> `http://dnccsp12.dn12.loc/IVRScripts/$calldata.tenantid$/PrpScript/`
> To:
> `http://scripts.yourdomain.loc/IVRScripts/$calldata.tenantid$/PrpScript/`
>
> The Path block is located at the beginning of:
>
> - IVR
> - Call Arrival
> - Music On Hold
> - Music On Hold In Queue
> - Outgoing Call

5. Share the *C:\IVRScripts* folder as, for example, *\\source_host\IVRScripts*. For the sharing permission, grant full permissions for svccosmocall only. For the folder properties/security/permissions, add Authenticated Users to the permissions list.
6. Set up *IVRScripts* folder synchronization to replicate the folder from the source host to other host instances in the same synchronization group. Set up automatic validation and monitoring processes for the synchronization process and *IVRScripts* folder copies, respectively.
7. On the other host instances, create VD in IIS called *IVRScripts* with physical path pointing to *C:\IVRScripts*.

> **Note**
>
> - Perform steps 3 and 4 every time the source script files are copied/updated from Provisioning Portal package on to the source host.

- The FQDN mentioned in step 4 is referenced by the Application URL of the source tenants to be created shortly. Such URL(s) are cloned from the source tenant to the new tenants. The FQDN should resolve to RR DNS IPC addresses of the host instances in the IVRScripts folder synchronization group. The FQDN in the URL may also be resolved by DNS to an LB VIP address on the IPC network that targets IPC address of the host instances in the IVRScripts synchronization group.

Provision the source tenant

The source tenant is referenced by Provisioning Portal when defining a template.

Note

The new tenant to provision in this step serves as the Provisioning Portal demo tenant. This tenant will be associated with the demo script template that is shipped with the Provisioning Portal package.

To provision the source tenant:

1. Provision a new tenant as the source tenant, using CCSP Tenant Provisioner.
2. On the source scripts host, copy *C:\IVRScripts\TemplateScript* as *C:\IVRScripts\[new tenant ID],* replacing *[new tenant ID]* with the actual ID of the new tenant, for example, *C:\IVRScripts\1022*.

Configure the source tenant

Configure the new source tenant with the legacy Administrator client tool.

To configure the source tenant:

1. Create a new application called *PrPScript* with URL pointing to *http://[your IIS FQDN]/IVRScripts/[new tenant ID]/PrpScript/,* replacing:
   - *[your IIS FQDN]* with RR DNS FQDN or LB VIP FQDN for scripts access
   - *[new tenant ID]* with the actual ID of the new tenant

   Note

   - After you've made a copy of the TemplateScript, you'll find a subfolder called *PrPScript*, do NOT change this folder name.
   - The Application URL should always be configured in the format stated above.

2. Create the following items for validating, using the scripts template provided in the Provisioning Portal package.

Item	Name	Details
Groups Permission profiles	PrPGroupDoNotRemove	**Note**: This is a special group for Provisioning Portal internal use only. Leave all

Item	Name	Details
		parameters for this group empty (default). This group must be created for all tenants.

	Sales	• Primary queue
		  ◦ Sales
		• Secondary queue
		  ◦ Support
		• Release codes
		  ◦ Sales Lunch

	Support	• Primary queue
		  ◦ Support
		• Secondary queue
		  ◦ Sales
		• Release codes
		  ◦ Support Lunch

	Sales	• All permissions for group Sales, queue Sales • Allow creation/deletion of objects
	Support	• All permissions for group Support, queue Support • Allow creation/deletion of objects

Personnel	Sales1	• Profile
		  ◦ (Agent)
		• Group
		  ◦ Sales
		• Skills
		  ◦ Sales
		  ◦ English
		• Team
		  ◦ TeamA

	Sales2	• Profile
		  ◦ Sales
		• Group
		  ◦ Sales

Item	Name	Details

		• Skills
		  ◦ Sales
		  ◦ English
		  ◦ Spanish
		• Team
		  ◦ TeamA
		  ◦ TeamB

	Support1	• Profile
		  ◦ (Agent)
		• Group
		  ◦ Support
		• Skills
		  ◦ Support
		  ◦ English
		• Team
		  ◦ TeamB

	Support2	• Profile
		  ◦ Support
		• Group
		  ◦ Support
		• Skills
		  ◦ Support
		  ◦ English
		  ◦ Spanish
		• Team
		  ◦ TeamB
		  ◦ TeamA

Queues	PrPQueueDoNotRemove	**Note**: This is a special queue for Provisioning Portal internal use only. Leave all parameters for this queue empty (default). This queue must be created for all tenants.
	Sales	• Primary groups ◦ Sales • Secondary groups

Item	Name	Details
		◦ Support • Wrap-up codes ◦ Sales call
	Support	• Primary groups ◦ Support • Secondary groups ◦ Sales • Wrap-up codes ◦ Support call
Release codes	Sales Lunch	
	Support Lunch	
Skills	Sales	
	Support	
	English	
	Spanish	
	RejectSkill	
Teams	TeamA	
	TeamB	
Wrap-up codes	Sales call	
	Support call	

Note

Avoid creating personnel with user ID prefix 'prp' in your source tenant. The reason is that the customer-submitted user ID with the tenant self-service portal is always prefixed with 'prp' for the first personnel creation for a new subscription. Therefore, by avoiding personnel ID prefix 'prp', you prevent future user ID conflicts at the tenant cloning process.

3. Confirm that the historical report URL of the new source tenant targets the same SQL Server Reporting Services (SSRS) host(s) where historical report templates will be automatically published for the new subscribers by the PS.TscSvc service.
   The configuration key is *...ReportsPublishUrl* in PS.TscSvc. For details, see Configure PS.TscSvc.

   When multiple SSRS hosts are involved, scale-out deployment feature is assumed. Scale-out deployment feature is available in the Enterprise Edition or higher.

   Consider the following example:

- Historical report URL in the legacy Administrator client tool for the new source tenant is *http://external_FQDN/Reports/Pages/Folder.aspx?ItemPath=/source1*
- ReportsPublishUrl in *tscCreateTenant.exe.config* for the PS.TscSvc service is *http://internal_FQDN (or NetBIOS name)/ReportServer*

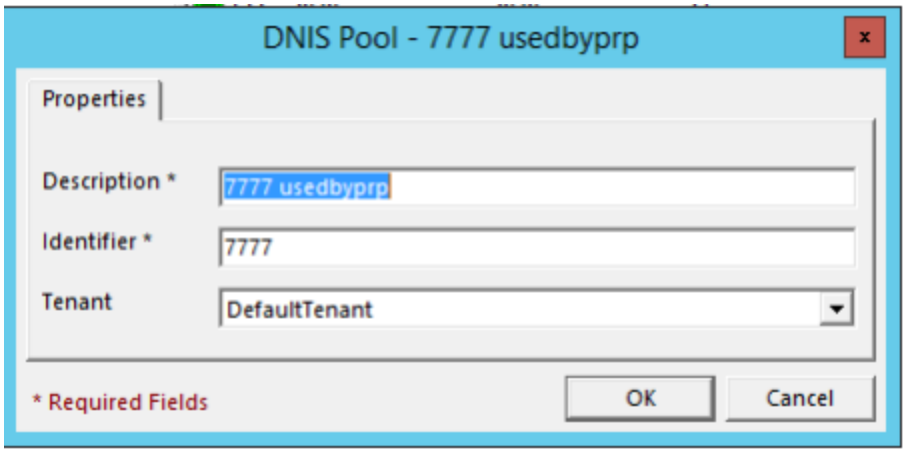
The report templates will be published by the PS.TscSvc service to the SQL Server Reporting Services running on the internal_FQDN for the new subscribers. In this case, external_FQDN of the historical report URL should target the same set of SQL Server Reporting Services host(s).

Provision DNIS

Provision new DNIS numbers with the legacy Administrator client for new tenant subscribers through Provisioning Portal.

Use the **Submit Bulk Numbers to the DNIS Pool** wizard to provision new DNIS numbers on CCSP.

> **Note**
> If you want to use DNIS numbers in DefaultTenant, you need to add in the description of the DNIS -> usedbyprp



Provision standard wave files

> **Note**
> Wave files must be in PCM 8kHz 16-bit mono. All other formats are unsupported.

CCSP ships certain wave files prerecorded in English that the Provisioning Portal reuses for the IVR service. These wave files are located on each VCS machine. The default path is *C:\Program Files (x86)\CosmoCom\Server Components\Scripts\AllWaves*.

Specifically the Provisioning Portal reuses wave files in the following subfolders:

- *Numbers*
- *Dates_and_Times*

To support a new language, duplicate the folder and replace the wave files in the appropriate language for at least the above subfolders. For example, to support Spanish for the IVR service with the Provisioning Portal:

1. Make a copy of the *AllWaves* folder from a VCS.
2. Replace the wave files for the new language.
3. On each VCS, deploy the folder with a new folder name such as *AllWavesSpanish* side-by-side with the *AllWaves* folder.

3: Deploy Provisioning Portal on the CCSP platform

This chapter contains the following information:

Create the database for Provisioning Portal

Provisioning Portaldatabase holds relevant Provisioning Portal configuration and states data. We recommend that you create this database on a dedicated SQL server named instance.

To create the database for Provisioning Portal:

1. Decide on which SQL server database engine instance to host the Provisioning Portal database.
2. From the database backup file called *PrPdb.bak* in the *SQL* folder of the Provisioning Portal package, restore the database as Provisioning Portal on the SQL server instance decided in step 1.
3. Follow the update steps in the next section (the included BAK file is not necessarily the latest).
4. Execute the stored procedure found in the Provisioning Portal database called *spPRP_Update_FirstTenantTemplate*.

> **Note**
> This stored procedure is only for the first template created during this deployment phase. This stored procedure updates sample data in the Provisioning Portal database tables so that the sample data becomes usable by the newly created source tenant.

```
use PrP;
go
exec dbo.spPRP_Update_FirstTenantTemplate
    @Tenant_ID=          -- The (tenant ID)       of the newly created source tenant
```

```
    @CCSPApplication_ID= -- The (application ID)  of the newly created source tenant
    @TenantName=         -- The (tenant name)      of the newly created source tenant
    @UPNSuffix=          -- The (tenant UPN Suffix) of the newly created source tenant
Go
```

Update the database for Provisioning Portal

Provisioning Portal database has incremental updates in the *SQL\PrPdb_Update* folder.

The current version of the SQL is found in the Provisioning Portal database using this command:

```
SELECT TOP 1 [VersionNum] FROM [PrP].[dbo].[verSchemaAudit] ORDER BY [VersionNum] DESC
```

The returned “VersionNum” number is where updates should continue from, for example, if currently it is 1013, then the SQL updates from 1014 onwards must be executed in sequence.

1023 will produce a warning about Cosmocall – the tenant creation process looks for Cosmocall on either a linked or local server instance, subsequently a warning will exist about whichever one does not exist, this can be ignored.

> **Note**
>
> The Config database SQL should be executed prior to the Provisioning Portal database, and the Historical database SQL afterwards.

Deploy PS.TscSvc

PS.TscSvc is a single instance backend service. Select a backend host or provision a new one for deploying PS.TscSvc that satisfy the following prerequisites:

Prerequisites:

- PS.TscSvc is responsible for tenant creation for Provisioning Portal. It depends on the following features of the CCSP Server Components and therefore the target host need to have permission to access to the servers where these features deployed:
  - Tenant Provisioning – this feature corresponds to the product called Provisioning Wizard in the Publisher.
  - MSR DB Connector – this feature corresponds to the product called MSR DB Connector in the Publisher.
  - Historical Reports of CosmoConsole – this feature corresponds to the product called Historical Reports in the Publisher.
  - CosmoAnalyst of CosmoConsole – this feature corresponds to the product called CosmoAnalyst Reports in the Publisher.
- Ensure that versions of all of the components under the above products in the Publisher are aligned with the target CCSP platform, otherwise, update them with the Publisher to the expected CCSP version.
- PS.TscSvc is responsible for publishing report templates for new tenants. It depends on SQL Server Reporting Services and therefore the target host is expected to have the feature deployed.
- PS.TscSvc requires SQL client the same way the CFM requires. Please refer to the *CCSP Prerequisites Guide* for instructions on how to setup SQL client components on the target machine for PS.TscSvc.
- PS.TscSvc requires SMTP service the same way the DTR requires. Please refer to the *CCSP Prerequisites Guide* for instructions on how to setup SMTP service on the target machine for PS.TscSvc.

To deploy PS.TscSvc:

1. Copy *PS.TscSvc* from Provisioning Portal package to *C:\Program Files (x86)\CosmoCom\PS.PrP*.
2. Execute *C:\Program Files (x86)\CosmoCom\PS.PrP\PS.TscSvc\reg_svc.bat* in admin command prompt window.
3. Configure automatic startup and service credentials for PS.TscSvc in Windows Services.

Configure PS.TscSvc

To configure PS.TscSvc:

1. Open *PS.TscSvc.exe.config* in *C:\Program Files (x86)\CosmoCom\PS.PrP\PS.TscSvc* for editing:

   a. For the following line in the file, replace the value with the public URL to Provisioning Portal:

   ```
   <add key="SendMail.template.portalUrl"
   value="http://dnccsp12.dn12.loc/PS.ProvisioningPortal" />
   ```

   b. For the following lines in the file, replace the values with your host for PS.TscSvc:

   ```
   <add key="SendMail.enabled" value="true" />
   <add key="SendMail.serverAddress" value="ccsp.lab.enghouse.loc" />
   <add key="SendMail.serverPort" value="25"/>
   <add key="SendMail.enableSsl" value="false"/>
   <add key="SendMail.senderEmailAddress" value="svcccsp@lab.enghouse.loc"/>
   <add key="SendMail.senderEmailAddress.base64" value="false"/>
   <add key="SendMail.providerRecipientEmailAddress" value="svcccsp@lab.enghouse.loc" />
   <add key="SendMail.senderPassword" value="xxxxxxx" />
   <add key="SendMail.senderPassword.base64" value="false" />
   ```

   c. For the following line in the file, replace server value with your SQL server instance for the Provisioning Portal database:

   ```
   <add name="PrpPortalDB" providerName="System.Data.SqlClient"
   connectionString="server=dnccsp12;database=PrP;Integrated Security=SSPI;"/>
   ```

   d. For the following lines in the file, replace with WebTop configurations if utilizing WebTop payments:

   ```
   <add key="Webtop.username" value="xxxxxxx" />
   <add key="Webtop.password" value="xxxxxxx" />
   <add key="Webtop.clientInitials" value="XXX" />
   <add key="Webtop.application" value="PrP" />
   <add key="Webtop.version" value="10.0.5" />
   <add key="Webtop.billingCycle" value="X" />
   <add key="Webtop.contactTypeId" value="X" />
   ```

   e. For the following line in the file, to enable auto SQL update, replace the value with your path to the Provisioning Portal database install SQL files:

   ```
   <add key ="SQLInstallFolder" value="C:\\Program Files (x86)\\CosmoCom\\Server
   Components\\PS.TscSvc\\sqlupdates"/> <!-- Populate with path to PrP DB install SQL
   ```

```
files if wanting to automate execution -->
```

2. Open *tscCreateTenant.exe.config* in *C:\Program Files (x86)\CosmoCom\PS.PrP\PS.TscSvc* for editing:

   a. For the following line in the file, replace the value with true to use integrated security:

   ```
   <add key="TscDbLib.UseIntegratedSecurity" value="false"/>
   ```

   b. For the following line in the file, replace UPN suffix value with your Windows domain for the CCSP platform:

   ```
   <add key="ConfigDefaults.UPNSuffixBase" value="dn12.loc"/>
   ```

   c. For the following line in the file, replace AdminIS value with your CCSP platform:

   ```
   <add key="ConfigDefaults.NewTenantConfiguration.SetupEnvironment.ConfigComParameters.
   AdminISAddress" value="10.116.80.11"/>
   ```

   d. For the following line in the file, replace value with your WebProvisioning network path: (physical path: *C:\Program Files (x86)\CosmoCom\Server Components\WebProvisioning\WebService*)

   ```
   <add key="ConfigDefaults.NewTenantConfiguration.SetupEnvironment.DBParameters.
   DBToolsNetworkFolder" value="\\CCSP1\DBScripts" />
   ```

   e. For the following line in the file, replace value with your MSR DB Connector network path: (physical path: *C:\Program Files (x86)\CosmoCom\Server Components\Setup\Messaging*)

   ```
   <add key="ConfigDefaults.NewTenantConfiguration.SetupEnvironment.DBParameters.
   MSRDBToolsNetworkFolder" value="\\CCSP1\Messaging" />
   ```

   f. For the following line in the file, replace value with your Tenant reports network path: (physical path: *C:\Program Files (x86)\CosmoCom\Server Components\CosmoReports\TenantReports*)

   ```
   <add key="ConfigDefaults.NewTenantConfiguration.ReportsPublishConfiguration.
   HistoricalReportsNetworkFolder" value="\\CCSP1\CosmoReports\TenantReports" />
   ```

   > **Note**
   >
   > The server which PS.TscSvc is installed must reach to the network folder which you used in sections d-f, also need to give permission **Change** to Messaging folder (MSR DB)
   >
   > WebProvisioning folder: *C:\Program Files (x86)\CosmoCom\Server Components\WebProvisioning\WebService*
   >
   > MSR DB folder: *C:\Program Files (x86)\CosmoCom\Server Components\Setup\Messaging*
   >
   > CosmoReports: *C:\Program Files (x86)\CosmoCom\Server Components\CosmoReports*

   g. For the following line in the file, replace value with your Analyst reports network path: (physical path: *C:\Program Files (x86)\CosmoCom\Server Components\CosmoReports\CosmoAnalyst*)

```
<add key="ConfigDefaults.NewTenantConfiguration.ReportsPublishConfiguration.
AnalystReportsNetworkFolder" value="\\CCSP1\CosmoReports\CosmoAnalyst" />
```

> **Note**
>
> If you are not using CosmoAnalyst you can leave the value empty:
>
> ```
> <add
> key="ConfigDefaults.NewTenantConfiguration.ReportsPublishConfiguration.
> AnalystReportsNetworkFolder" value="" />
> ```

h. For each Reports Server that you have in your platform add this key replace the host part of the URL in the value with the FQDN or NetBIOS name with the SQL Server Reporting Services host:

```
<add
key="ConfigDefaults.NewTenantConfiguration.ReportsPublishConfiguration.ReportsPublishU
rl.x" value="http://dnccsp12.dn12.loc/reportserver"/>

Each x replace with order number list staring from 1, example- for 2 report servers:

<add
key="ConfigDefaults.NewTenantConfiguration.ReportsPublishConfiguration.ReportsPublishU
rl.1" value="http://dnccsp12.dn12.loc/reportserver"/>

<add
key="ConfigDefaults.NewTenantConfiguration.ReportsPublishConfiguration.ReportsPublishU
rl.2" value="http://dnccsp13.dn12.loc/reportserver"/>
```

i. For the following two lines in the file, replace user and password values with your SQL server reporting services:

```
<add
key="ConfigDefaults.NewTenantConfiguration.ReportsPublishConfiguration.ReportPublishUs
er" value="sa"/>

<add key="ConfigDefaults.NewTenantConfiguration.ReportsPublishConfiguration.
ReportPublishUserPassword" value="sa123"/>
```

> **Note**
>
> All the reports servers need to run with the same user and password (#e).
>
> The host part of the URL refers to the host that has reporting services installed. We strongly recommend the Enterprise Edition of reporting services. By the SQL server reporting services design, published reports are not synchronized automatically between reporting services instances that are not at least the Enterprise Edition. When you have multiple reporting services instances but not at least the Enterprise Edition, consult with your ASG Project Manager.

j. For the following line in the file, replace server value with the SQL server database instance for Provisioning Portal database:

```
<add name="PrpPortalDb" providerName="System.Data.SqlClient"
connectionString="server=DNCCSP12;database=PrP;Integrated Security=SSPI;"/>
```

Configure syslog service for PS.TscSvc

For the host instance deployed with PS.TscSvc, add the following lines to *syslog.conf* in *D:\LogRotationScript*, assuming that log rotation scripts were deployed through Log Collector on *D:* drive.

```
ADMIN_POINT.*             __SYSLOG_ROOT__\admin_point.txt
TP.*                  __SYSLOG_ROOT__\touch_point.txt
CIS.*                  __SYSLOG_ROOT__\cis.txt
mail.*                 __SYSLOG_ROOT__\prptcsvc.txt
```

When log rotation scripts were not deployed through Log Collector, add lines similar to the below to *syslog.conf* in *C:\Program Files (x86)\Common Files\CosmoCom\CosmoCall Universe*.

```
ADMIN_POINT.*             D:\syslogd\admin_point.txt
TP.*                  D:\syslogd\touch_point.txt
CIS.*                  D:\syslogd\cis.txt
mail.*                 D:\syslogd\prptcsvc.txt
```

> **Note**
>
> Since release 15.4 tenant-creation utilizes a tool called NLog to write logs, this is used for SysLog, local file logs (in "*logs*" sub-folder), and DB logging. Logging levels are managed in NLog.config's rules as defined here, and no longer managed in *PS.TscSvc.exe.config*. The local log file will separate between the service and creation executables, the syslogs for both use the "mail" facility by default. Log content has also changed since previous releases.

Deploy Provisioning Portal

Provisioning Portal is a front-end service. Provisioning Portal supports N+1. It depends on CCSP ConfigCOM and Provisioning Portal database connectivity to run. In addition, Provisioning Portal reuses CCSP UI Admin framework, so we recommend that the target machines have CCSP UI Admin installed.

Select at least two frontend hosts or provision new ones for deploying Provisioning Portal that satisfy the following prerequisites:

- Provisioning Portal requires SQL client in the same way as the CFM. Please refer to the *CCSP Prerequisites Guide* for instructions on how to setup SQL client components on the target machine for Provisioning Portal.
- Provisioning Portal depends on CCSP ConfigCOM and Provisioning Portal reuses the CCSP UI Admin framework, so we recommend that the target machines have CCSP UI Admin installed. CCSP UI Admin implements the CCSP UI Admin framework.

To deploy Provisioning Portal:

1. Copy the *Prp* folder from the Provisioning Portal package to *C:\Program Files (x86)\CosmoCom\PS.PrP*.
2. Create an application pool on IIS with the following attributes. Accept the other defaults.

Attribute	Value
Name	Prp
Enabled 32-bit Applications	True
Identity	Your CCSP service account
Managed Pipeline Mode	Integrated
.Net CLR Version	V4.0
Queue Length	4000

3. Create a virtual directory in IIS called *Prp* with the physical path pointing to *C:\Program Files (x86)\CosmoCom\PS.PrP\Prp*.
4. Add MIME type *.woff2 application/font-woff2* to the *Prp* virtual directory.
5. Convert the *Prp* virtual directory to an application with *Prp* as the application pool.

Configure Provisioning Portal

To configure the Provisioning Portal:

1. Navigate to *C:\Program Files (x86)\CosmoCom\PS.PrP\Prp\config.js.*
2. Open *GeneralParam.xml* in a text editor.
3. For the following line in the file, replace the Prp Path with Provisioning Portal service FQDN (fully qualify domain name) server address.

   ```
   window.PrP.Path = 'https://ccsp1735a.pj16.loc/PS.ProvisioningPortal';
   ```

4. If TTS (text to speech) service is installed and configured in the server, you can assign the service URL to the window.Prp.TTSUrl property.

   ```
   window.PrP.TTSUrl = "https://ccsp1735a.pj16.loc/IBMTTS/IBMTTS";
   ```

5. Assign the Config Portal URL to the Window.PrP.ConfigPortal_URL property.

   ```
   Window.PrP.ConfigPortal_URL = 'https://ccsp1735a.pj16.loc/PS.ConfigPortal/';
   ```

6. If you want to enable OIDC (SSO) single sign-on configuration on your CCSP system, you need to configure the following section.

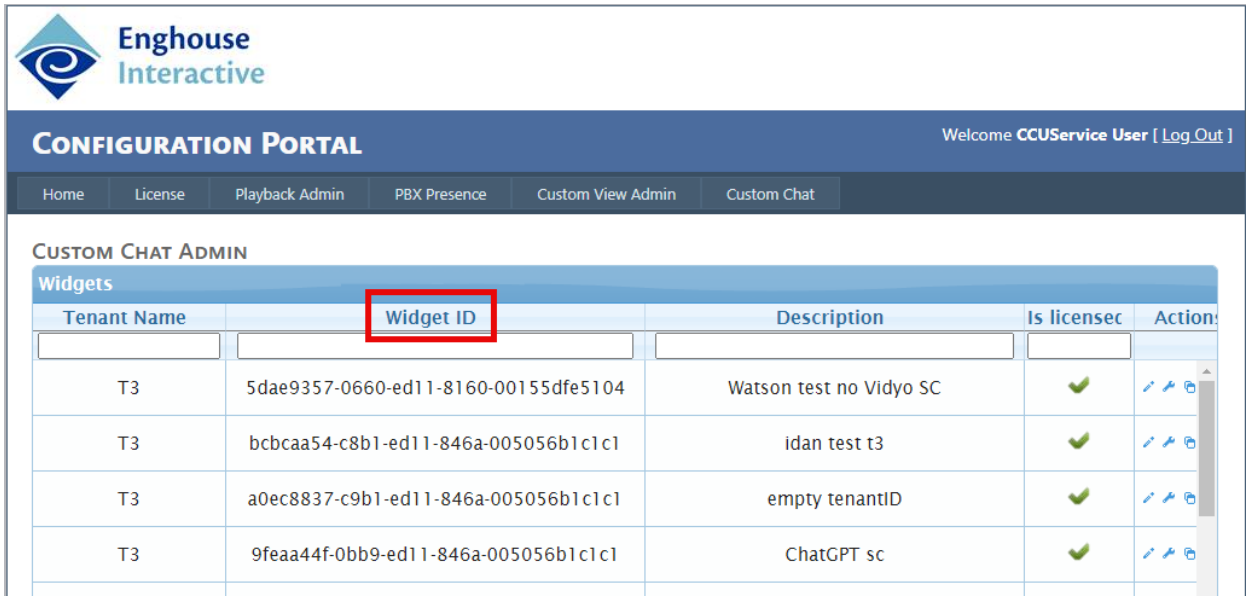
   ```
   window.PrP.AuthServerURL = "https://ccsp1735a.pj16.loc/AuthServer/";
   window.PrP.ClientID = "29903b3b25f44884b934f750568257a1";
   window.PrP.ClientName = "PrP";
   ```

   It is mandatory to insert the correct CCSP Auth Server address in AuthServerURL property. The default Provisioning Portal Client ID and Client Name default to the value specified above.

   To verify the Client ID and Name for SSO configuration, navigate to Admin, under Custom SSO Applications you should be able to verify the particulars of all registered instances. For more information, see Appendix G – SSO Configuration.

7. For post survey preview, a WidgetId sample from ConfigPortal is required. Eventually it will take the styling of the survey and implement it with the survey data. If you would like to have it configured for Provisioning Portal, insert the widget ID in the following property.

   ```
   window.PrP.PostSurveySampleWidgetId = 'bcbcaa54-c8b1-ed11-846a-005056b1c1c1';
   ```


Enghouse
Interactive
CONFIGURATION PORTAL
Welcome CCUService User [Log Out]
Home
License
Playback Admin
PBX Presence
Custom View Admin
Custom Chat
CUSTOM CHAT ADMIN
Widgets
Tenant Name
Widget ID
Description
Is licensed
Actions
T3
5dae9357-0660-ed11-8160-00155dfe5104
Watson test no Vidyo SC
T3
bcbcaa54-c8b1-ed11-846a-005056b1c1c1
idan test t3
T3
a0ec8837-c9b1-ed11-846a-005056b1c1c1
empty tenantID
T3
9feaa44f-0bb9-ed11-846a-005056b1c1c1
ChatGPT sc

Configure syslog service for Provisioning Portal

Provisioning Portal uses CCSP syslog service for debug logging. For host instances deployed with Provisioning Portal, add the following lines to *syslog.conf* in *D:\LogRotationScript*, assuming log rotation scripts were deployed through Log Collector on *D:* drive:

```
ADMIN_POINT.*      __SYSLOG_ROOT__\admin_point.txt
TP.*               __SYSLOG_ROOT__\touch_point.txt
CIS.*              __SYSLOG_ROOT__\cis.txt
mail.*             __SYSLOG_ROOT__\prptcsvc.txt
news.*             __SYSLOG_ROOT__\prptcscmd.txt
```

When log rotation scripts were not deployed through Log Collector, add lines similar to the below to *syslog.conf* in *C:\Program Files (x86)\Common Files\CosmoCom\CosmoCall Universe*:

```
ADMIN_POINT.*       D:\syslogd\admin_point.txt
TP.*                D:\syslogd\touch_point.txt
CIS.*               D:\syslogd\cis.txt
mail.*              D:\syslogd\prptcsvc.txt
news.*              D:\syslogd\prptcscmd.txt
```

Update XMLInterpreter

To update XMLInterpreter:

1. Export a production copy of *XMLInterpreter.vbs* for local editing, using legacy Administrator client tool.
2. Search for the following items in *XMLInterpreter_PrP.vbs* and add the changes to the local *XMLInterpreter.vbs* copy:
   - **ActiveWaitTime**
   - **Lang**
   - **MultiTimeZone**
   - **RegExTest**
3. When standard wave files were provisioned for additional IVR language support, look for the following section in the local *XMLInterpreter.vbs* copy.

```
 if (calldata.option("lang")="Sp/") then
     Call LogEvent(" SetPlayLang : GetPathToWaves", "Spanish" )
     GetPathToWaves = replace(GetPathToWaves,"\AllWaves\","\AllWavesSpanish\")
     pathToNumbers  = GetPathToWaves & "Numbers\"
     pathToMath     = GetPathToWaves & "Math\"
     pathToDates_and_times  = GetPathToWaves & "Dates_and_times\"
     pathToAlphabet  = GetPathToWaves & "Alphabet\"
     pathToCommon    = GetPathToWaves & "Common\"
     pathToMiscWords  = GetPathToWaves & "MiscWords\"
     pathToCurrency  = GetPathToWaves & "Currency\"
 ElseIf (calldata.option("lang")="Fr/") then
     Call LogEvent(" SetPlayLang : GetPathToWaves", "French" )
     GetPathToWaves = replace(GetPathToWaves,"\AllWaves\","\AllWavesFrench\")
     pathToNumbers  = GetPathToWaves & "Numbers\"
     pathToMath     = GetPathToWaves & "Math\"
     pathToDates_and_times   = GetPathToWaves & "Dates_and_times\"
     pathToAlphabet  = GetPathToWaves & "Alphabet\"
     pathToCommon    = GetPathToWaves & "Common\"
     pathToMiscWords = GetPathToWaves & "MiscWords\"
     pathToCurrency  = GetPathToWaves & "Currency\"
Else
     Call LogEvent("SetPlayLang : GetPathToWaves", "English " )
     GetPathToWaves = replace(GetPathToWaves,"\AllWaves\","\AllWaves\")
     pathToNumbers  = GetPathToWaves & "Numbers\"
     pathToMath     = GetPathToWaves & "Math\"
     pathToDates_and_times   = GetPathToWaves & "Dates_and_times\"
```

```
    pathToAlphabet  = GetPathToWaves & "Alphabet\"
    pathToCommon    = GetPathToWaves & "Common\"
    pathToMiscWords = GetPathToWaves & "MiscWords\"
    pathToCurrency  = GetPathToWaves & "Currency\"
End if
```

Add a relevant else case for your language code that points to the relevant wave files folder name you've created.

Below is an example of adding Cantonese support to the Provisioning Portal.

```
 if (calldata.option("lang")="Sp/") then
    Call LogEvent(" SetPlayLang : GetPathToWaves", "Spanish" )
    GetPathToWaves = replace(GetPathToWaves,"\AllWaves\","\AllWavesSpanish\")
    ......
 ElseIf (calldata.option("lang")="Fr/") then
    Call LogEvent(" SetPlayLang : GetPathToWaves", "French" )
    GetPathToWaves = replace(GetPathToWaves,"\AllWaves\","\AllWavesFrench\")
    ......
 ElseIf (calldata.option("lang")="Ca/") then
    Call LogEvent(" SetPlayLang : GetPathToWaves", "Cantonese" )
    GetPathToWaves = replace(GetPathToWaves,"\AllWaves\","\AllWavesCantonese\")
    pathToNumbers  = GetPathToWaves & "Numbers\"
    pathToMath     = GetPathToWaves & "Math\"
    pathToDates_and_times   = GetPathToWaves & "Dates_and_times\"
    pathToAlphabet  = GetPathToWaves & "Alphabet\"
    pathToCommon    = GetPathToWaves & "Common\"
    pathToMiscWords = GetPathToWaves & "MiscWords\"
    pathToCurrency  = GetPathToWaves & "Currency\"
Else
    Call LogEvent("SetPlayLang : GetPathToWaves", "English " )
    GetPathToWaves = replace(GetPathToWaves,"\AllWaves\","\AllWaves\")
    ......
End if
```

4. Upload the updated local *XMLInterpreter.vbs* copy using legacy Administrator client tool and restart VCS instances.

Enable auto SQL update

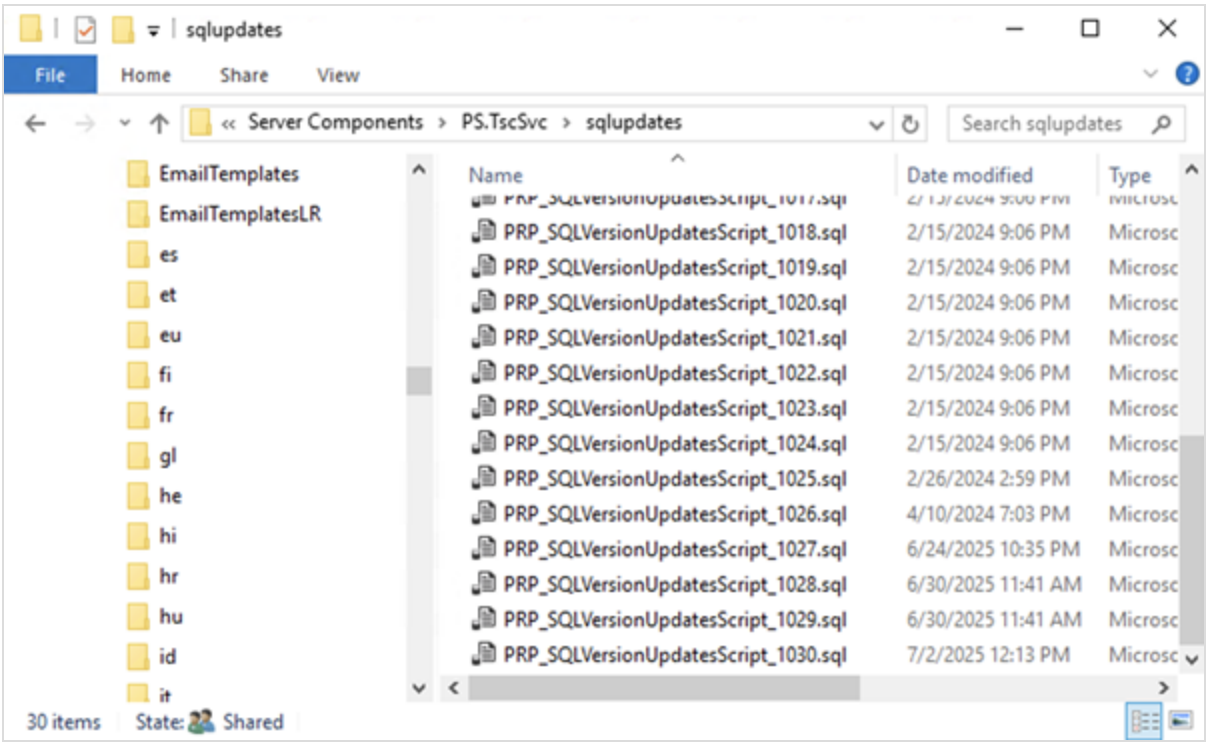
The Provisioning Portal Tenant Creation service (Ps.TscSvc) can now automatically perform the required SQL updates on start-up, if the `SQLInstallFolder` configuration is set to the folder containing the to Provisioning Portal database install SQL files and the SQL version update scripts with the corresponding version number are available in the specified path under *SQLInstallFolder*.

If the SQLInstallFolder configuration is set, the Provisioning Portal service retrieves the current version of the database from there, looks for each SQL file in the configured folder to determine if there is a newer version, and then executes it accordingly.

To enable this feature, set the following configuration in the *PS.Tsc.Svc.exe.config* file:

```
<add key ="SQLInstallFolder" value="C:\\Program Files (x86)\\CosmoCom\\Server
Components\\PS.TscSvc\\sqlupdates"/> <!-- Populate with path to PrP DB install SQL files if
wanting to automate execution -->
```

You must also ensure that the Provisioning Portal SQL version update scripts with the corresponding version number are available in the *sqlupdates* folder, in the path specified under *SQLInstallFolder*.



Troubleshooting

Startup SQL check – Provisioning Portal database is up to date

When the PS.TscSvc service starts, the following log is observed.

```
2025-08-07 13:10:46.8596 | [5] | INFO | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |  |
SQL Update automation active, checking SQL path:C:\\Program Files (x86)\\CosmoCom\\Server
Components\\PS.TscSvc\\sqlupdates

2025-08-07 13:10:46.9846 | [5] | DEBUG | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |  |
Current PrP version from the PrP DB is 1030
```

All SQL transactional activities are logged to the PS.TscSvc log file. If you encounter issues with the SQL update, refer to this log file for more information.

Startup SQL check – Provisioning Portal database is outdated, an updated is applied

The following log provides an example message indicating that an update is being applied to an existing version of the database.

```
2025-08-07 18:48:28.8942 | [6] | INFO | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |  |
SQL Update automation active, checking SQL path:C:\\Program Files (x86)\\CosmoCom\\Server
Components\\PS.TscSvc\\sqlupdates
2025-08-07 18:48:28.9192 | [6] | DEBUG | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |  |
Current PrP version from the PrP DB is 1030
2025-08-07 18:48:28.9332 | [6] | INFO | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |  |
Executing SQL PrP version 1031
```

This log shows a successful upgrade of the Provisioning Portal database from version 1030 to 1031.

Startup SQL check – Error when upgrade the database with a SQL file

The following log includes examples of errors encountered during the upgrade process when a SQL file is applied to the Provisioning Portal database.

```
2025-08-07 20:20:50.1884 | [6] | INFO | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |  |
SQL Update automation active, checking SQL path:C:\\Program Files (x86)\\CosmoCom\\Server
Components\\PS.TscSvc\\sqlupdates

2025-08-07 20:20:50.2094 | [6] | DEBUG | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |  |
Current PrP version from the PrP DB is 1030

2025-08-07 20:20:50.2204 | [6] | INFO | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |  |
Executing SQL PrP version 1031

2025-08-07 20:20:50.2204 | [6] | ERROR | PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate |
System.Data.SqlClient.SqlException

Incorrect syntax near '!'.

   at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean
breakConnection, Action`1 wrapCloseInAction)

   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean
breakConnection, Action`1 wrapCloseInAction)

   at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj,
Boolean callerHasConnectionLock, Boolean asyncClose)

   at System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler,
SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject
stateObj, Boolean& dataReady)

   at System.Data.SqlClient.SqlCommand.RunExecuteNonQueryTds(String methodName, Boolean async,
Int32 timeout, Boolean asyncWrite)
```

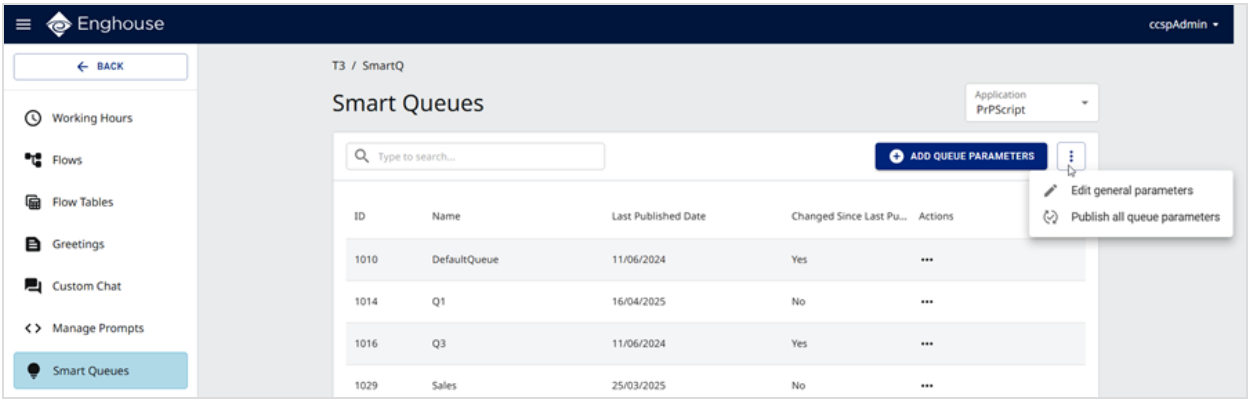
```
   at System.Data.SqlClient.SqlCommand.InternalExecuteNonQuery(TaskCompletionSource`1
completion, String methodName, Boolean sendToPipe, Int32 timeout, Boolean& usedCache, Boolean
asyncWrite, Boolean inRetry)
   at System.Data.SqlClient.SqlCommand.ExecuteNonQuery()
   at PS.TscProcessPrpRequestLib.SQLUpdate.CheckUpdate(String connstr, String path) in
E:\Git\ip-ccsp-
asg\Projects\PS.ProvisioningPortal\PS.TenantSelfCreation\PS.TscProcessPrpRequestLib\SQLUpdate.
cs:line 85 | Error processing SQL Update automation:Incorrect syntax near '!'., SQL:
/******************************************************
*            SQL CHANGES BETWEEN VERSIONS
*            PLEASE Follow the Instructions below
*
*
******************************************************/
-- This is a dummy SQL to test auto update on restart

Forcing error!
2025-08-07 20:20:50.2554 | [6] | WARN | PS.TscSvc.TscSvc.OnStart |  | 1 SQL Update errors
encountered:
Error processing SQL Update automation:Incorrect syntax near '!'., SQL:
/******************************************************
*            SQL CHANGES BETWEEN VERSIONS
*            PLEASE Follow the Instructions below
*
*
******************************************************/
-- This is a dummy SQL to test auto update on restart

Forcing error!
2025-08-07 20:20:50.2554 | [6] | INFO | PS.TscSvc.TscSvc.OnStart |  | OnStart Done.
```

As indicated in the log file, the database upgrade process failed due to an issue with a SQL file. The log contains details about the error and the procedures that were followed.

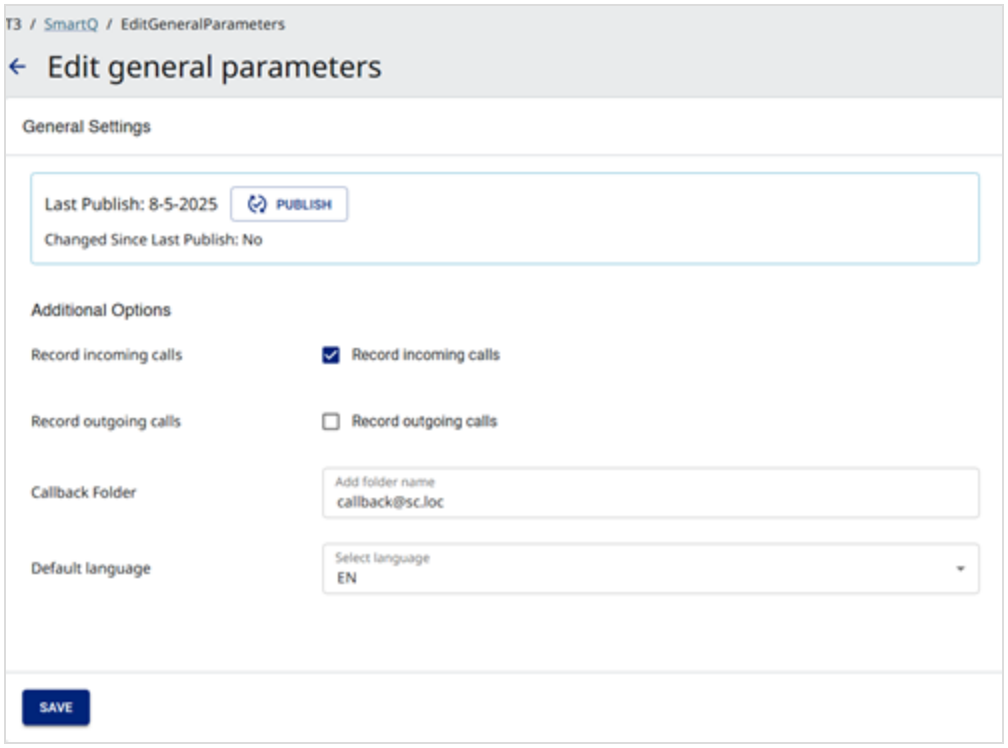
4: Finalize Provisioning Portal deployment

Perform the following steps to finalize configurations needed for the newly provisioned source tenant for the Provisioning Portal deployment:

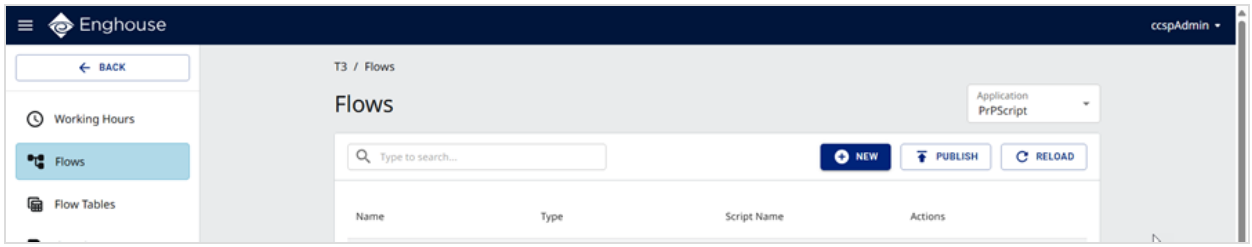
1. Log on to the Provisioning Portal as the administrator of the newly created source tenant.
2. Switch to the **Smart Queues** module.



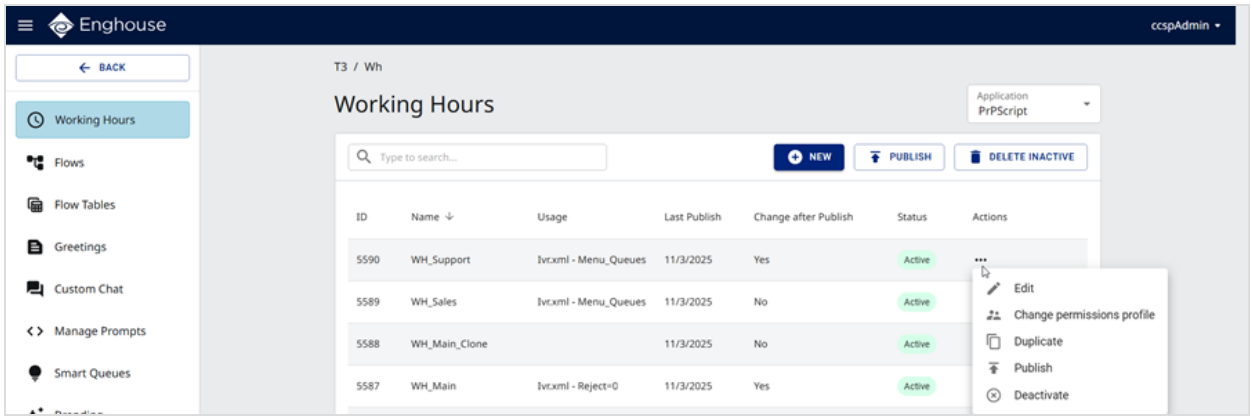
3. Click ⋮ and then click ✎ **Edit general parameters**. The **Edit General Parameters** dialog appears.



4. Configure the items listed on the screen and click **Save**.
5. Click ⋮ and then click **Publish all queue parameters.**
6. Select **PrPScript** in the **Application** list. **PrPScript** is the application created earlier for the newly provisioned source tenant.
7. Add **Queue Parameters** for the **Support** queue, and the **Sales** queue. For information on how to add queue parameters for Smart Queues, see the *Smart Queues* section of the *Provisioning Portal Administrator Guide.*
8. Switch to the **Flows** module.



9. Click RELOAD. In the **Reload** dialog, click **Reload**.
10. Switch to the **Working Hours** module.



11. Publish all the working hours placeholders by clicking PUBLISH or publish each one individually by clicking ••• and then clicking **Publish**.

5: Deploy Provisioning Portal Helper

This chapter contains the following information:

Deploy Provisioning Portal Helper

Provisioning Portal Helper is a package of IVR scripts and API's that provide some abilities of Provisioning Portal to be mange via phone call.

- Recording Prompts and Publish/Deactivate from IVR
- Open/Close WorkingHours from IVR

To deploy Provisioning Portal Helper:

1. Execute the following scripts on the Provisioning Portal database from Provisioning Portal package *PrPHelper\SQL* folder :
    - *spPRPHelper_GetPrompt.sql*
    - *spPRPHelper_IVRPromptAction.sql*
    - *spPRPHelper_IsSpecialDateExist.sql*
    - *spPRPHelper_IVRGetWHData.sql*
    - *spPRPHelper_IVRWHAction.sql*
    - *spPRPHelper_WHforSC.sql*
2. Copy *PS.PrPHelperWebInterface* from Provisioning Portal package Provisioning PortalHelper to *C:\Program Files (x86)\CosmoCom\PS.PrP* on each web server where IVRScripts are created.
3. Create application pool on IIS with the following attributes. Accept other defaults.

Attribute	Value
Name	PS.PrPHelperAppPool
Enabled 32-bit Applications	True
Identity	Your CCSP service account
Managed Pipeline Mode	Integrated
.Net CLR Version	V4.0
Queue Length	4000

4. Create VD in IIS called *PS.PrPHelperWebInterface* with physical path pointing to *C:\Program Files (x86)\CosmoCom\PS.PrP\PS.PrPHelperWebInterface*.
5. Convert *PS.PrPHelperWebInterface* VD to application with *PS.PrPHelperAppPool* as the application pool.
6. Copy IVR applications – *RecordingPrompts* and *WorkingHoursActivation* to your IVR folder and create application per script in the tenant that will use those scripts.

> **Note**
>
> For the IVR to play Greetings it requires the Provisioning Portal Helper(s) to have a location to store recordings that is accessible to all instances of Provisioning Portal Helper (for example, network share or ADFS replication), and a URL to retrieve the recordings by all VCS. These are set in the *web.config*:
>
> - **RecordingsPath** - the location to store recording files that is configured and accessible by all Provisioning Portal Helper instances (whether single shared folder or replicated to appear the same)
> - **RecordingsURL** – the HTTP (VCS does not support HTTPS) URL that the VCS can access the recordings from (pointing to **RecordingsPath** location)

Configure RecordingPrompts IVR application

1. Edit pincode table: pinCode and TenantId
2. Edit **Path** block – URL of *RecordingPrompts* VD
3. Edit **Wavs** block – URL of Provisioning Portal publish prompts folder:
   http://DNS/IVRScripts/$calldata.tenantid$/Wavs/
4. Edit **promptsFolder** block – physical path where Provisioning Portal Helper WebInterface deployed for saving temporary prompts:
   C:\Program Files (x86)\CosmoCom\PS.PrP\PS.PrPHelperWebInterface\prompts
5. Edit **isFileReady** block – URL of *PrPHelperWeb.asmx*:
   http://DNCCSP12.DN12.LOC/PS.PrPHelperWebInterface/PrPHelperWeb.asmx
6. Edit **IVRPromptAction** block – URL of *PrPHelperWeb.asmx*:
   http://DNCCSP12.DN12.LOC/PS.PrPHelperWebInterface/PrPHelperWeb.asmx

Configure WorkingHoursActivation IVR application

1. Edit pincode table: pinCode and TenantId
2. Edit **Path** block – URL of *WorkingHoursActivation* VD
3. Edit **IsSpecialDateExist** block – URL of *PrPHelperWeb.asmx*:
   http://DNCCSP12.DN12.LOC/PS.PrPHelperWebInterface/PrPHelperWeb.asmx
4. Edit **WHAction** block – URL of *PrPHelperWeb.asmx*:
   http://DNCCSP12.DN12.LOC/PS.PrPHelperWebInterface/PrPHelperWeb.asmx

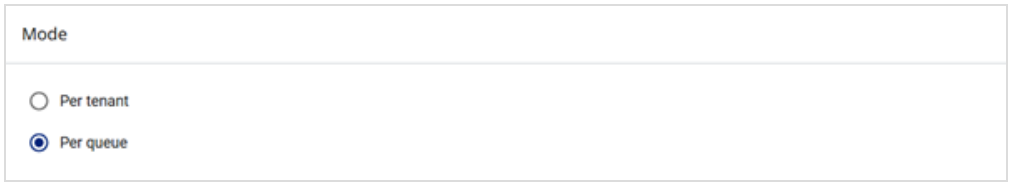
6: Appendixes

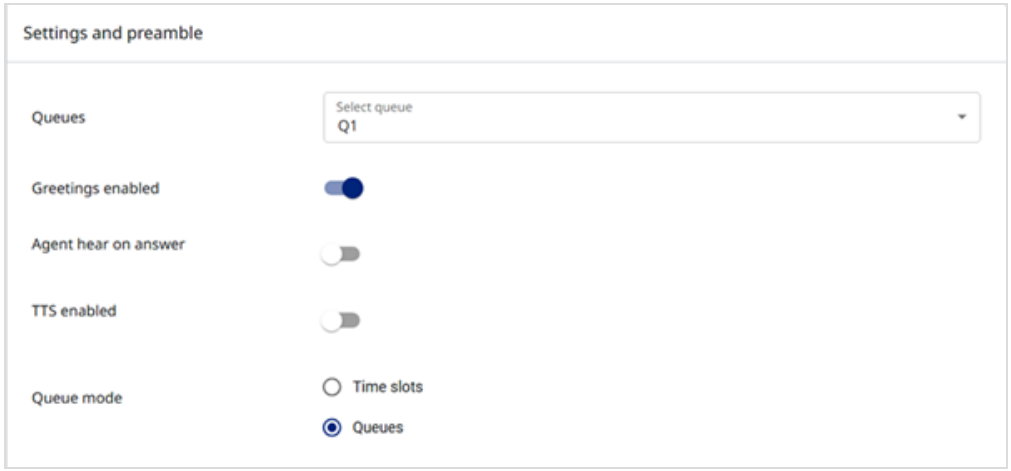
This section contains the following appendixes:

Appendix A – Greetings

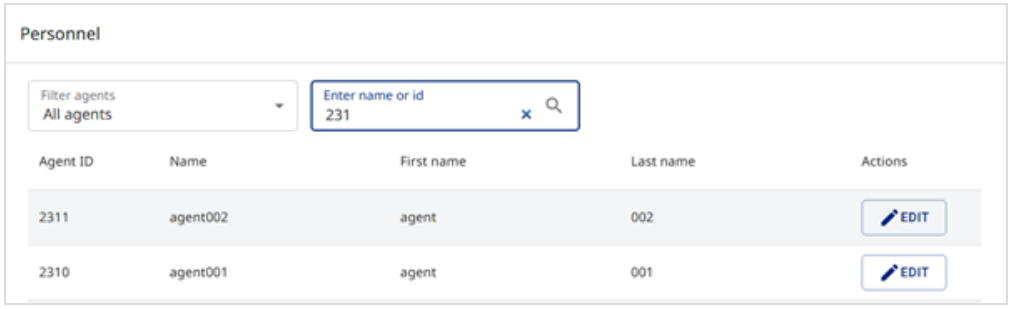
Configuration

- Modes – Per Tenant = configure for all queues the same, Per Queue = each queue can be configured separately to the others.





- Queues – if mode is Per Queue this drop-down list of queues is displayed to selected which queue to configure
- Greetings enabled – whether Greetings enabled (applied Per Tenant or per selected queue depending in selected Mode)
- Agent hear on answer – whether the greeting audio heard by callers is also heard by the agent
- TTS enabled – whether to use the TTS service to play the greeting audio if none recorded.
- The subsequent text boxes are the TTS locale and content. The content has these items dynamically replaced:
    - {label} – the current Timeslot label
    - {firstName} – the agent's first name
    - {lastName} – the agent's last name
    - {queueName} – the queue of the call
- Queue Mode – whether the queue's greeting audio is per Timeslot or not. When in Timeslot mode you may define timespans of the day for different greeting audio (these may be added and edited). Each mode allows optional "preamble" audio – common audio played before the agent-specific audio. When in Chrome with HTTPS you may record audio from Provisioning Portal using the record icon. Existing audio may be downloaded for review.
- Agents list – list of agents whose Group contains the selected queue as a Primary Queue. The list may be filtered by those with / without recordings, or by text. The headers in this table may be clicked to sort by them too. Each agent may be edited to download / record their greeting audio (recordings based on whether Timeslots enabled – if disabled it will be a single recording, otherwise it will be 1 per Timeslot)

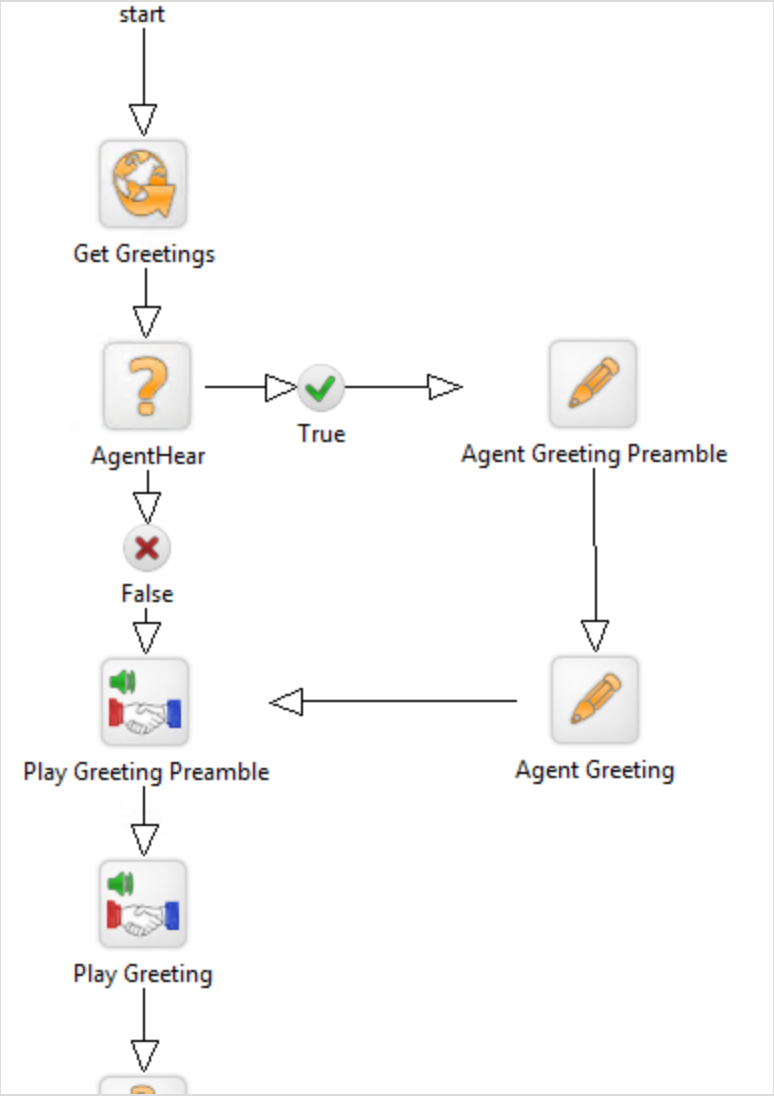


Deployment

The Designer script requires the PathHelperAPI Optional Parameter to be set, usually wherever the Path Optional Parameters is currently set, to the URL of the Provisioning Portal Helper, for example:

http://ccsp1.pj16.loc/PS.PrPHelperWebInterface/

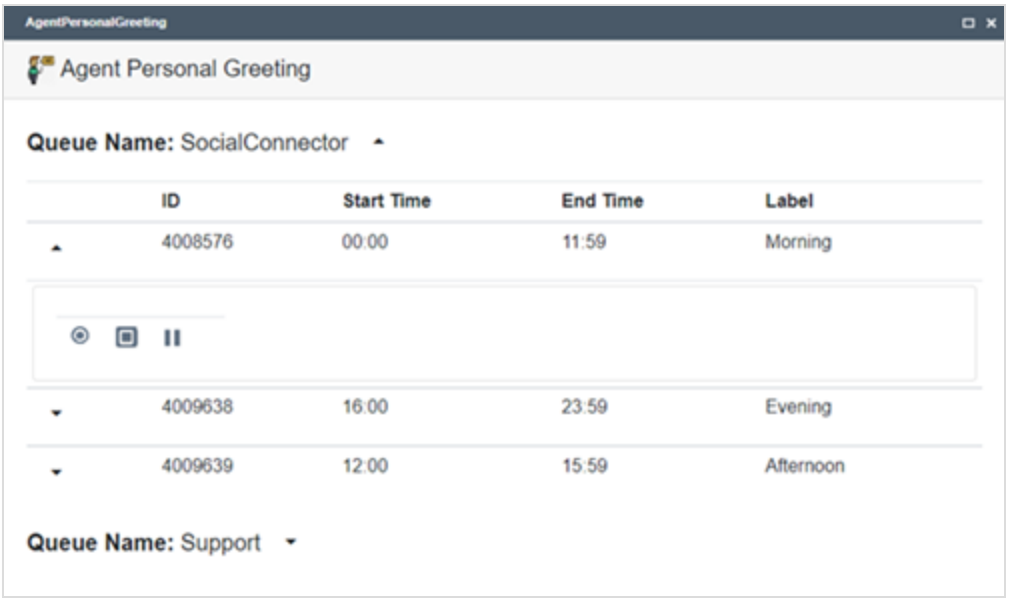
Copy this section from the template Call Answered script to retrieve the recordings and play on answer:



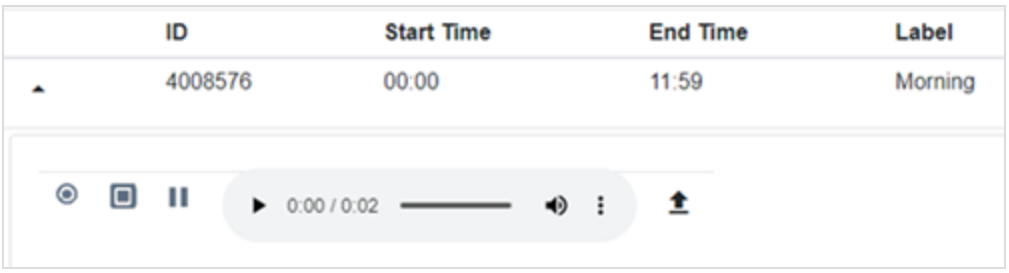
Install the **AgentPersonalGreeting** gadget to allow agent's to record their own greetings from CCSP UI:

- Default State = Float
- Allowed Instances = 1
- Target URL (HTML) = App/gadgets/AgentPersonalGreeting/agentPersonalGreeting.html
- Icon = Images\icon.png
- PrPHelperURL = URL of Provisioning Portal Helper API for Greetings, for example: *https://ccsp1.pj16.loc/PS.PrPHelperWebInterface/PersonalGreeting.asmx/*
- Other settings nay be adjusted to personal preference

When the agent selects the gadget they will see the enabled queues that their Group has as Primary Queues (unless tenant is set to Per Tenant mode, then they will see a single **All** queue). These queues may be expanded to record / play their greeting, when in Timeslot mode it will be a recording row per Timeslot.



When pressing the recording icon (requires Chrome and HTTPS) the recording can subsequently be reviewed and uploaded:



Example log scenarios

Below follows some example log excerpts from Designer and Provisioning Portal Helper of Greetings usage during calls.

Example of Designer when no preamble recording, but exists personal greeting (missing audio is treated as empty TTS and ignored, otherwise the returned URL is played):

```
08/07/2021 17:55:36 :: WebServicesCall :: XMLRequest
parameters=<tenantID>2</tenantID><agentID>4827</agentID><password>T35tP455w0rd</password><queu
eName>DefaultQueue</queueName><firstName>sc</firstName><lastName>3</lastName>

08/07/2021 17:55:36 :: WebServicesCall :: XMLRequest with parameters=<?xml version="1.0"
encoding="UTF-8"?> <soap:Envelope xmlns="http://CCSP.PS.PrPHelperWebInterface/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <soap:Body>     <GetGreetingRecordingIVR
xmlns="http://CCSP.PS.PrPHelperWebInterface/
"><tenantID>2</tenantID><agentID>4827</agentID><password>T35tP455w0rd</password><queueName>Def
```

```
aultQueue</queueName><firstName>sc</firstName><lastName>3</lastName></GetGreetingRecordingIVR>
</soap:Body></soap:Envelope>

08/07/2021 17:55:36 :: WebServicesCall :: Waiting a maximum of 15 seconds for response...

08/07/2021 17:55:36 :: WebServicesCall :: Got response in less than 1 second

08/07/2021 17:55:36 :: WebServicesCall :: XMLHTTP.readyState: 4

08/07/2021 17:55:36 :: WebServicesCall :: XMLResponse=<?xml version="1.0"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><GetGreetingRecordingIVRResponse
xmlns="http://CCSP.PS.PrPHelperWebInterface/"><GetGreetingRecordingIVRResult><Url>
http://localhost/Greetings/2_2.wav
</Url></GetGreetingRecordingIVRResult></GetGreetingRecordingIVRResponse></soap:Body></soap:Env
elope>

08/07/2021 17:55:36 :: setValues :: INFO:
'WSResult.GetGreetingRecordingIVRResponse.xmlns'='http://CCSP.PS.PrPHelperWebInterface/'

08/07/2021 17:55:36 :: WriteDictionary :: WSResult.GetGreetingRecordingIVRResponse.xmlns =
'http://CCSP.PS.PrPHelperWebInterface/'

08/07/2021 17:55:36 :: readDictionary :: calldata.option("pathHelperAPI") =
http://localhost/PS.PrPHelperWebInterface/

08/07/2021 17:55:36 :: readDictionary :: calldata.tenantid = 2

08/07/2021 17:55:36 :: readDictionary :: calldata.answeringagentid = 4827

08/07/2021 17:55:36 :: readDictionary :: calldata.callsetname = DefaultQueue

08/07/2021 17:55:36 :: readDictionary :: calldata.answeringagentfirstname = sc

08/07/2021 17:55:36 :: readDictionary :: calldata.answeringagentlastname = 3

08/07/2021 17:55:36 :: setValues :: INFO:
'WSResult.GetGreetingRecordingIVRResponse.GetGreetingRecordingIVRResult.Url'='http://localhost
/Greetings/2_2.wav'

08/07/2021 17:55:36 :: WriteDictionary ::
WSResult.GetGreetingRecordingIVRResponse.GetGreetingRecordingIVRResult.Url =
'http://localhost/Greetings/2_2.wav'

08/07/2021 17:55:36 :: CallXMLWebServices ::
WebServicesCallResult=<GetGreetingRecordingIVRResponse
xmlns="http://CCSP.PS.PrPHelperWebInterface/"><GetGreetingRecordingIVRResult><Url>
http://localhost/Greetings/2_
2.wav</Url></GetGreetingRecordingIVRResult></GetGreetingRecordingIVRResponse>

08/07/2021 17:55:36 :: ParseNode :: Action Result Received:  NOP:

08/07/2021 17:55:36 :: ParseNode :: goto

08/07/2021 17:55:36 :: ActOnNode111 :: goto

08/07/2021 17:55:36 :: CallXMLgoto :: Go to node #Play Greeting PreAmble

08/07/2021 17:55:36 :: CallXMLgoto :: No variables specified, sending all

08/07/2021 17:55:36 :: CallXMLgoto :: INFO: Use default 'get' method

08/07/2021 17:55:36 :: readDictionary :: calldata.option("pathHelperAPI") =
http://localhost/PS.PrPHelperWebInterface/

08/07/2021 17:55:36 :: readDictionary :: calldata.tenantid = 2

08/07/2021 17:55:36 :: readDictionary :: calldata.answeringagentid = 4827

08/07/2021 17:55:36 :: readDictionary :: calldata.callsetname = DefaultQueue

08/07/2021 17:55:36 :: readDictionary :: calldata.answeringagentfirstname = sc

08/07/2021 17:55:36 :: readDictionary :: calldata.answeringagentlastname = 3
```

```
08/07/2021 17:55:36 :: ParseNode :: Action Result Received:  NAV:#Play Greeting PreAmble

08/07/2021 17:55:36 :: XMLInterpreter :: nBlocksInterpreted=2 Maximum allowed number =300

08/07/2021 17:55:36 :: XMLInterpreter :: Beginning CallXML Application at #Play Greeting
PreAmble

08/07/2021 17:55:36 :: LoadNextNode :: Looking for node=Play Greeting PreAmble in document

08/07/2021 17:55:36 :: ParseBlock :: INFO: Parsing block 'Play Greeting PreAmble NodeType
:<block label="Play Greeting PreAmble" GoID="" xPos=" 300" yPos=" 531"><playGreeting
greeting="$wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.PreambleUrl
$" agentGreeting="" async="0" comment="PrP test Preamble"/><goto value="#Play
Greeting"/></block>

08/07/2021 17:55:36 :: ParseNode :: playGreeting

08/07/2021 17:55:36 :: ActOnNode111 :: playGreeting

08/07/2021 17:55:36 :: CallXMLplayGreeting :: Enter

08/07/2021 17:55:36 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.PreambleUrl =

08/07/2021 17:55:36 :: CallXMLplayGreeting :: !!! agentGreeting =

08/07/2021 17:55:36 :: CallXMLplayGreeting :: !!! strGreeting =

08/07/2021 17:55:36 :: CallXMLplayGreeting :: Async mode is: 0

08/07/2021 17:55:36 :: CallXMLplayGreeting :: caller greeting Play text!!!

08/07/2021 17:55:36 :: CallXMLplayGreeting :: !!! synchronous playback, waiting for completion

08/07/2021 17:55:36 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.PreambleUrl =

08/07/2021 17:55:36 :: ParseNode :: Action Result Received:  NOP:

08/07/2021 17:55:36 :: ParseNode :: goto

08/07/2021 17:55:36 :: ActOnNode111 :: goto

08/07/2021 17:55:36 :: CallXMLgoto :: Go to node #Play Greeting

08/07/2021 17:55:36 :: CallXMLgoto :: No variables specified, sending all

08/07/2021 17:55:36 :: CallXMLgoto :: INFO: Use default 'get' method

08/07/2021 17:55:36 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.PreambleUrl =

08/07/2021 17:55:36 :: ParseNode :: Action Result Received:  NAV:#Play Greeting

08/07/2021 17:55:36 :: XMLInterpreter :: nBlocksInterpreted=3 Maximum allowed number =300

08/07/2021 17:55:36 :: XMLInterpreter :: Beginning CallXML Application at #Play Greeting

08/07/2021 17:55:36 :: LoadNextNode :: Looking for node=Play Greeting in document

08/07/2021 17:55:36 :: ParseBlock :: INFO: Parsing block 'Play Greeting NodeType :<block
label="Play Greeting" GoID="" xPos=" 300" yPos=" 653"><playGreeting
greeting="$wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.Url$"
agentGreeting="" async="0" comment="PrP test URL"/><goto/></block>

08/07/2021 17:55:36 :: ParseNode :: playGreeting

08/07/2021 17:55:36 :: ActOnNode111 :: playGreeting

08/07/2021 17:55:36 :: CallXMLplayGreeting :: Enter

08/07/2021 17:55:36 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.Url =
http://localhost/Greetings/2_2.wav

08/07/2021 17:55:36 :: CallXMLplayGreeting :: !!! agentGreeting =
```

```
08/07/2021 17:55:36 :: CallXMLplayGreeting :: !!! strGreeting = http://localhost/Greetings/2_2.wav

08/07/2021 17:55:36 :: CallXMLplayGreeting :: Async mode is: 0

08/07/2021 17:55:36 :: CallXMLplayGreeting :: caller greeting Play .wav!!!

08/07/2021 17:55:36 :: CallXMLplayGreeting :: !!! synchronous playback, waiting for completion

08/07/2021 17:55:39 :: readDictionary :: wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.Url = http://localhost/Greetings/2_2.wav
```

Example of this in API Logs:

```
2021-07-08 17:55:35.1850 | [12] | DEBUG | PrPHelperWebInterface.PersonalGreeting.GetGreetings |  | Request for greetings from tenant 2 queue -1 agent 4827 with password T35tP455w0rd

2021-07-08 17:55:36.1270 | [12] | DEBUG | PrPHelperWebInterface.PersonalGreeting.GetGreetingRecordingIVR |  | Request for greeting IVR recording from tenant 2 agent 4827 with password T35tP455w0rd and queueName DefaultQueue and firstName sc and lastName 3

2021-07-08 17:55:36.4641 | [12] | DEBUG | PrPHelperWebInterface.PersonalGreeting.GetGreetingRecordingIVR |  | Queue DefaultQueue disabled=False

2021-07-08 17:55:36.4641 | [12] | DEBUG | PrPHelperWebInterface.PersonalGreeting.WriteFile | | Request for greetings IVR recording from tenant 2 agent 4827 has recording already on file-system
```

Example when agent has recordings enabled but no recording/TTS:

```
08/07/2021 18:05:43 :: WebServicesCall :: XMLRequest without parameters=<?xml version="1.0" encoding="UTF-8"?> <soap:Envelope xmlns="http://CCSP.PS.PrPHelperWebInterface/" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <soap:Body>     <GetGreetingRecordingIVR xmlns="http://CCSP.PS.PrPHelperWebInterface/">%1</GetGreetingRecordingIVR> </soap:Body></soap:Envelope>

08/07/2021 18:05:43 :: readDictionary :: calldata.tenantid = 2

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentid = 3502

08/07/2021 18:05:43 :: readDictionary :: calldata.callsetname = DefaultQueue

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentfirstname = Social

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentlastname = Agent2

08/07/2021 18:05:43 :: WebServicesCall :: XMLRequest parameters=<tenantID>2</tenantID><agentID>3502</agentID><password>T35tP455w0rd</password><queueName>DefaultQueue</queueName><firstName>Social</firstName><lastName>Agent2</lastName>

08/07/2021 18:05:43 :: WebServicesCall :: XMLRequest with parameters=<?xml version="1.0" encoding="UTF-8"?> <soap:Envelope xmlns="http://CCSP.PS.PrPHelperWebInterface/" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <soap:Body>     <GetGreetingRecordingIVR xmlns="http://CCSP.PS.PrPHelperWebInterface/"><tenantID>2</tenantID><agentID>3502</agentID><password>T35tP455w0rd</password><queueName>DefaultQueue</queueName><firstName>Social</firstName><lastName>Agent2</lastName></GetGreetingRecordingIVR> </soap:Body></soap:Envelope>

08/07/2021 18:05:43 :: WebServicesCall :: Waiting a maximum of 15 seconds for response...
```

```
08/07/2021 18:05:43 :: WebServicesCall :: Got response in less than 1 second

08/07/2021 18:05:43 :: WebServicesCall :: XMLHTTP.readyState: 4

08/07/2021 18:05:43 :: WebServicesCall :: XMLResponse=<?xml version="1.0"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><GetGreetingRecordingIVRResponse
xmlns="http://CCSP.PS.PrPHelperWebInterface/"/></soap:Body></soap:Envelope>


08/07/2021 18:05:43 :: setValues :: INFO:
'WSResult.GetGreetingRecordingIVRResponse.xmlns'='http://CCSP.PS.PrPHelperWebInterface/'

08/07/2021 18:05:43 :: WriteDictionary :: WSResult.GetGreetingRecordingIVRResponse.xmlns =
'http://CCSP.PS.PrPHelperWebInterface/'

08/07/2021 18:05:43 :: readDictionary :: calldata.option("pathHelperAPI") =
http://localhost/PS.PrPHelperWebInterface/

08/07/2021 18:05:43 :: readDictionary :: calldata.tenantid = 2

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentid = 3502

08/07/2021 18:05:43 :: readDictionary :: calldata.callsetname = DefaultQueue

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentfirstname = Social

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentlastname = Agent2

08/07/2021 18:05:43 :: setValues :: INFO: 'WSResult.GetGreetingRecordingIVRResponse'=''

08/07/2021 18:05:43 :: WriteDictionary :: WSResult.GetGreetingRecordingIVRResponse = ''

08/07/2021 18:05:43 :: CallXMLWebServices ::
WebServicesCallResult=<GetGreetingRecordingIVRResponse
xmlns="http://CCSP.PS.PrPHelperWebInterface/"/>

08/07/2021 18:05:43 :: ParseNode :: Action Result Received:  NOP:

08/07/2021 18:05:43 :: ParseNode :: goto

08/07/2021 18:05:43 :: ActOnNode111 :: goto

08/07/2021 18:05:43 :: CallXMLgoto :: Go to node #Play Greeting PreAmble

08/07/2021 18:05:43 :: CallXMLgoto :: No variables specified, sending all

08/07/2021 18:05:43 :: CallXMLgoto :: INFO: Use default 'get' method

08/07/2021 18:05:43 :: readDictionary :: calldata.option("pathHelperAPI") =
http://localhost/PS.PrPHelperWebInterface/

08/07/2021 18:05:43 :: readDictionary :: calldata.tenantid = 2

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentid = 3502

08/07/2021 18:05:43 :: readDictionary :: calldata.callsetname = DefaultQueue

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentfirstname = Social

08/07/2021 18:05:43 :: readDictionary :: calldata.answeringagentlastname = Agent2

08/07/2021 18:05:43 :: ParseNode :: Action Result Received:  NAV:#Play Greeting PreAmble

08/07/2021 18:05:43 :: XMLInterpreter :: nBlocksInterpreted=2 Maximum allowed number =300

08/07/2021 18:05:43 :: XMLInterpreter :: Beginning CallXML Application at #Play Greeting
PreAmble

08/07/2021 18:05:43 :: LoadNextNode :: Looking for node=Play Greeting PreAmble in document

08/07/2021 18:05:43 :: ParseBlock :: INFO: Parsing block 'Play Greeting PreAmble NodeType
:<block label="Play Greeting PreAmble" GoID="" xPos=" 300" yPos=" 531"><playGreeting
greeting="$wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.PreambleUrl
```

```
$" agentGreeting="" async="0" comment="PrP test Preamble"/><goto value="#Play
Greeting"/></block>

08/07/2021 18:05:43 :: ParseNode :: playGreeting

08/07/2021 18:05:43 :: ActOnNode111 :: playGreeting

08/07/2021 18:05:43 :: CallXMLplayGreeting :: Enter

08/07/2021 18:05:43 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.PreambleUrl =

08/07/2021 18:05:43 :: CallXMLplayGreeting :: !!! agentGreeting =

08/07/2021 18:05:43 :: CallXMLplayGreeting :: !!! strGreeting =

08/07/2021 18:05:43 :: CallXMLplayGreeting :: Async mode is: 0

08/07/2021 18:05:43 :: CallXMLplayGreeting :: caller greeting Play text!!!

08/07/2021 18:05:43 :: CallXMLplayGreeting :: !!! synchronous playback, waiting for completion

08/07/2021 18:05:43 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.PreambleUrl =

08/07/2021 18:05:43 :: ParseNode :: Action Result Received:  NOP:

08/07/2021 18:05:43 :: ParseNode :: goto

08/07/2021 18:05:43 :: ActOnNode111 :: goto

08/07/2021 18:05:43 :: CallXMLgoto :: Go to node #Play Greeting

08/07/2021 18:05:43 :: CallXMLgoto :: No variables specified, sending all

08/07/2021 18:05:43 :: CallXMLgoto :: INFO: Use default 'get' method

08/07/2021 18:05:43 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.PreambleUrl =

08/07/2021 18:05:43 :: ParseNode :: Action Result Received:  NAV:#Play Greeting

08/07/2021 18:05:43 :: XMLInterpreter :: nBlocksInterpreted=3 Maximum allowed number =300

08/07/2021 18:05:43 :: XMLInterpreter :: Beginning CallXML Application at #Play Greeting

08/07/2021 18:05:43 :: LoadNextNode :: Looking for node=Play Greeting in document

08/07/2021 18:05:43 :: ParseBlock :: INFO: Parsing block 'Play Greeting NodeType :<block
label="Play Greeting" GoID="" xPos=" 300" yPos=" 653"><playGreeting
greeting="$wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.Url$"
agentGreeting="" async="0" comment="PrP test URL"/><goto/></block>

08/07/2021 18:05:43 :: ParseNode :: playGreeting

08/07/2021 18:05:43 :: ActOnNode111 :: playGreeting

08/07/2021 18:05:43 :: CallXMLplayGreeting :: Enter

08/07/2021 18:05:43 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.Url =

08/07/2021 18:05:43 :: CallXMLplayGreeting :: !!! agentGreeting =

08/07/2021 18:05:43 :: CallXMLplayGreeting :: !!! strGreeting =

08/07/2021 18:05:43 :: CallXMLplayGreeting :: Async mode is: 0

08/07/2021 18:05:43 :: CallXMLplayGreeting :: caller greeting Play text!!!

08/07/2021 18:05:43 :: CallXMLplayGreeting :: !!! synchronous playback, waiting for completion

08/07/2021 18:05:43 :: readDictionary ::
wsresult.getgreetingrecordingivrresponse.getgreetingrecordingivrresult.Url =
```

Example of this in API Logs:

```
2021-07-08 18:05:43.4082 | [27] | DEBUG |
PrPHelperWebInterface.PersonalGreeting.GetGreetingRecordingIVR |  | Request for greeting IVR
recording from tenant 2 agent 3502 with password T35tP455w0rd and queueName DefaultQueue and
firstName Social and lastName Agent2

2021-07-08 18:05:43.5182 | [27] | DEBUG |
PrPHelperWebInterface.PersonalGreeting.GetGreetingRecordingIVR |  | Queue DefaultQueue
disabled=False

2021-07-08 18:05:43.5182 | [27] | WARN |
PrPHelperWebInterface.PersonalGreeting.GetGreetingRecordingIVR | System.Exception

Queue DefaultQueue enabled but no recording found for agent 3502

   at PrPHelperWebInterface.PersonalGreeting.GetGreetingRecordingIVR(Int32 tenantID, Int32
agentID, String password, String queueName, String firstName, String lastName) in
E:\PerForce\CosmoCom\PS\Projects\PS.ProvisioningPortal\PrPHelper\PrPHelperWebInterface\PrPHelp
erWebInterface\PersonalGreeting.asmx.cs:line 270 | An error occured in SqlDataProvider.Execute
- spPRP_GetGreetingRecordingIVR.
```

Example of this in API Logs when no audio just because disabled:

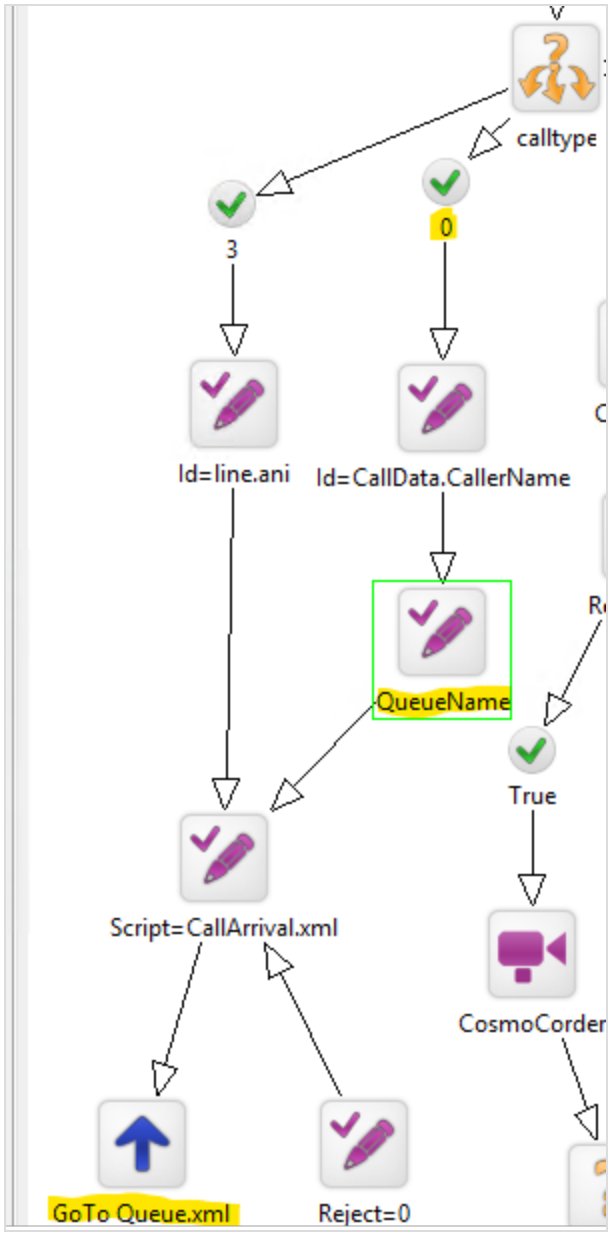
```
2021-07-08 18:11:04.4036 | [30] | DEBUG |
PrPHelperWebInterface.PersonalGreeting.GetGreetingRecordingIVR |  | Request for greeting IVR
recording from tenant 2 agent 3502 with password T35tP455w0rd and queueName DefaultQueue and
firstName Social and lastName Agent2

2021-07-08 18:11:04.4566 | [30] | DEBUG |
PrPHelperWebInterface.PersonalGreeting.GetGreetingRecordingIVR |  | Queue DefaultQueue
disabled=True
```

Appendix B – Callbacks

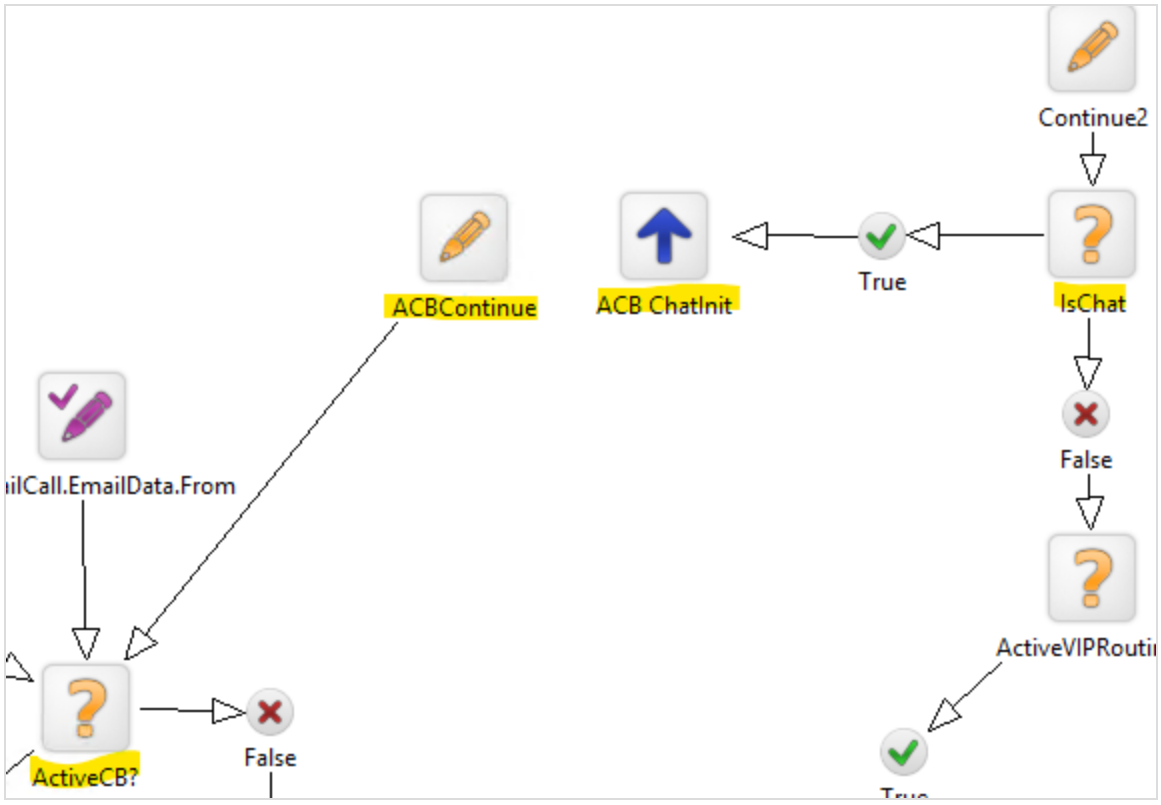
For Abandoned Callbacks (ACB) and Callback Deletions (CBD), the Designer script requires the "PathHelperAPI" Optional Parameter to be set, usually wherever the "Path" Optional Parameters is currently set, to the URL of the Provisioning Portal Helper, for example:
http://ccsp1.pj16.loc/PS.PrPHelperWebInterface/

Script changes required to support ACB and CBD

1. For chats set the QueueName Optional Parameter to the active queue ($calldata.callsetname$) in **CallArrival** to facilitate loading the correct queue configuration:

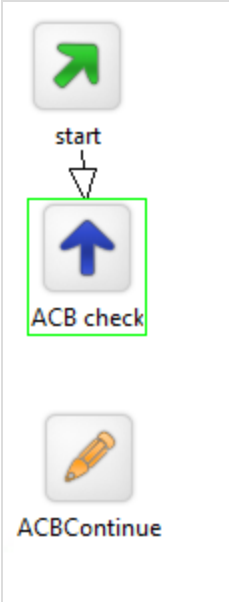


2. On the **CallArrival** script adjust the “Continue2” flow to initialize chats (“IsChat” is $cclevent.calltype$=0) calling “ACB ChatInit” and add the return-point “ACBContinue” to continue:



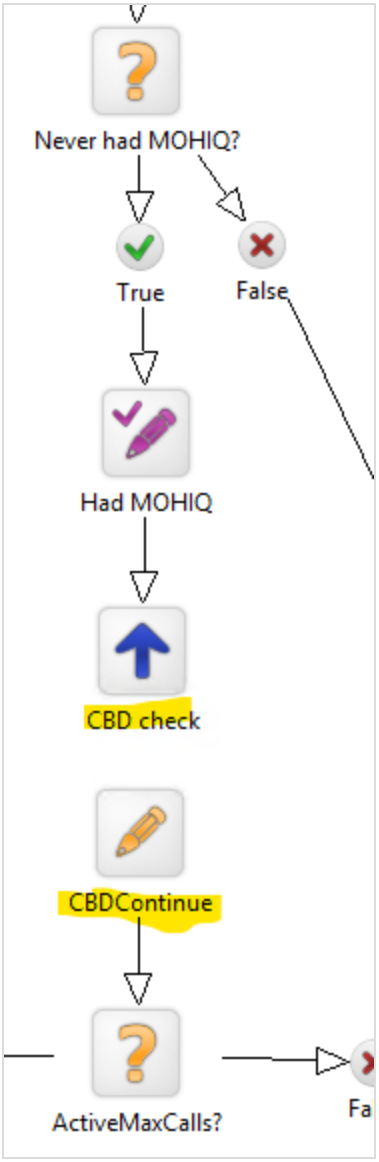
“ACB ChatInit” has the destination as:
$calldata.option("Path")$$calldata.option("ScriptCB")$#ACBChatInit

3. On the **Call Termination** script call “ACB Check” and the return-point “ACBContinue” to continue:



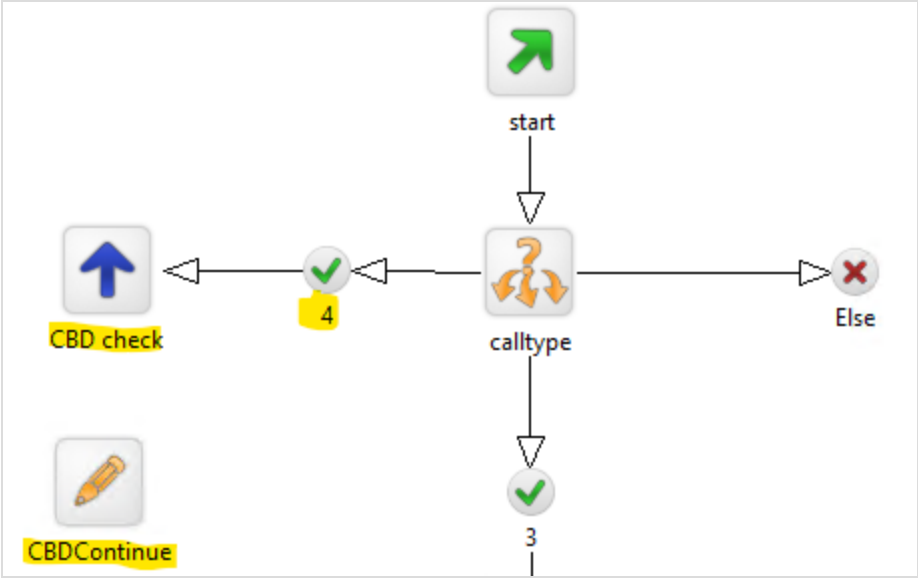
“ACB Check” has the destination as:
$calldata.option("Path")$$calldata.option("ScriptCB")$#ACBCheck

4. On the **Music On Hold In Queue** script call “CBD Check” and the return-point “CBDContinue” on 1st time:



“CBD Check” has the destination as:
$calldata.option("Path")$$calldata.option("ScriptCB")$#CBDInQCheck

5. On the **In Queue Timeout** script call “CBD Check” and the return-point “CBDContinue” for Callback call-types:



“CBD Check” has the destination as:
$calldata.option("Path")$$calldata.option("ScriptCB")$#CBDInQTimeoutCheck

Appendix C – SMS

From release 15.4 SmartQ support for SMS has been available via Social Connector:

- No Agents Available - for notifying the call-centre calls have arrived at an unattended queue
- SMS – for informing callers who timeout in-queue that there are alternative ways to contact.

Social Connector should be configured to include an SMS provider, and the “instance ID” of the configuration is set in the SmartQ SMS configurations. Additionally the SMS.XML file should be edited:

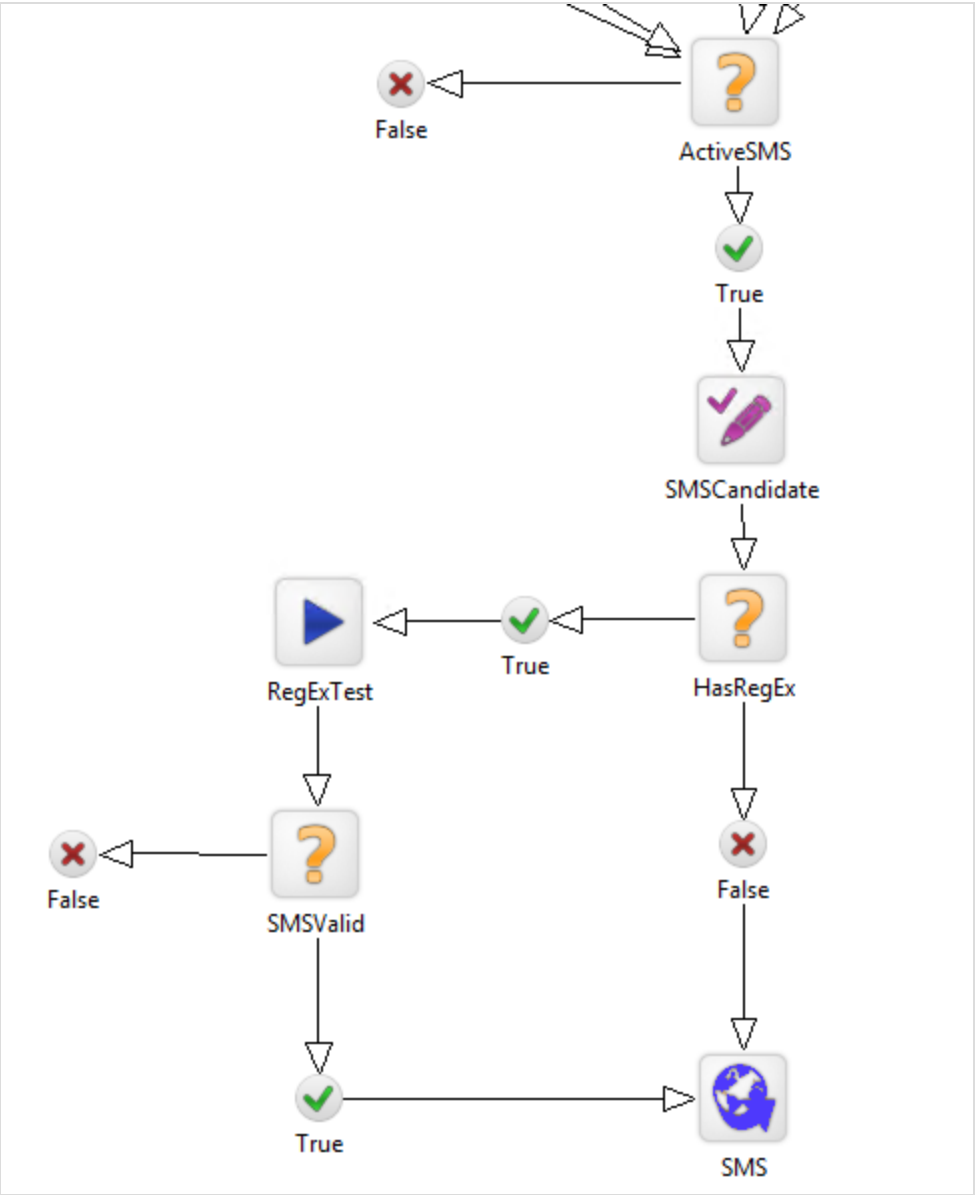
- pluginID – the ID of the plugin that the SMS integration will be performed by
- subchannel – the subchannel of the plugin (usually “sms”) as some integrations accept multiple content types.

> **Note**
>
> Theoretically, it is not limited to SMS as any Social Connector channel type that supports proactive notifications may be utilized, but this is untested and should only be considered with consultation from Enghouse Interactive. Most channels’ vendors do not permit sending messages to recipients with no recent history of direct interaction on that sending channel.

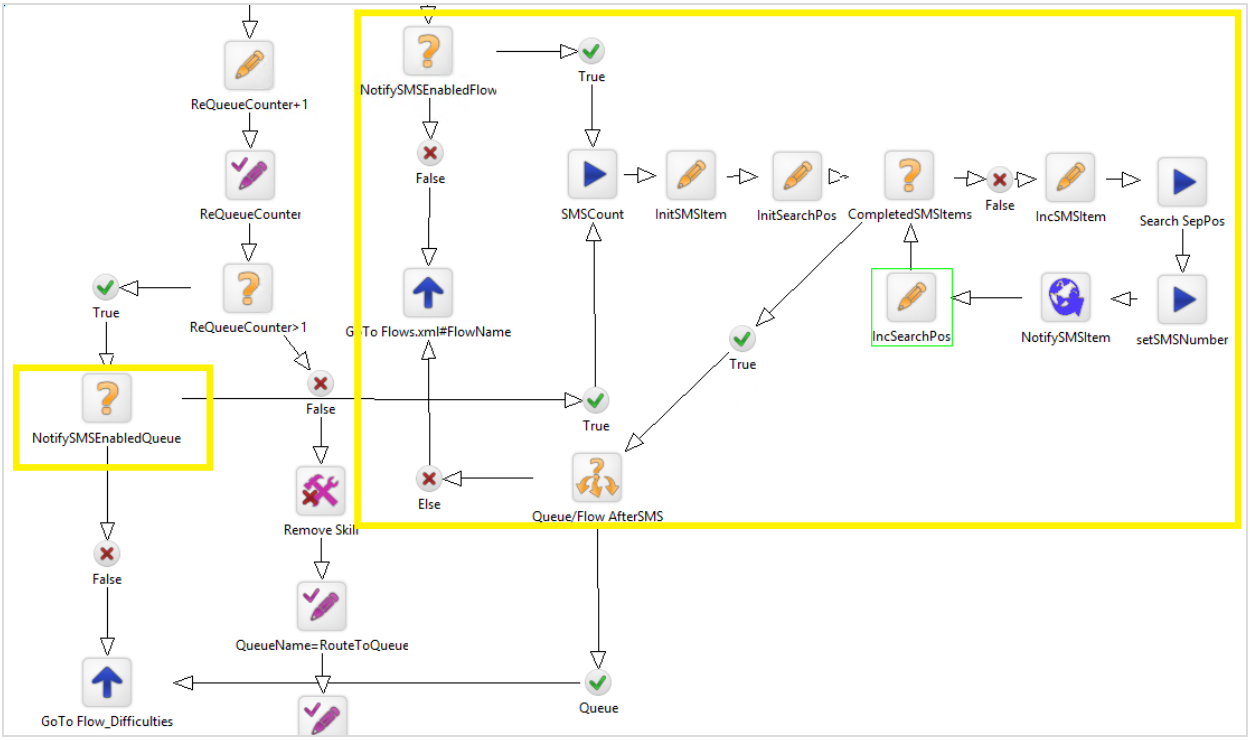
Additional to the SMS.XML modification and SmartQ configuration, the Provisioning Portal script has to know where to send the request, that is, the location of the Social Connector ServiceAPI – this should be set (in CallArrival and IVR scripts) using the Optional Parameter “SMSURL”:

https://<FQDN>:18440/serviceApi/messages/sendProactiveMessage

In **Queue Timeout** script change:



Call Rejected script change (no agents available):



Appendix D - Providers

Providers were introduced in 15.4 and are primarily a way of grouping tenant creation templates for different providers that will use the WebTop integration for payments:

- Each provider has their own Terms and Conditions page
- Each provider is assigned a set of tenant creation templates, rather than standard creation which uses a single-set of templates
- Each provider has some CSS styling control over the presentation of their tenant-creation page
- Each tenant created by a provider template may utilize CSS styling to hide Provisioning Portal menu options for their created tenants
- Tenant creation is separated into 3 major steps instead of 1:
    - Obtain tenant creation details, pass to WebTop
    - WebTop emails customer with link to resume payment
    - WebTop notifies Provisioning Portal payment successful and tenant creation can continue

Add a provider

The Provisioning Portal database now has a new “prpTenantProviders” table, that allows you to set up aspects of a new provider:

- Name – the provider name
- BillingScript – a C# script that will manage the integration through WebTop – may be shared between providers, but may be separated if different options are required between providers
- Emailxxx – manages the provider emails templates / subjects for successful and failed tenant creations to customers/providers
- TNC – the HTML of the Terms and Conditions

Once the provider is added, a few new fields were added to the “prpTenantTemplates” table to link templates to providers (unlinked templates are displayed on legacy non-provider tenant creation):

- BillingCode – the WebTop billing code of the template, to ensure correct product/price is billed
- ProviderID – the ID of the provider from “prpTenantProviders” – NULL is a legacy non-provider template

Style a provider

Tenant creation

The tenant creation URL for Providers is similar to legacy, taking this format (x is the Provider ID from “prpTenantProviders”):

https://<FQDN>/PS.ProvisioningPortal/tclrwizard?providerId=x

Per-Provider styling is limited but is possible by adding/adapting a CSS file for the Provider located here (again x is the Provider ID from “prpTenantProviders”):

PS.ProvisioningPortal/App/Content/Styles/providerx.css

> **Note**
>
> Every provider should have this corresponding CSS file, simplest approach is to copy the supplied example provider1.css to suit the new provider ID then tailor it.

Modify the Provisioning Portal menu for provider tenants

This CSS file may be edited to hide menu options per-provider:
PS.ProvisioningPortal/App/Content/Styles/Custom.css

Included in the release is an example for Provider ID 1 – in it you will see several menu options are hidden per Provider ID, for example this hides the Working Hours tab:

```
#maintitle_p1_HoursTab
{
    display: none;
}
```

Each tenant created by a template assigned to a provider will have the menu options assigned a CSS ID with the provider ID (“p1” for provider 1 above) to allow them to be easily hidden per-provider, for when created tenants are not going to need features and the interface should be simplified from CSS accordingly. These are the possible per-provider CSS IDs (Admin may be disabled per configuration globally, x is the provider ID from “prpTenantProviders”):

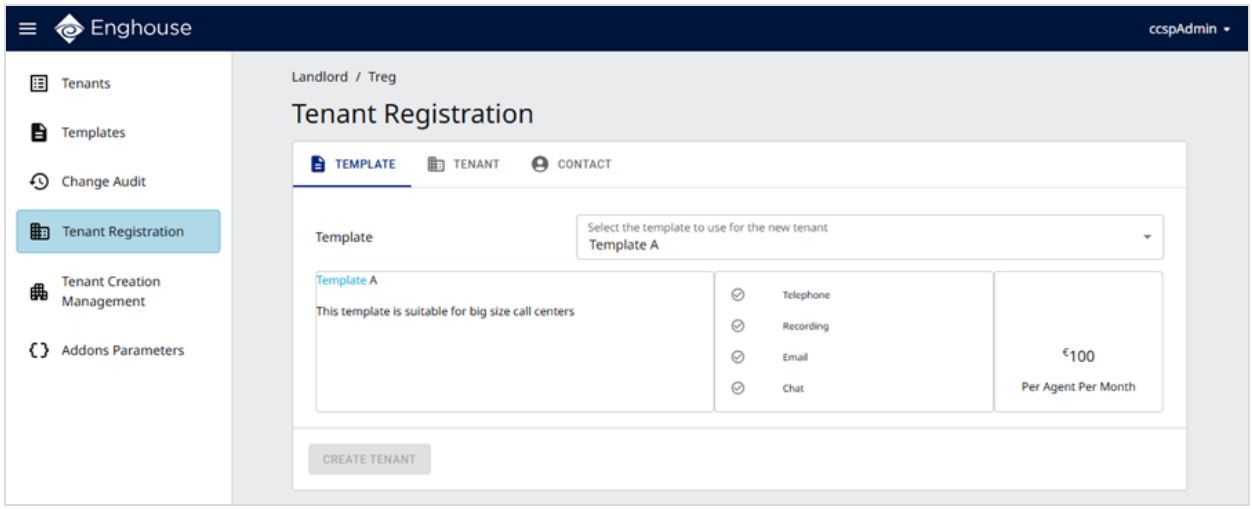
- maintitle_px_HoursTab – Working Hours
- maintitle_px_FlowsTab - Flows
- maintitle_px_FlowsTablesTab – Flow Tables
- maintitle_px_GreetingsTab - Greetings
- maintitle_px_CustomChatTab – Custom Chat
- maintitle_px_PromptsTab - Prompts
- maintitle_px_ParametersTab – SmartQ Parameters
- maintitle_px_AdminTab - Admin
- adminstitle_px_PersonnelTab – Admin Personnel
- adminstitle_px_GroupsTab – Admin Groups
- adminstitle_px_QueuesTab – Admin Queues
- adminstitle_px_SkillsTab – Admin Skills
- adminstitle_px_ProfilesTab – Admin Permission Profiles
- adminstitle_px_ReleaseCodesTab – Admin Release Codes
- adminstitle_px_WrapupCodesTab – Admin Wrap-Up Codes
- adminstitle_px_TeamsTab – Admin Teams
- adminstitle_px_FoldersTab – Admin Folders

- adminstitle_px_BrandingTab – Admin Branding
- list_px_Personnel – Admin Personnel content (as default tab this should be set to match adminstitle_p1_PersonnelTab to prevent tab AND content)

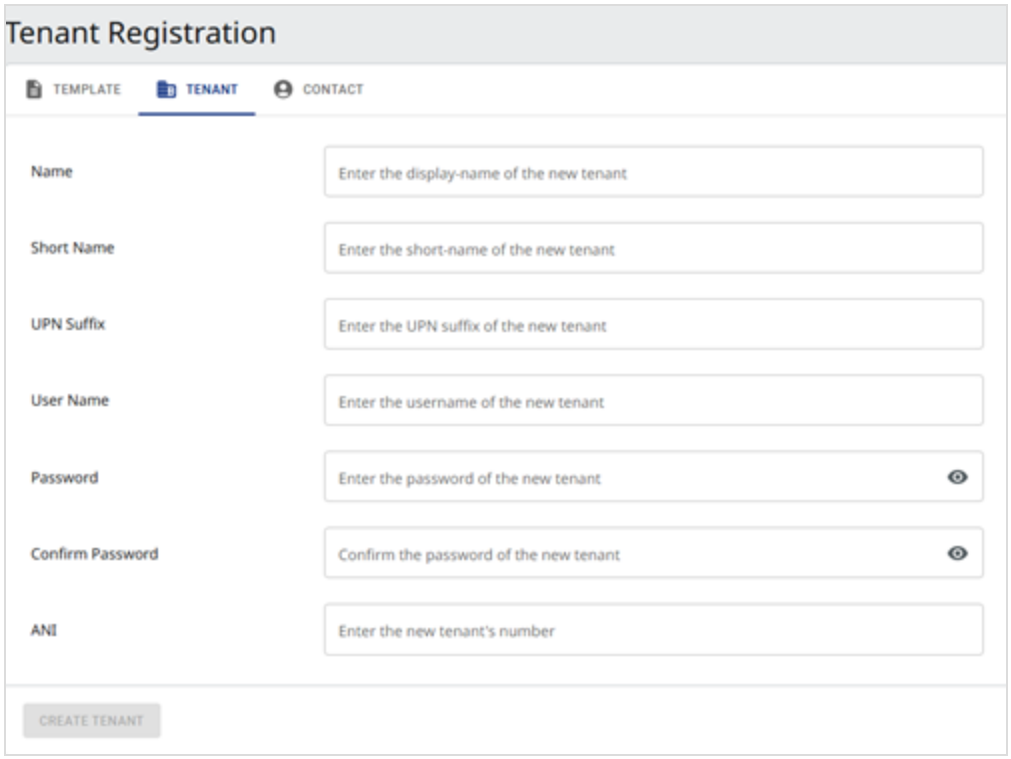
Appendix E - Tenant registration

Prior to tenant creation, a landlord or an administrator must register the tenant particulars by completing the following information – Template, Tenant, and Contact.

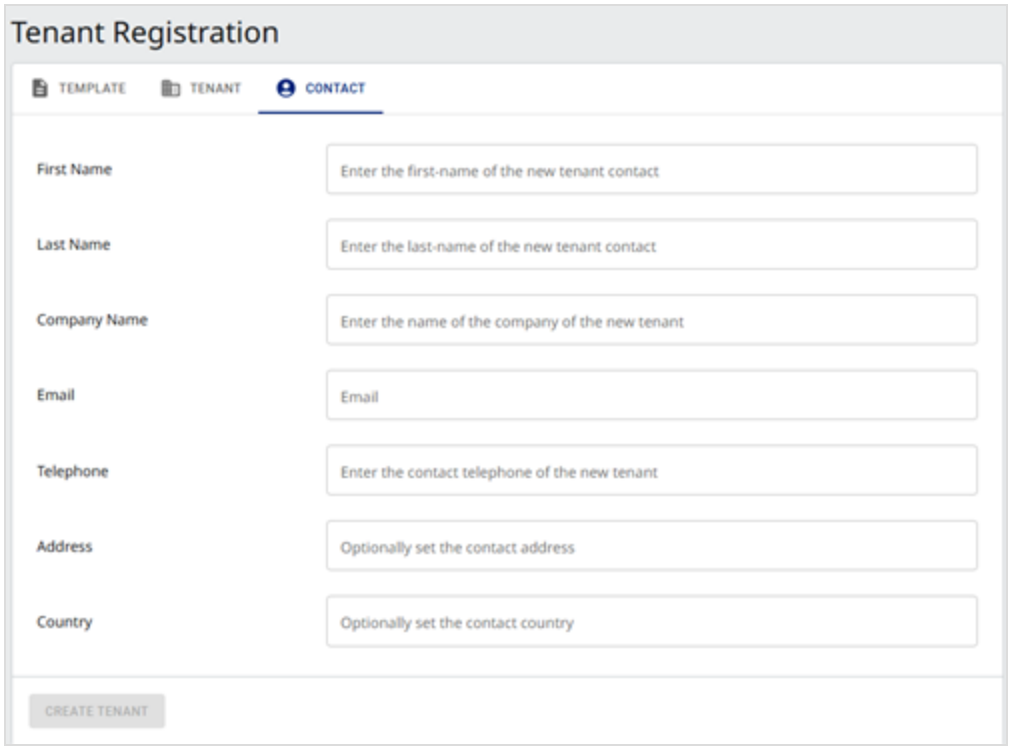
1. On the **Tenant Registration** page, on the **Template** tab, choose a suitable preconfigured template that aligns with the client's tenant requirements.



2. Click the **Tenant** tab and complete the tenant information, including Name, Short Name, UPN Suffix, User Name, Password, and the ANI (Automatic Number Identification) of the new tenant's phone number.



3. Click the **Contact** tab and enter the tenant's relevant contact information, such as first and last name, company name, email, phone number, and contact address, to register in the system.



4. After completing the registration information outlined above, click **Create Tenant** to register a new tenant in the system.

For more details on how to verify the registration process or check the status, please refer to Appendix F - Tenant creation.

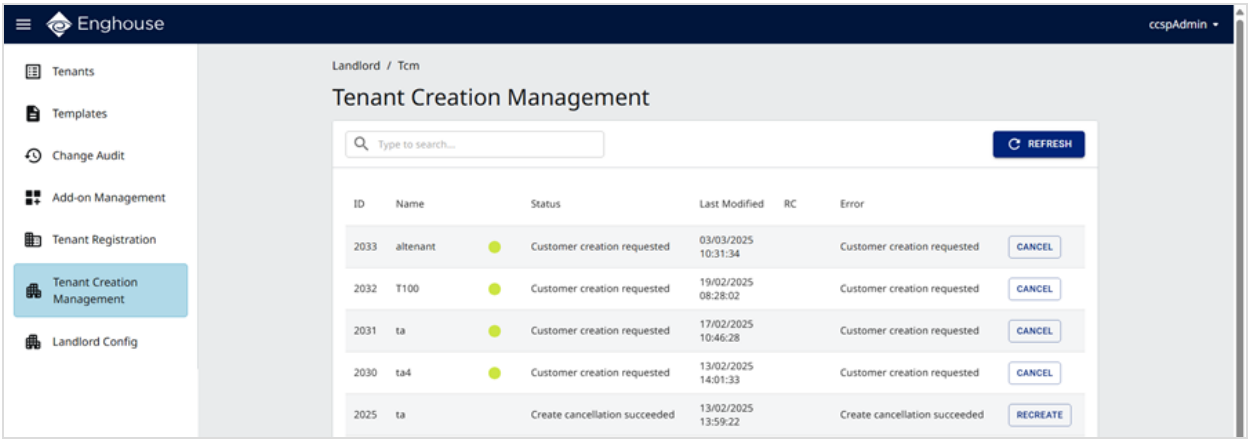
Appendix F - Tenant creation

Landlords creating tenants using the Provisioning Portal Tenant Creation page since release 1.0.15.7 have a facility to manage some aspects of that tenant creation progress.

The management is enabled in Provisioning Portal using this *web.config* setting:

```
<add key="EnableTSCMgmt" value="true" />
```

If enabled a new **Tenant Creation Management** page will appear on the main screen.



This tool shows the status of each tenant creation request, and where appropriate actions that apply to the status.

> **Note**
>
> Tenants previously created with previous release of Provisioning Portal will likely show their status as in-progress unless the database is manually altered to the relevant success/failed outcome – for example, executing this SQL on Provisioning Portal database (while tenant creation service TSC stopped):

```
UPDATE [prpTenantCreationRequests]
SET BillingStatus=6
FROM [prpTenantCreationRequests] req
JOIN [prpTenantCreationLogHeaders] head ON req.ID = head.Request_ID
WHERE req.BillingStatus IN(0,4)
AND EXISTS(SELECT * FROM [prpTenantCreationResults] res WHERE res.Request_ID=req.ID)
AND (SELECT TOP 1 ProcessID FROM [prpTenantCreationLogHeaders] head2 WHERE head2.Request_
ID=req.ID ORDER BY ISNULL(head2.LastModifiedDate, head2.InsertDate) DESC) = head.ProcessID
AND (SELECT TOP 1 Status FROM [prpTenantCreationLogHeaders] head2 WHERE head2.Request_
ID=req.ID AND head2.ProcessID=head.ProcessID ORDER BY ISNULL(head2.LastModifiedDate,
head2.InsertDate) DESC) = 'Error'
```

```
UPDATE [prpTenantCreationRequests]
```

```
SET BillingStatus=5
FROM [prpTenantCreationRequests] req
JOIN [prpTenantCreationLogHeaders] head ON req.ID = head.Request_ID
WHERE req.BillingStatus IN(0,4)
AND EXISTS(SELECT * FROM [prpTenantCreationResults] res WHERE res.Request_ID=req.ID)
AND (SELECT TOP 1 ProcessID FROM [prpTenantCreationLogHeaders] head2 WHERE head2.Request_
ID=req.ID ORDER BY ISNULL(head2.LastModifiedDate, head2.InsertDate) DESC) = head.ProcessID
AND (SELECT TOP 1 Status FROM [prpTenantCreationLogHeaders] head2 WHERE head2.Request_
ID=req.ID AND head2.ProcessID=head.ProcessID ORDER BY ISNULL(head2.LastModifiedDate,
head2.InsertDate) DESC) = 'Success'
```

There are 2 fundamental types of tenant creation:

- Legacy – 2-step:
    - Web UI gets tenant details, takes payment, then stores in database with status 0
    - TSC reads pending creation (status 0) from database and commences tenant creation (changes status to 4)
- Provider – 4-step:
    - Web UI gets tenant details, stores in database with status 1
    - TSC reads pending customer creation (status 1) and commences to create customer (changes status to 2) in payment integration (for example, WebTop or Stripe).
    - Customer system notifies customer to pay, when customer pays it notifies Provisioning Portal tenant creation may proceed (changes status to 3)
    - TSC reads pending creation (status 3) from database and commences tenant creation (changes status to 4)

From this point on the 2 types of tenant creation are effectively the same, if successful the status is changed to 5, if unsuccessful the status is changed to 6.

The new tab in Provisioning Portal uses this status to display tenant creation progress and where appropriate some actions, the status descriptions and actions are best identified by their translations *(App\Content\Translations\default.xml)* with BillingStatus_<Status> being the status description and BillingStatusAction_<status> being the action button if appropriate.

Statuses 0-9

If the status is 2 it means the customer has been created and notified to pay, so we are not allowing cancel/rollback here. If status is 4 it means TSC is in the process of creating the tenant so we are not allowing cancel/rollback here. Other statuses <10 offer some action to cancel/rollback – behind the scenes this is basically +10 to the status code, so if a tenant successfully created with status 5 the action will change the status to 15 kind of thing.

Status 10-19

The web UI action button for a status has been pressed to indicate a cancel/rollback is required. If <15 it is a cancellation – nothing is created so nothing is rolled back. If status 15+ it means tenant creation was started

and so we want to rollback – removing the tenant entirely from CCSP, the databases from server, and from ConfigPortal database the tenant's widgets from (CHT_Widgets) and tenant's menu entries (CNP_MenuItemRole). TSC will check for the statuses 15/16 as rollback - when TSC starts to performs a rollback it will increment the status by a further 10. This group is primarily a pending cancel/rollback state, so the actions are to cancel the cancel/rollback and just revert the status to what it was before (-10).

Status 20-29

Cancel/rollback in-progress – on success +10 added, on failure +20 added

Status 30-39

Successful cancel/rollback – actions to recreate (change status to 0 for TSC to consider new creation)

Status 40-49

Failed cancel/rollback – actions to recreate (change status to 0 to consider new creation)

Statuses where we are waiting for billing system or tenant creation/rollback show their icon pulsating, others show a static color (or no icon if status 30+). Throughout tenant creation, the Refresh Statuses button should be pressed to see changes (latest updates should be listed first).

Enable SQL updates on start-up

The Provisioning Portal Tenant Creation service (Ps.TscSvc) can now automatically perform the required SQL updates on start-up, if the `SQLInstallFolder` configuration is set to the folder containing the to Provisioning Portal database install SQL files.

For details, see Enable auto SQL updates.

Appendix G - Tenant configuration

Landlords and tenants can now use the **Addons Parameters** page of the Provisioning Portal to configure the following addons.

Addon	Enabled by landlord via
Contact History	`PrPConnectionString` in the tenant *.config* file
Social Connector KPI Dashboard	`PrPConnectionString` in the tenant *.config* file
Social Connector File Transfer	`PrPConnectionString` in the *web.config* file
Social Connector Translator*	`PrPConnectionString` in the *web.config* file
Suggestions	`PrPDB` connection string in *appSettings.json*
Text to Speech	`PrPDB` connection string in *appSettings.json*
* Used by Custom Chat.	

Landlords can configure the defaults for each addon and configure the KPI addon. Note that these addons must be updated to utilise the new configuration via the Provisioning Portal.

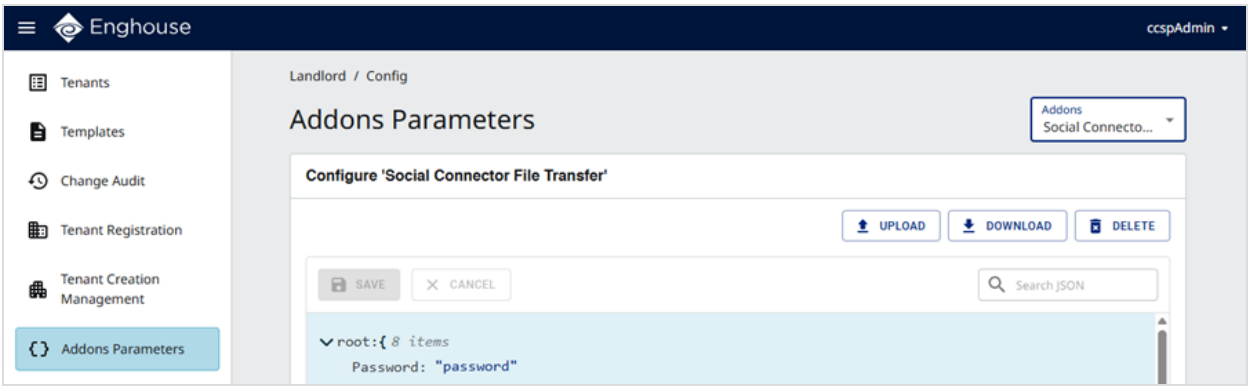
Tenants can administer their own add-ons for Contact History, Social Connector Translations and File-transfers, Suggestions, and TTS.

> **Note**
>
> - If the `PrPConnectionString` is configured and enabled, the corresponding addon instance always retrieves and uses the configuration from the database.
> - If the `PrPConnectionString` is not configured, it uses the JSON configuration file located in the installation directory instead.
> - The `ConfigCacheMinutes` setting used by SC Helper API is used to periodically retrieve the configuration from the database.

Download an existing configuration file

The landlord or tenant administrator can download a backup copy of the configuration file. The configuration file defaults to the JSON format.

1. On the **Addons Parameters** page, select the component from the **Addons** list. The **Configure [Addon Name]** dialog appears.



2. Click DOWNLOAD.
3. The configuration is saved as a JSON file to your downloads folder.

Upload a configuration file

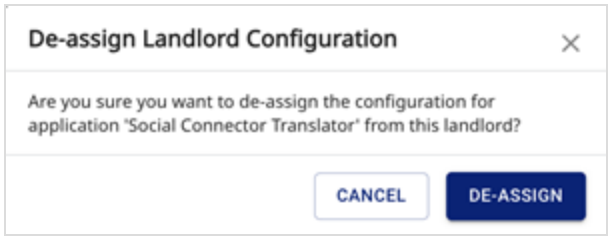
> **Note**
>
> - Each addon has an associated schema format that is enforced to ensure the appropriate JSON configuration file is uploaded and processed by the system. See Schema files.
> - The schema format is enforced when you click **Save**, not on upload.

If the current configuration is corrupt or incorrect and an error has occurred, you can restore a backup copy of the existing configuration file.

1. On the **Addons Parameters** page, select the component from the **Addons** list. The **Configure [Addon Name]** dialog appears.
2. Click UPLOAD. The file selection dialog appears.
3. Select the JSON file and click **Open**.
4. When the JSON configuration is uploaded to the system, the configuration data appears in the form.
5. Verify the configuration constrains and modify if required. See Edit a configuration field.
6. When the configuration file is finalized, to upload the configuration to the database, click SAVE.

Delete a configuration file

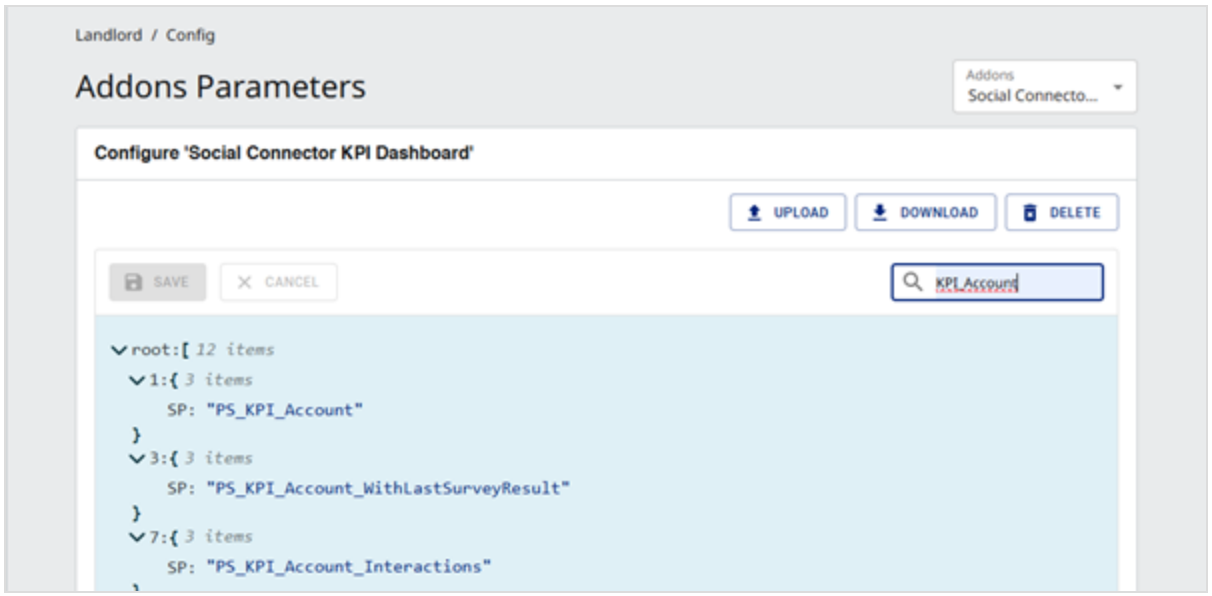
1. On the **Addons Parameters** page, select the component from the **Addons** list. The **Configure [Addon Name]** dialog appears.
2. Click DELETE. The **De-assign Landlord Configuration** dialog appears.



3. Click **De-assign**. The configuration data is removed from the Provisioning Portal.

Search for a configuration property

1. On the **Addons Parameters** page, select the component from the **Addons** list. The **Configure [Addon Name]** dialog appears.
2. In the **Search JSON** field, enter the search text, for example, *KPI_Account*. Configuration items that contain the search text appear in the dialog.



Edit a configuration file

Note

- The Save and Cancel buttons become active after you accept a change to a property's value or delete a property's value.
- Schema validation is applied to the submitted JSON before saving the configuration in the database. See Schema files.
- When you save the changes, they are committed to the database and the new configuration is enforced .

1. On the **Addons Parameters** page, select the component from the **Addons** list. The **Configure [Addon Name]** dialog appears.
2. Select the property to edit. The icons to copy, edit or delete the property's value appear next to the selected property.

3. To edit the property's value:

   a. Click . The property's value field and a list box of data types (such as number, string, and boolean) appear.

b. Select the data type and then edit the value.

c. To accept that change, click or to discard that change, click .

4. To delete the property's value, click .

5. To commit the change to the database, click SAVE.

Schema files

Parent folder

Each add-on optionally has a schema file in the *PS.ProvisioningPortal\JSONSchemas* folder.

Landlords can upload an existing configuration that may span multiple tenants and have a default.

The default schema files ensure that the property names or format of an addon's JSON configuration file are configured correctly when a user adds or edits the configuration file.

If the schema file does not exist, the uploaded JSON is just accepted. Each JSON file defines the expected attributes and required attributes for landlord-level uploads for the add-on specified in their filename.

Tenant sub-folder

There is a *tenant* sub-folder for each tenant that contains the tenant version of each schema. The tenant-level schema is specifically for the single-tenant's configuration.

Upload schema files

The landlord and each tenant can upload the following schema files.

Addon	Landlord	Tenant
Contact History	Uploads and modifies the provided template *config.json* (in *XRM* folder) with the relevant URLs	Uploads previous configurations they downloaded from Provisioning Portal
Social Connector KPI Dashboard	Uploads the *kpis.json* from the SC Helper API	Not applicable.
Social Connector File Transfer	Uploads the *filetransfer.json* from the SC Helper API	Uploads previous configurations they downloaded from Provisioning Portal
Social Connector Translator	Uploads the *translators.json* from the SC Helper API	Uploads previous configurations they downloaded from Provisioning Portal
Suggestions	Uploads the current *appSettings.json* from these web-apps' folders	Uploads previous configurations they downloaded from Provisioning Portal
Text to Speech	Uploads the current *appSettings.json* from these web-apps' folders	Uploads previous configurations they downloaded from Provisioning Portal

Appendix H – Deployment script

From release 1.0.16.1, a PowerShell deployment script is included into the bundle to simplify the deployment process of the Provisioning Portal. This PowerShell script allows you to:

- Perform a fresh installation
- Upgrade an existing installation

Prerequisites

Please ensure .net management framework is up to date. Minimum version 5.1 is required to run the script appropriately.

- Windows SQL Server 2016.
- Please ensure you have not moved the script from the default directory, or else the script will not run appropriately for deploying the required components for Provisioning Portal.
- Please ensure CCSP UI Admin is installed on the server as it is one of the mandatory requirements prior to deploying Provisioning Portal.
- Please ensure the correct IIS path for CCSP UI Admin and Provisioning Portal is specified correctly in the configuration file. Or else, the upgrade process might not be successful as the service is actively reading/running the .dll file in the directory that prevents modification.
- *config.json* file and Bulk Import script must be in the same folder for the script to work. Verify those files are in the same folder. If *config.json* file is not found the script will not run and an error will be printed to the console.

  Please follow the guideline to edit the JSON configuration file in this doc, and to ensure the syntax of the JSON file is correct. To check the syntax of the JSON file, please visit the following URL: https://jsonlint.com

- Open Windows PowerShell ISE as administrator (or command window), open the PowerShell script, run it and follow the instructions printed to the console. PowerShell has many security settings and sometimes they block scripts from running.

  If you have any problem, please try changing execution policy as explained in the following link: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-7.3

PowerShell diagnosis

If SQL parameter wasn't executed properly, please run the following command:

```
get-help invoke-sqlcmd -full
```

Fresh installation

> **Note**
>
> - Please ensure that you have configured the settings in configuration file *pws_config.json* before using the PowerShell deployment script
> - The directory specified under the 'prp_installation' section is created. Otherwise, if the deployment script cannot locate the file path, it will automatically generate the path and continue with the deployment.

For fresh installation, there are three options in the menu.

- Deployment of SQL (database)
- Installation of Provisioning Portal binaries
- Installation of Provisioning Portal (Service) binaries

Deployment of SQL (database)

For SQL deployment, the script restores a backup image of the Provisioning Portal database included in the package. Once the restoration process is completed, the last updated version is found by retrieving the information through the version schema table. Lastly, if the retrieved schema version is outdated, the script will proceed with applying the needed SQL patches to the database to ensure the Provisioning Portal database is up to date.

> **Note**
>
> Currently, the script does not support remote execution. For example, if the SQL server is installed on an auxiliary server and the script is running on the primary server, the database deployment process will fail. Thus, restoring or deploying artifacts to the SQL instance requires running the script on the local machine that hosts the SQL server.

Installation of Provisioning Portal binaries

The script will copy the latest binaries from the package to the specified installation path from the configuration file. When it is done, it will perform a quick check on IIS web server to determine if an existingProvisioning Portal site or web app is installed on the server. If no Provisioning Portal site or web app is found in the system, it will then proceed with deploying Provisioning Portal app pool and site on the server.

> **Note**
>
> The created Provisioning Portal app pool and site on IIS server will not be started by the script as you need to verify that the *web.config* file in Provisioning Portal is configured correctly. You can then start the app pool manually through IIS management console/server.

Installation of Provisioning Portal (Service) binaries

Prior to the installation of Provisioning Portal (Service) binaries, the script performs a check on whether The Provisioning Portal Service is installed on the server. If an existing service is found and currently running on the server, it will then stop the service before proceeding with the installation.

> **Note**
>
> After the installation of Provisioning Portal (service) binaries, the service will not be started by the script as you need to verify that the *web.config* file in the binaries folder is configured correctly. You can then start the service manually through the server.

Upgrade installation

> **Note**
>
> - Please ensure that you have configured the settings in configuration file *pws_config.json* before using the PowerShell deployment script.
> - The directory specified under the 'prp_installation' section is created. Otherwise, if the deployment script cannot locate the file path, it will automatically generate the path and continue with the deployment.

For upgrade installation, there are three options in the Provisioning Portal Upgrade Installation menu:

- Upgrade of Provisioning Portal database
- Upgrade of an existing installation of Provisioning Portal binaries
- Upgrade of an existing installation of PS.TscSvc service (optional)

Deployment of SQL (database)

There won't be restoration for Provisioning Portal database will not take place with the upgrade procedure. The script will execute a SQL command to determine the last updated version by retrieving the info through the version schema table. Lastly, if the retrieved schema version is outdated, it will then proceed with applying the required SQL patches to the database to ensure the Provisioning Portal database is up to date.

Upgrade of Provisioning Portal binaries

Prior to upgrade the existing binaries from the specified path, the script will look up if there are any active provisioning portal server (IIS) app pool running on the server. If an active app pool is found, it will be stopped and begin updating the binaries from the package to the specified installation path from the configuration file.

> **Note**
>
> The stopped Provisioning Portal app pool and site on IIS server will not be started by the script as you need to verify that the *web.config* file in Provisioning Portal is configured correctly. You can then start the app pool manually through IIS management console/server.

Upgrade of Provisioning Portal (Service) binaries

Prior to upgrade the existing Provisioning Portal service binaries from the specified path, the script will look up if PS.TscSvc is running on the server. If the service is started, it will be stopped and begin updating the service binaries from the package to the specified installation path from the configuration file.

> **Note**
>
> The stopped Provisioning Portal service PS.TscSvc will not be started by the script as you need to verify the *web.config* file in Provisioning Portal is configured correctly. You can then start the service manually from the server.

Appendix I – SSO configuration

> **Note**
> From CCSP HF74-70186 (7.4 SSO Direct Login), the Provisioning Portal application supports direct SSO login. For more information, refer to the *CCSP 7.4 Single Sign On Guide*.

From release 15.10 Provisioning Portal has OIDC integration with the CCSP AuthServer to permit Single Sign-On (SSO) via new OIDC and ConfigCOM integrations.

To enable this, use the new CCSP Admin Custom SSO Applications to create a configuration for Provisioning Portal. The recommended parameters are:

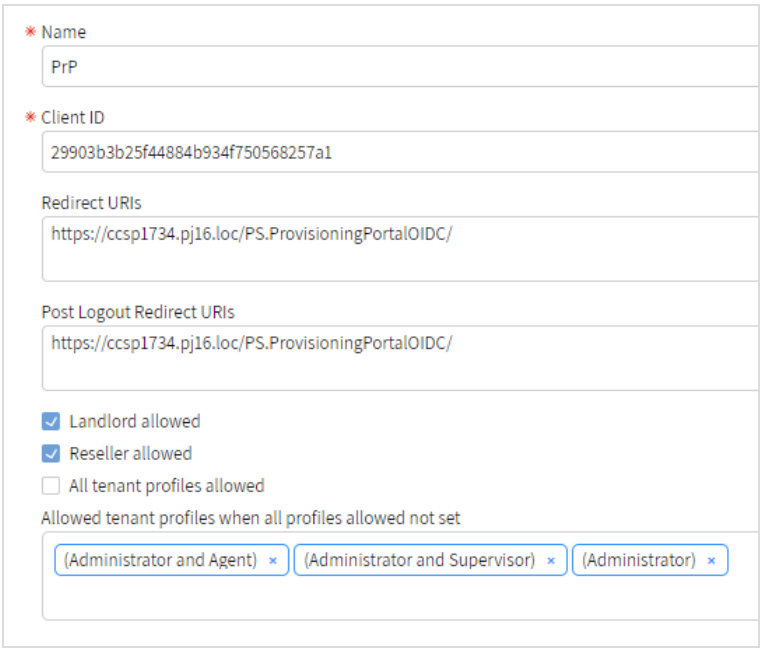
- Name: PrP
- Client ID: 29903b3b25f44884b934f750568257a1
- Redirect URIs: comma-separated list of URL paths that may be used for Provisioning Portal ending with /, for example: *"https://ccsp1734.pj16.loc/PS.ProvisioningPortal/"*
- Post Logout Redirect URIs: same as Redirect URIs
- Landlord allowed: checked
- Reseller allowed: checked
- All tenant profiles allowed: unchecked
- Allowed tenant profiles: any with Administrator. To permit Supervisors create a Permissions Profile for the logins that require access for that tenant as demonstrated in the example screenshots that follow

In the *web.config* of Provisioning Portal, populate these "appSettings":

- OAuthSecretKey: the secret key configured on the platform across CCSP UI, etc.
- AuthServers: comma-delimited Auth Server Issuer list for Single login page functionality
- Audiences: valid client IDs (minimally including the Client ID above)
- AuthServerURL: Auth Server Issuer URL for Single login page functionality - must include only 1 URL and end with /, for example:
  "https://ccsp1734.pj16.loc/AuthServer/"
- ClientID: the Client ID defined above
- ClientName: the Name defined above

Example configuration from Admin:

SSO Application:



If tenants only require administrators to access Provisioning Portal, the above is fine, but to permit supervisors the tenants are required to have a Permission Profile assigned to those supervisors as with previous versions of Provisioning Portal. This profile should then be assigned to the Provisioning Portal SSO application (along with the administrators included in the profiles), for example: