Provision and Catalog Tutorial

# Purpose of this tutorial

The intro to Wingtip SaaS walkthrough showed how the Wingtip app can provision a tenant database with its initial schema and register it in the catalog, making it available for use by the rest of the application. To do this, the app uses ARM APIs and SQL commands. The same APIs are available for scripting also. This tutorial explores PowerShell scripts that do the same thing, and uses them to learn more about the provisioning and cataloging patterns used, and how the catalog has been implemented.

# Introduction to the SaaS Catalog Pattern

In a database-backed multi-tenant SaaS application, it’s important to know where information for a tenant is stored. In the SaaS Catalog pattern, a catalog database is used to hold a mapping between a tenant and a database in which that tenant’s data is stored. The basic pattern applies whether a multi-tenant or a single-tenant database is used.

The catalog in the WTP app is implemented using Shard Management technology in the Elastic Database Client Library (EDCL). This library is responsible for creating and managing a database-backed ‘catalog’ in which a shard map is maintained, which contains mapping between sharding keys and databases. While the data is readily accessible in the database it should not be manipulated except through the APIs exposed by the EDCL. Direct access to tables data risks corrupting data and is not supported.

EDCL also provides important features that enable a client application to connect to and manipulate the catalog and to use it to retrieve an ADO.NET connection for a given key value, enabling the application to connect to the correct database. The client caches this connection information to minimize the traffic to the catalog database and speed up the application.



# Using the Catalog

The Wingtip application is built using a single-tenant database model. A tenant id is formed for each tenant, and this is used as the key to map the tenant to its database. Exactly how the key is formed is not critical, although the key either has to be known to or derivable by any client application that needs to access the tenant’s data. In the WTP application, a hash algorithm is used to create an integer key from the tenant’s name. This allows the tenant name portion of the application URL to be used to construct the key and retrieve the connection. Other id schemes could be used without impacting the overall pattern.

The tutorial walks how the catalog is formed and used when provisioning a tenant.

# Setting Up

**SaaS in a Box Learning Modules and Utilities.zip** should be downloaded and extracted to a convenient <root> folder.

**WTP Application** must be installed and setup, with the catalog initialized. See <link>

**PowerShell ISE** is recommended executing scripts and following their execution in debug mode.

PowerShell ISE Once open in the ISE, use F5 to execute. F9 to set a breakpoint. F10 to step through a script and F11 to step into a function, Shift-F11 to step out. Use <#....#> to comment or uncomment sections before using F5. F5 must be used for $PSScriptRoot to be evaluated – using F8 to execute fragments will cause errors.

**SSMS** should be used to explore database schema and execute SQL queries directly.

# Walkthrough

## Getting Started

Open the following scripts in **PowerShell ISE**

* …\Learning Modules\Provision and Catalog\Demo-ProvisionAndCatalog.ps1
* …\Learning Modules\Provision and Catalog\New-Tenant.ps1, New-TenantBatch.ps1
* …\Learning Modules\Common\CatalogAndDatabaseManagement.psm1
* …\Learning Modules\Common\ShardManagement.psm1

## Provision a new tenant – 40,000 ft

This exercise will create a new tenant and register it in the catalog.

* Open Demo-ProvisionAndCatalog.ps1
* Modify $WtpResourceGroupName – use the resource group in which you deployed the WTP app
* Modify $WtpUser – use the user name used when you deployed the WTP app
* Modify $TenantName – use the name of the Venue you plan to provision
* Modify $VenueType – use one of the pre-defined venue types. You’ll modify this set in a later tutorial.
* Check that the script reference $PSScriptRoot\New-Tenant.ps1 is uncommented
* Execute using F5

The script will execute and create a new tenant, provisioning a new database and register it in the catalog. It will then open both the events (tenant user) and the admin apps associated for the tenant. Note the URL pattern used for both apps.

## Provision a new Tenant – 500 ft

Now do the same again and step through the script and see what is done.

* Modify $TenantName to create another venue
* Use F9 to add a breakpoint to the New-Tenant.ps1 execution line
* Use F5 and the F11 to step into New-Tenant.ps1 and trace its execution
* Imports modules containing useful PowerShell functions:



* + SubscriptionManagement manages Azure login and subscription selection
  + CatalogAndDatabaseManagement provides a catalog- and tenant-level abstraction over the Shard Management functions
* **Get the configuration**. Step into Get-Configuration with F11 to see how config is specified.



* **Get a catalog object**. Step into Get-Catalog to see how the catalog is initialized using Shard Management functions that are imported from AzureShardManagement.psm1



* + $catalogServerName is constructed using the standard stem plus your $WtpUser
  + $catalogDatabaseName is retrieved from config
  + $shardMapManager object is initialized from the catalog database
  + $shardMap object is initialized from the “customercatalog” shard map in the catalog database
  + A catalog object is composed and returned, and used in the higher-level the script.
* **Get the tenant key**. The tenant name is used to create the tenant key using a hash function (MD5 in next version)
* **Check the tenant doesn’t already exist**. The catalog is checked to ensure the tenant key has not already been used.
* **The tenant database is provisioned.** Use F11 to see how the database is provisioned using an ARM template.
  + Two ARM templates are supported (controlled in config)
  + **Create a new database by copying a ‘golden’ database** (baseTenantDB on the catalog server)
  + **Create a new empty database and then import a golden bacpac**
  + The WTP app currently provisions by copy so the scripts use the ARM copy-based template as the default
  + The ARM templates are in …\Learning Modules\Common\
    - tenantdatabasetemplate.json
    - tenantdatabasecopytemplate.json
* **Register the tenant database in the catalog** using the tenant key. Use F11 to step into the function:
  + Add the catalog database to the shard map
  + Add the mapping to the key
  + Add additional meta data to describe the tenant.