

# **How to Deploy Datacap Web Services (wTM) in Microsoft Azure**

## **Technical White Paper**

---

**by**

**Open Connections**

# Configuring the IBM Datacap Transaction Endpoints in Microsoft Azure in a Highly Available Manner

## 1. Introduction

The purpose of this paper is to present the deployment of Datacap Web Services (wTM) in the Azure Cloud in response to a customer requirement. The requirement was for Datacap Web Services to be implemented in Azure in a highly available manner, which was dynamically scalable in response to workload and system demand. The customer had expressed a desire to consume Datacap's Transactional Endpoints to enrich their documents in an efficient, scalable and reliable way. This paper describes how this was implemented in a Proof-of-Concept environment in the Microsoft Azure Cloud, using cloud-native features including:

- **Azure VM Scale Sets**  
allowing horizontal scaling of the Web Servers on demand, depending on the system load
- **Azure SQL**  
for hosting the Datacap databases (Admin, Engine, Fingerprint)
- **Azure Application Gateway**  
for balancing traffic to Datacap Web Servers, (IIS Servers)

We were keen to discover the extent to which it was possible to achieve this in Azure and document the solution.

## 2. The Business Problem

The customer receives unstructured documents into their business from a variety of sources. These documents are all processed and stored in a line of business system where logic is applied to them to determine whether they need to be “enriched”. The process of enrichment typically involves carrying out OCR on the documents in question and extracting some data from them, which is stored with the documents in the line of business system.

The requirement was that this “enrichment engine” be highly available, scalable and robust and that it should make use of the customer’s existing investment in Datacap. The customer had a target of enriching up to 4000 documents per hour at peak times. They therefore require the ability to increase or decrease the number of Web Servers (horizontal scaling) in response to demand to meet this SLA.

### 3. The Solution

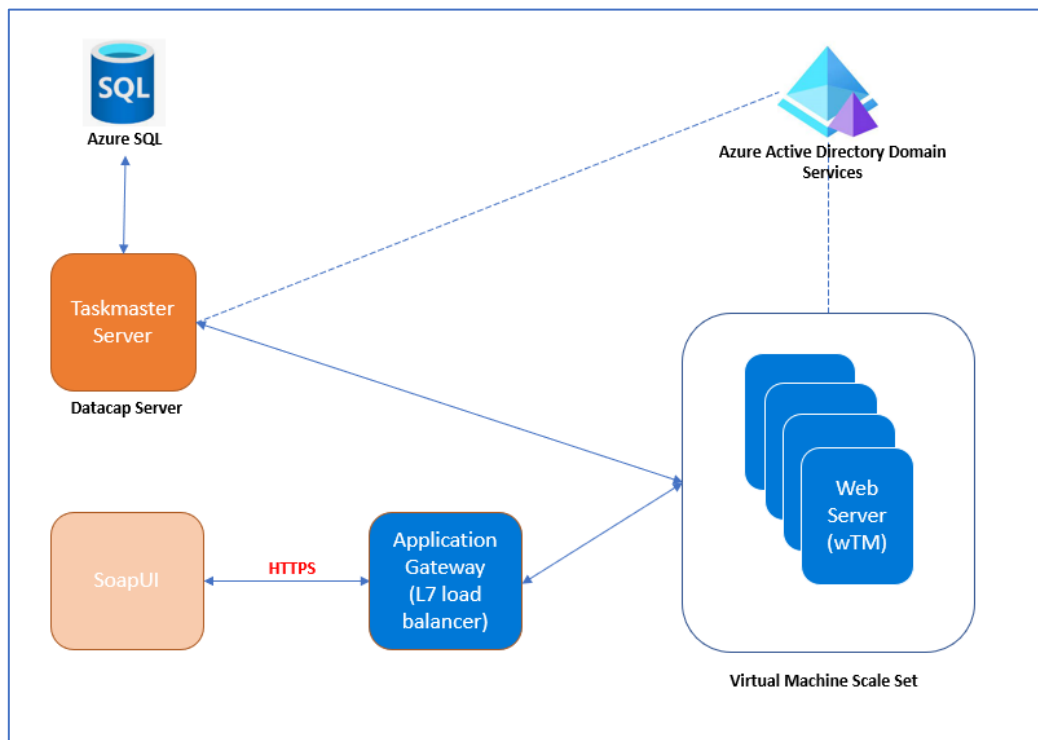
Copies of documents in need of enrichment are to be submitted to Datacap by the customer's own middleware platform. Datacap carries out the enrichment and returns the resulting data to the middleware platform. The document copies which were submitted to Datacap are then destroyed.

For this Proof-of-Concept, there was no middleware platform. Rather, we used SoapUI (<https://www.soapui.org>), a free and Open-Source web service test tool to consume the web-service endpoints exposed by Datacap and view the results of the extraction. We built a simple Datacap Application which would OCR a TIF file and store the resulting text in a TXT file, then attach a piece of metadata to the document. The OCR text and metadata are returned using a web service call.

The architecture consists of a Datacap Taskmaster Server which hosts the Datacap Application and authenticates requests from SoapUI to establish sessions with the Web Servers. The Datacap Admin Database, which stores Datacap user permissions on a per-application basis, is hosted in Azure SQL. The Engine and Fingerprint databases are not required in this Proof-of-concept because no "traditional" Datacap batches are created by the Transactional endpoints and this application does not use Fingerprints. The requirement for a flexible and scalable processing architecture is met by using an Azure Virtual Machine Scale Set (VMSS). One Web Server was created initially and added to the Scale Set. The Scale Set is configured to create more web servers automatically as demand increases. The Azure VMSS supports a maximum of 1000 virtual machines. When demand decreases, web servers are automatically shut down and removed. The Application Gateway provides load balancing, directing traffic to the appropriate web server.

The web servers and Datacap Taskmaster Server must belong to the same domain. Azure Active Directory Domain Services provides the means of accomplishing this.

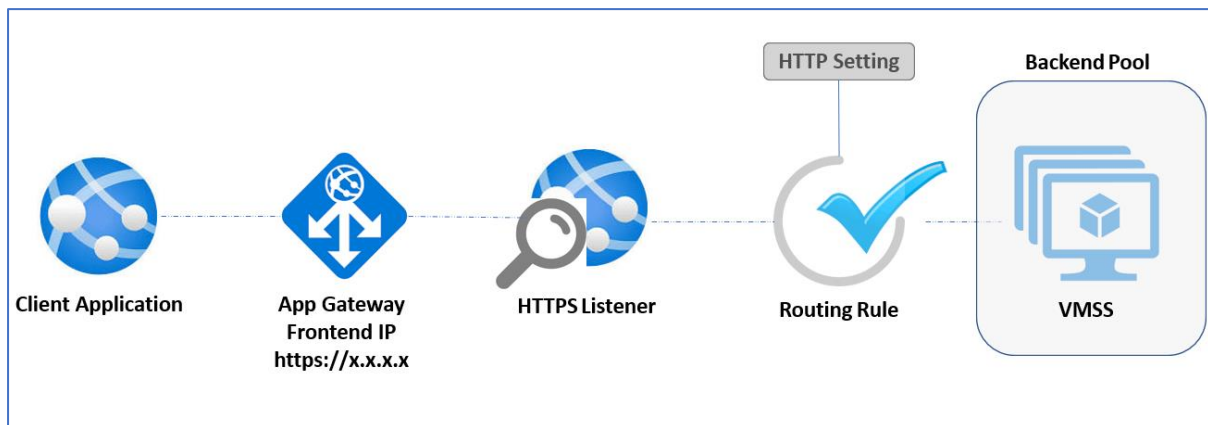
The proposed basic Logical Architecture looks like this:



## 4. Load-balancing Datacap Web Services using the Azure Application Gateway

Communication with Datacap Web Services is always via the HTTP protocol. Load-balancer functions such as application health monitoring, HTTP header manipulation and encryption therefore require a Layer 7 Load Balancer. In Azure, this is provided by the Application Gateway.

The configuration of the Application Gateway can be seen in the diagram below:



The Client Application, in this case SoapUI, makes calls to the Datacap Web Services Transaction Endpoints via SSL. These calls are made to the Application Gateway Frontend IP. An HTTPS Listener is configured to listen for requests on a certain port (in this case 443). The application gateway evaluates the request routing rule that's associated with the listener. This action determines which Backend Pool to route the request to. In our Proof-of-Concept, there is only one Backend Pool and it contains the VMSS. In practice, there could be several Backend Pools, each containing different resources or App Services which are available for consumption.

After the application gateway determines the backend pool and resource, it opens a new TCP session with the web server based on HTTP settings. HTTP settings specify the protocol, port, and other routing-related settings that are required to establish a new session with the web server.

The port and protocol used in HTTP settings determine whether the traffic between the application gateway and backend servers is encrypted or is unencrypted. For the proof-of-concept, this remained unencrypted.

If the VMSS in the backend pool contains multiple servers, the application gateway uses a round-robin algorithm to route the requests between them. A related series of web service calls related to a given document (a "session") needs to be completed on the web server which handled the initial request to process that document. Cookies are used to maintain sessions on given web servers.

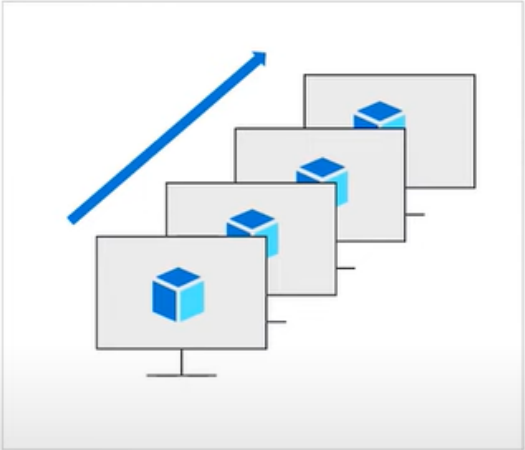
## 5. Datacap Web Servers in an Azure VM Scale Set (VMSS)

This diagram from Microsoft defines a Scale Set (see <https://aka.ms/VMSSWebPage>)

### Virtual Machine Scale Sets defined

Azure Compute service that lets you:

- Create and manage up to 1,000 load balanced VMs per availability zone in minutes
- Scale your applications automatically based on demand while increasing resiliency
- Use Azure Resource Manager templates to deploy Linux and Windows platform images, as well as custom images and extensions



[aka.ms/VMSSWebPage](https://aka.ms/VMSSWebPage)

We built a Datacap Web Server VM in the Azure Portal and used this to create a Virtual Machine Scale-set which would scale out (create additional web servers) when the average CPU load exceeds 75% for 5 minutes, scaling in when the average CPU load is less than 25%. There are several available metrics which are adjustable as demands change:

☒ Scale based on a metric
 ☐ Scale to a specific instance count

**Scale out**

When	DatacapVMSS2	(Average) Percentage CPU > 75	Increase count by 1
------	--------------	-------------------------------	---------------------

**Scale in**

When	DatacapVMSS2	(Average) Percentage CPU < 25	Decrease count by 1
------	--------------	-------------------------------	---------------------

[+ Add a rule](#)

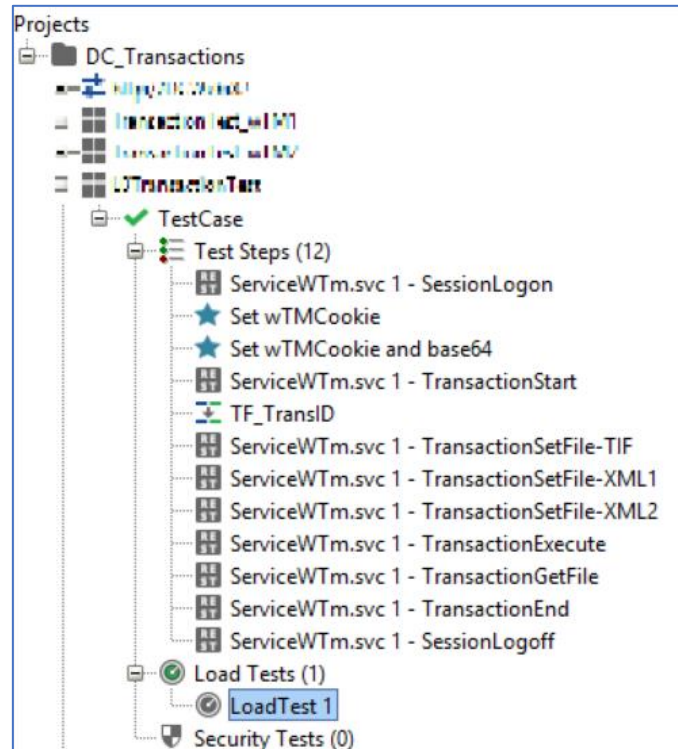
Minimum ⓘ

Maximum ⓘ  
 ✓

Default ⓘ  
 ✓

## 6. Bringing It Together – Exercising Datacap REST API Endpoints using SoapUI

A series of requests are made to the Datacap REST API to “enrich” a document:



**Session/Logon:** This authenticates a Datacap User with a given Datacap Application and establishes a connection with a given Web Server. If successful, a cookie is returned by Datacap Web Services. This must be used in all subsequent requests.

**Transaction/Start:** This indicates to the REST API that we wish to start a new Transaction. If successful, a Transaction ID is returned. This must be used in all subsequent requests.

**Transaction/SetFile:** This allows files to be uploaded to the transaction (i.e. the document to be enriched, a TIFF file in this example followed some XML files that Datacap requires for its internal operations)

**Transaction/Execute:** Executes a Datacap Ruleset on the uploaded document.

**Transaction/GetFile:** Gets the enriched data from the transaction

**Transaction/End:** Ends the Transaction, destroying the uploaded documents and data.

**Session/Logoff:** Logs the user out of the Datacap Application and ends the session with Web Server.

Further details on these and other available endpoints can be found by referring to this IBM webpage: <https://www.ibm.com/docs/en/datacap/9.1.6?topic=reference-datacap-web-services-rest-api-methods>

The REST API web service endpoints can be exercised by carrying out a Load Test which makes the defined series of requests in a configurable number of concurrent sessions. It can be observed that the VMSS automatically scales out in line with the autoscaling settings. Logging in to the web servers, it's possible to observe the temporary batch folders appear, become populated with the documents and XML and disappear when the session ends.

## 7. Conclusion

The Datacap REST API provides a little-known means of leveraging Datacap functionality by exposing several web services endpoints and making them available for consumption by external applications. This is available with every Datacap license and just needs to be installed and configured.

The requirement that spearheaded this work was for the fast enrichment of semi-structured and unstructured business documents.

While it is possible to achieve this on-premises, the advantage of using a public cloud such as Azure is that it provides the ability to scale resources horizontally on demand, whilst providing a secure and highly available environment in which the workload can be executed. Such functionality would be very difficult to implement on premises.

If you process large numbers of documents today, it's very likely that the Datacap REST API could help your organisation to process these quickly and efficiently. It's worth considering!

**\*\*\* END OF DOCUMENT \*\*\***