Booking System

[Architectural Design 1](#_Toc325947086)

[Business Requirements 1](#_Toc325947087)

[Security Considerations 1](#_Toc325947088)

[Solution Design 2](#_Toc325947089)

[Bookings.DAL 2](#_Toc325947090)

[Bookings.Endpoint.Azure 2](#_Toc325947091)

[Bookings.Models 2](#_Toc325947092)

[Bookings.Service 2](#_Toc325947093)

[Bookings.Endpoint 2](#_Toc325947094)

[SecurityModule 2](#_Toc325947095)

[Bookings.ConsoleApp 2](#_Toc325947096)

[Architectural Improvements 2](#_Toc325947097)

[Deployment of the service over ssl 2](#_Toc325947098)

[Asyncronus infrastructure 2](#_Toc325947099)

[Enabling Logging & Performance counters 3](#_Toc325947100)

[Enabling Azure AutoScaling 3](#_Toc325947101)

[Unit Test 3](#_Toc325947102)

[Integration test 3](#_Toc325947103)

[Comments /documentation /clean up / refactoring 3](#_Toc325947104)

[Points of Interest 3](#_Toc325947105)

[ACS security model 3](#_Toc325947106)

[Pessimistic lock on the dataEntities 3](#_Toc325947107)

[References 3](#_Toc325947108)

# Architectural Design

The architectural design was driven by the technical requirements.

Azure Components

* Azure WebRole
* SQL Azure
* ACS with custom Service Identities and Custom business rules

Technology used:

Entity Framework 4.31

WCF Restful Service to expose the service with json entities

HttpModule to validate ACS Tokens and pass user Claims to the service

Console application to demonstrate the service calls

# Business Requirements

To provide a better user experience for customers & operators I have implemented a "PreBooking" status so that a client can complete the booking without risking that somebody else take his tickets...

This approach is also useful to reduce locks and contention in the advanced status of the reservation…

In the current implementation there is missing a component to cleanUp the reservations in standby after X minutes and keeping an history of the "elapsed preBook"

An other possible solution to this issue would be to change the where condition to return tickets with a status PreBooked but elapsed after x minutes…

The other possible business model was that one to try to give to the users/agents the same tickets… in this scenario tickets would go to the users who complete first the transaction and not to users who started first the transaction.

# Security Considerations

The implemented solution is pretty simple, the client send username and password to the ACS service (over ssl) and the acs service return a token with an expiration timeout (eg. 10 mins)

The client send the token in the header of each request, the server validate all the request to ensure that the sent token is valid and retrieve the user claims to use in the application context.

The solution released is not 100% secure, to improve the security would be recommended to host the wcf service on HTTPS to avoid tampering of messages… or eventually using a certificate to sign the messages but that would improve the solution complexity and would affect on the deployment strategy and so on.

In the console application username and password are stored in clear in the config file, in a real world application username and password would be entered by the user or retrieved from a service after the registration.

Using ACS there is also the advantage of using 3rd parties identity provider and keep business rules on the azure management portal…. In our sample business rules are used to retrieve user name and eventually organization of the current user.

# Solution Design

## Bookings.DAL

Data Access Layer to call EF

## Bookings.Endpoint.Azure

Azure deployment project

## Bookings.Models

Model classes shared between applications

## Bookings.Service

Business logic of the service

## Bookings.Endpoint

SVC endpoint and config file

## SecurityModule

HttpModule to validate the SWT token and pass user claims to the service

## Bookings.ConsoleApp

Console application to invoke the service

# Architectural Improvements

## Deployment of the service over ssl

It would be useful to improve the security level, but it was not requested explicitly in the project scope

## Asyncronus infrastructure

If the service should be used for high traffic web sites it could make sense start considering to move Confirmation Booking Request in an asynchronous architecture to improve reliability of the application under high load...

In the case it could be considered using storage Queues and a worker role to process it.

## Enabling Logging & Performance counters

There was not an explicit requirement but it would be good for any application deployed on azure to have at least logging & performance counters enabled.

## Enabling Azure AutoScaling

There was not an explicit requirement but it would be good to have

## Unit Test

There was not an explicit requirement but it would be good to have.

The code released would require some adjustment to allow an easier dependency injection implementation.

## Integration test

There was not an explicit requirement but it would be good to have especially to ensure that token validation is working properly and the validation key is correct…

## Comments /documentation /clean up / refactoring

Code released is not ready for production and need some clean up and refactoring,

# Points of Interest

## ACS security model

The httpmodule and the claims model to secure the restful service.

Probably there is a better way to validate the header in the messages with some wcf inspector message, but this solution was simpler to implement for me.

## Pessimistic lock on the dataEntities

The data entities use ConcurrencyCheck data annotations on the Status and AgentName properties to avoid conflict if 2 agents requests seats at the same time…

In case the application would be used by million of users at the same time to book the same event, it would be possible to have an “automatic” retry to give a better user experience.

Transaction would make a lock on the db table but it would not be useful in this scenario where we implement an applicative pessimistic lock.

# References

http://code.msdn.microsoft.com/windowsazure/REST-WCF-With-SWT-Token-123d93c0