Politecnico Di Milano

Prototype Implementation of a Cloud Solution

Supervisor: Prof Sara Comai
Student: Sina Alviri 754816
2011-2012

ABSTRACT

This thesis is based on a proof of concept done with collaboration of 4ward s.r.l. It was a six month project, done for Global Blue an international financial institution.

This project is proposed to create a hybrid cloud to expose financial web services either on cloud or on-premis, on order to guarantee the maximum availability. Thus the concentration was on high-availability and then performance after it.

Thus, after an analysis of the main cloud providers in the market we have developed Saas distribution model on the following platforms:

- Amazon AWS (Cloud in Irlanda e Singapore)
- Microsoft Azure (Cloud in Irlanda, Olanda e Singapore)
- vmWare vCloud (On-Premise e Cloud in Francia, Italia e Singapore)

TABLE OF CONTENTS

INTRODUCTION

The primary concern of this project was to create a service which will be available to different clients. Therefore the availability of this client service was the priority because our client in an international enterprise that has different offices around the world (Singapore, Netherland, Austria, Italy, UK).

Thus the problem was that they had large number of clients that have to able to use the financial system of the company from all over the world, using different devices.

The main goal was to create an enterprise service on the cloud, in order to make it accessible from all the offices around the world.

About 4ward

4ward is a private company established in 2002. It is a young company that in the beginning started as an IT consulting entity in the market, then turned into Microsoft's golden partner in Italy. Thus the base technology used by 4ward, are those from Microsoft. 4ward gives consultation service on TFS(Team Foundation Serve), Sharepoint, Biztalk, windows server and SqlServer. On the other hand 4ward involves in research and development of latest Microsoft technologies. In the R&D department, there is a team of professional software engineers that work on new software products in the market. I had the opportunity to work with this team during my internship and take part in a huge Cloud project which then became the main topic of this thesis

Objectives

In this project I was involved in the development of the frontend Web Services and frontend Web Client as a software developer.

The primary goal was to make this service available to different clients; therefore the front end had a significant importance.

Considering these facts, we have developed a basic framework in order to hide the presence of the provider and the implementation specifics of every one of them. In this way ,it was possible to create a common layer of reusable software on all of the providers.

- Frontend Web Service: was responsible to respond to all the terminals simulated, web client and distribution of all the information elaborated from all nodes by means of:

o SOAP

o REST con XML/JSON

- Azure Service Bus
- AWS SNS/SQS

- Frontend Web Client: was responsible to simulate the test in a distributed manner from cloud. The website is deployed in different areas. I.e. Ireland, France, Italy, Netherland, using the following frameworks:

- ASP.NET 4
- MVC 3
- Jquery

1. General Architecture

1.1. 4 WPOCs

4 different architectures have been tested:

- Azure
- Amazon
- vCloud
    - with Azure
    - with AWS

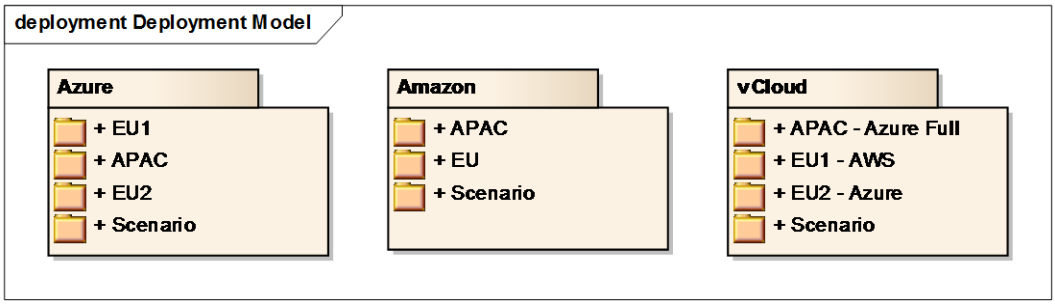

Figure 1. The deployment model of the cloud

1.2. Main components

Here follows a diagram with main components used in the WPOC. Some of them are described in detail the present document.

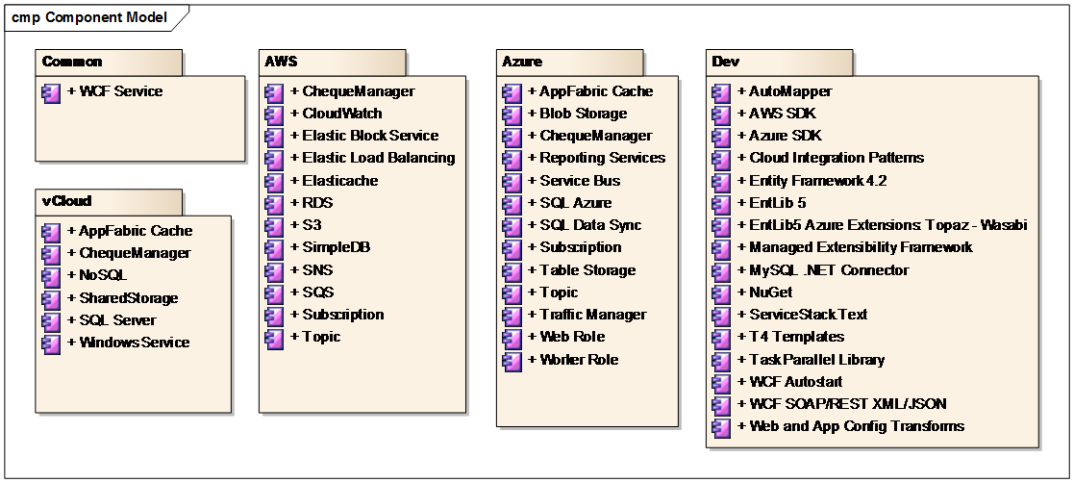

Figure 2. OMP Component Model

1.3. Use Cases and Main Sequence Diagrams

Here follows some screenshot taken from the Enterprise Architect file related to WPOC. For more details please refer directly to it

1.3.1. Use Cases

Here follows a diagram for Use Cases of Azure, but it's the same for all WPOC platforms.

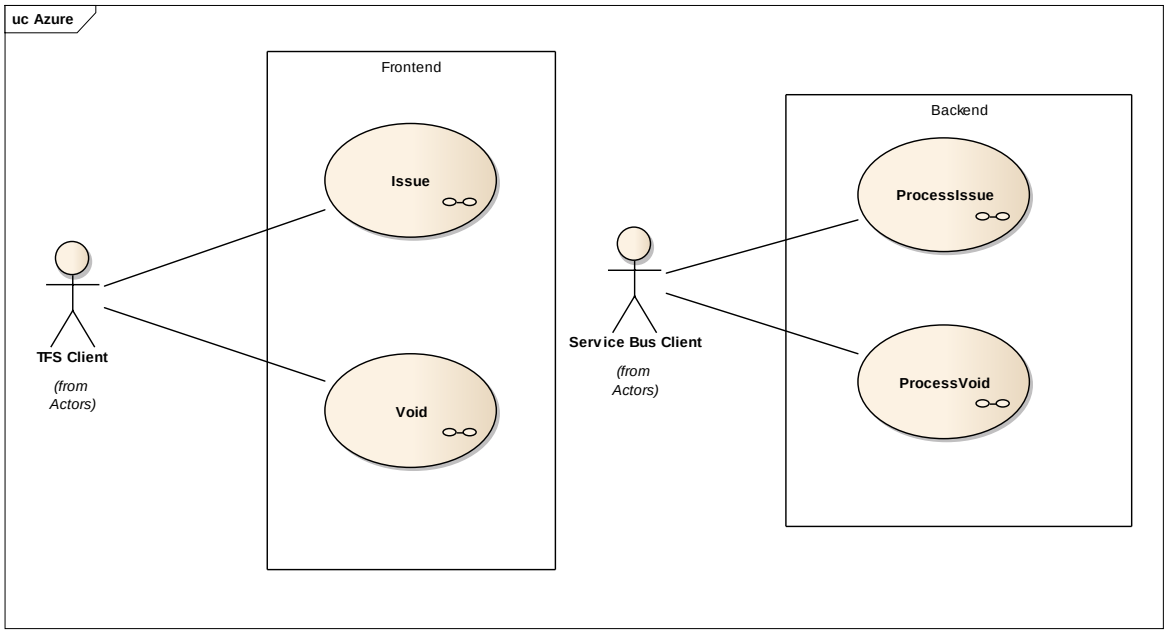

Figure 3. Use Case Diagram of Azure

1.3.2. Sequence Diagrams

Issue

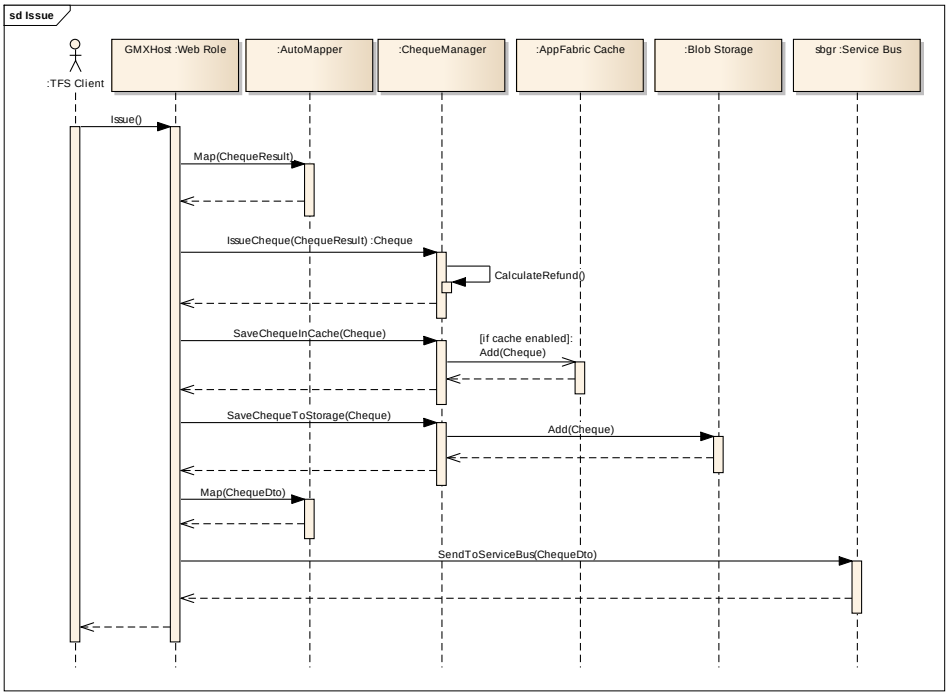

Figure 4. Sequence diagram for an Issue cheque

Void

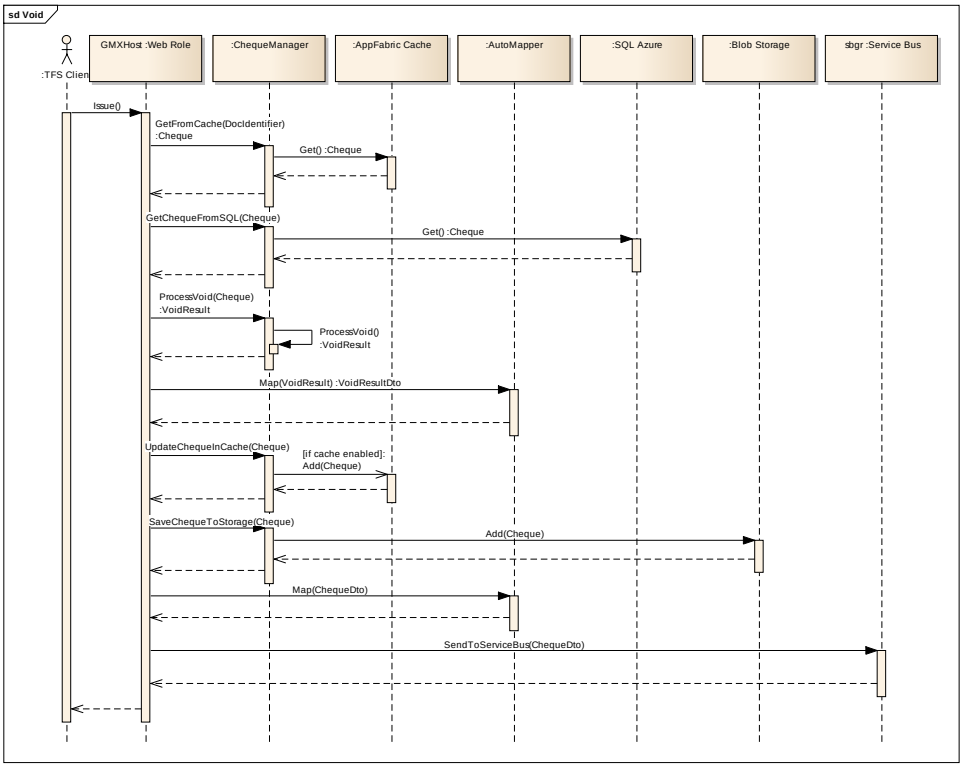

Figure 5. Sequence diagram for a void cheque

2. FROM SOURCE TO BUILD

This paragraph will guide the reader from the configuration of the development machine up to compilation of sources.

2.1. Development machine

In order to be able to download, compile and deploy the sources of the 99xx WPOC a standard Global Blue development machine with Windows 7 and Visual Studio 2010 is needed.
It must be configured to access Global Blue SVN repository.

These additional tools must be installed:

- Azure SDK (Nov 2011 minimum)
  - From: https://www.windowsazure.com/en-us/develop/downloads/
- Windows Server AppFabric (1.1 minimum)
  - From: http://www.microsoft.com/download/en/details.aspx?id=27115
- SQL 2008 R2 Management Studio
  - This can be installed from SQL 2008 R2 Development Edition
- NuGet (1.6 minimum)
  - From: http://www.nuget.org/
- MySQL Connector
  - From: http://dev.mysql.com/downloads/connector/net/
- MySQL Workbench
  - From: http://dev.mysql.com/downloads/workbench/5.2.html

2.2. Get sources and build

Here follows a step by step guide to compile the POC in the configured development machine.

1. Connect to GB Subversion using a client like TortoiseSVN. Repository URL: https://dev.global-blue.com/svn-lab/POC/sp143/
2. Checkout trunk folder
3. Open GB.POC99.Common.sln in GB.POC99.Common folder
4. Right click on Solution and select Enable NuGet Package Restore
5. Compile, in case of errors recompile again up to three times. The errors are related to the fact that NuGet packages are downloaded dynamically from the web during build process
6. Close Solution to free RAM on development machine
7. Open GripsCloudSolution.sln in GripsCloudCoordinator folder
8. Right click on Solution and select Enable NuGet Package Restore
9. Compile, in case of errors recompile again up to three times. The errors are related to the fact that NuGet packages are downloaded dynamically from the web during build process
10. Close Solution to free RAM on development machine

11. Open MessageBusReceiver.sln from MessageBusReceiver
12. Right click on Solution and select Enable NuGet Package Restore
13. Compile, in case of errors recompile again up to three times. The errors are related to the fact that NuGet packages are downloaded dynamically from the web during build process
14. Close Solution to free RAM on development machine

2.3. Additional NuGet packages

The following packages are needed and have been downloaded directly from NuGet during build of solutions:

- Automapper
- AWSSDK .Net
- Entity Framework
- Enterprise Library 5
- Enterprise Library 5 Azure Extensions
- ServiceStack.Text

3. DEPLOYMENT

Following are described the steps in order to deploy all entities involved in POC across all environments.

3.1. Microsoft Windows Azure

Following are described the steps in order to deploy all entities needed for Microsoft Windows Azure. In the POC everything is already configured. For more details please refer to Windows Azure portal directly: https://www.windowsazure.com/en-us/develop/downloads/

3.1.1. Azure Instances

In order to be able to deploy Azure roles you have to configure instances in the cloud.
Here follows the list of steps needed to create instances and other related items like storage, affinity groups, etc.

All operations are performed through Azure web interface:

1. Connect to:
2. http://windows.azure.com

3. Using user gbazure@live.com and related password

Affinity Groups

In order to be able to install all related entities together and improve performance reducing latency, affinity groups are needed.

1. Go to Hosted Services , Storage, Accounts & CDN
2. Select Affinity Groups

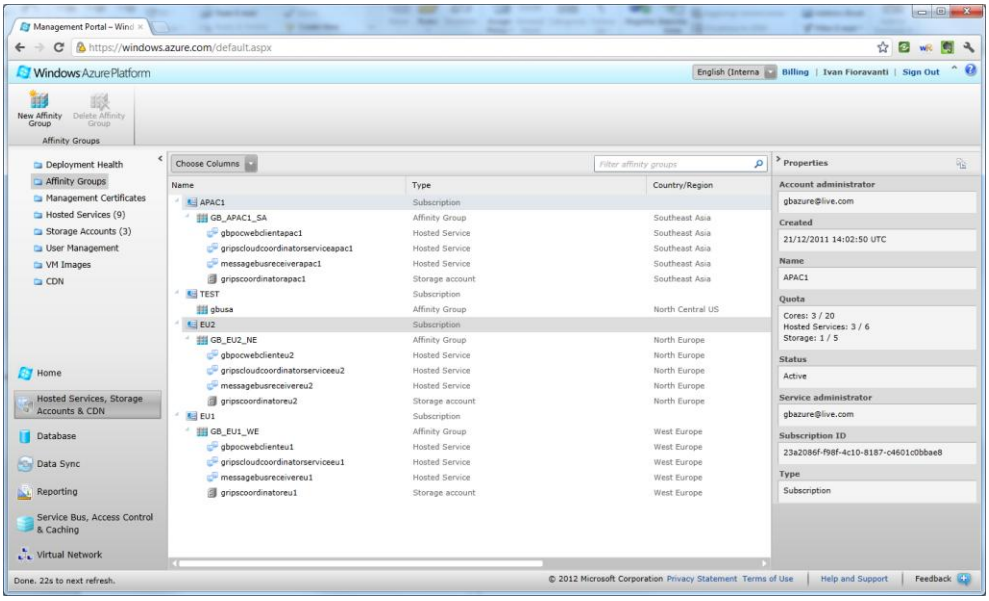

Figure 6

3. Select subscriptions on right pane and click on Create New Affinity group in order to create it.
4. Assign a name and select the region

Storage Accounts

Storage is shared across instances and can be of different types:

- Blob
- Table
- Queue

1. Go to Hosted Services , Storage, Accounts & CDN
2. Select Storage Accounts
3. Select subscription on right pane and click New Storage Account
4. Choose a URL and affinity group created before

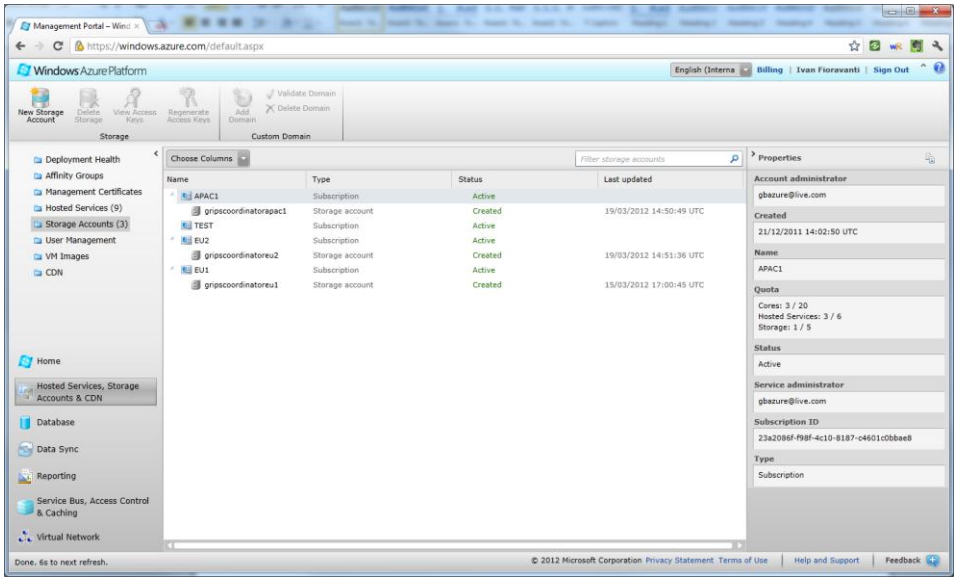

Figure 7

Hosted Services

Azure roles are deployed on hosted services. Each role requires a hosted service to be deployed. Each instance of a hosted service can have a different size from Extra Small to Extra Large. The size is defined during deployment of a package and can be changed when needed. During Hosted Service creation there is no need to define size and you have the choice to deploy directly a package during creation or not.

1. Go to Hosted Services , Storage, Accounts & CDN
2. Select Hosted Services

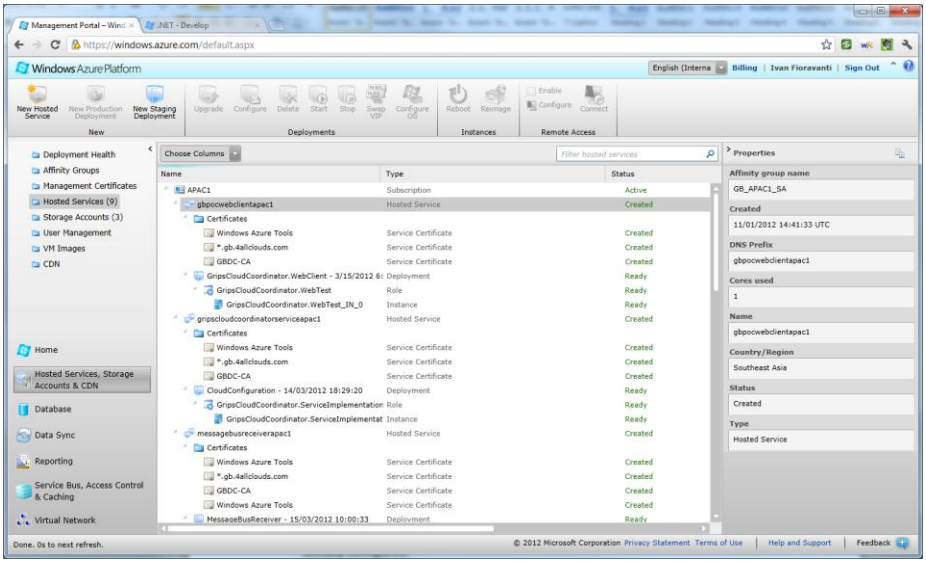

Figure 8

3. Select subscriptions on right pane and click on New Hosted Service in order to create it.

4. Assign a name, URL and choose the affinity group created before, and select Do Not Deploy option

3.1.2. Visual Studio Deployment

Here follows a step by step guide in order to deploy the GripsCloudCoordinator service and MessageBusReceiver service.

Certificate Import

Before to deploy the solution you have to import in your development machine the certificate which is located in the folder
"reposotoryFolder\sp143\trunk\GripsCloudCoordinator\SolutionItems".
Here follows all steps needed to import the "RDP Azure.pfx" certificate:

- Double click on "RDP Azure.pfx", the "Certificate Import Wizard" windows appears and then press "Next".
- In "File To Import" step press Next
- In the password type "Global4cloud" and press Next
- In Certificate Store step select "Place the certificate in the following store", click the Browse button, select Personal and then click Next
- Click Finish to import the certificate.

GripsCloudCoordinator Service

Here follows a step by step guide in order to publish the solution in Azure:

- Open the GripsCloudCoordinator visual studio solution
- Locate the CloudConfiguration web role, right-click on it and then select Publish as shown in the Figure 1.

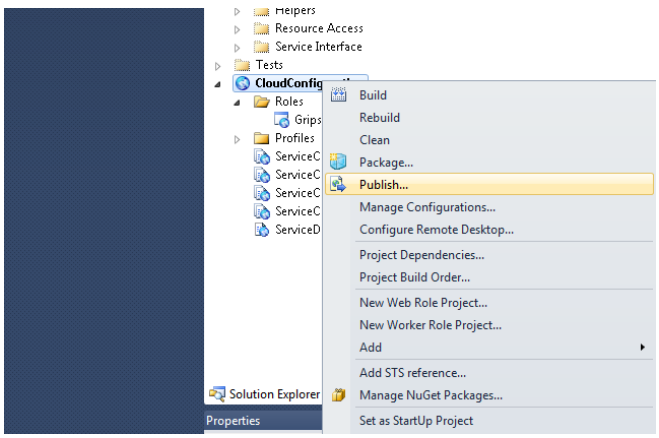

Figure 9

(Figure 1)

- The “Publish Windows Azure Application” wizard will appear.
- In the “Target Profile” dropdown select the Windows Azure profile for the environment where you want to deploy the application:
    - EU1 → West Europe
    - APAC1 → Asia Pacific

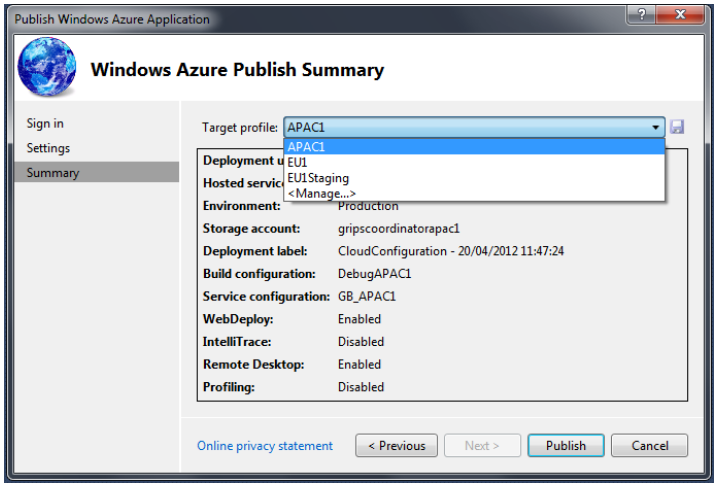

Figure 10

- If you want to enable the IntelliTrace feature, click the “Previous” button, select Advanced Settings tab and then flag the “Enable IntelliTrace” check box.

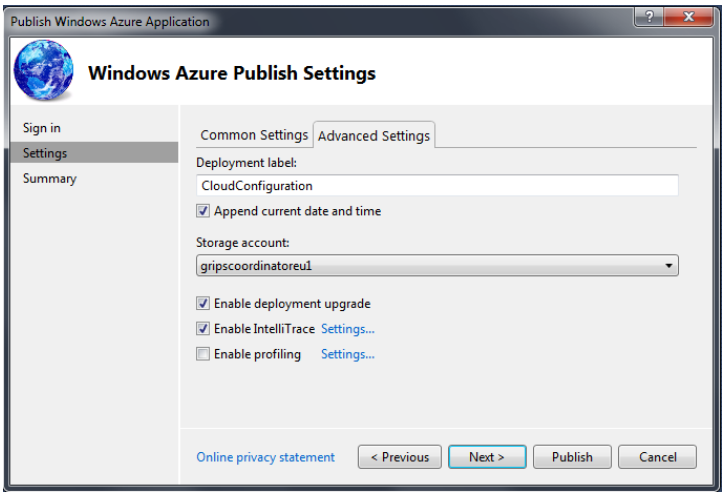

Figure 11

- Click the “Publish” button in order to deploy the solution.

MessageBusReceiver service

Here follows a step by step guide in order to publish the solution in Azure:

- Open the MessageBusReceiver visual studio solution
- Locate the MessageBusReceiver worker role, right-click on it and then select Publish as shown in the Figure 4.

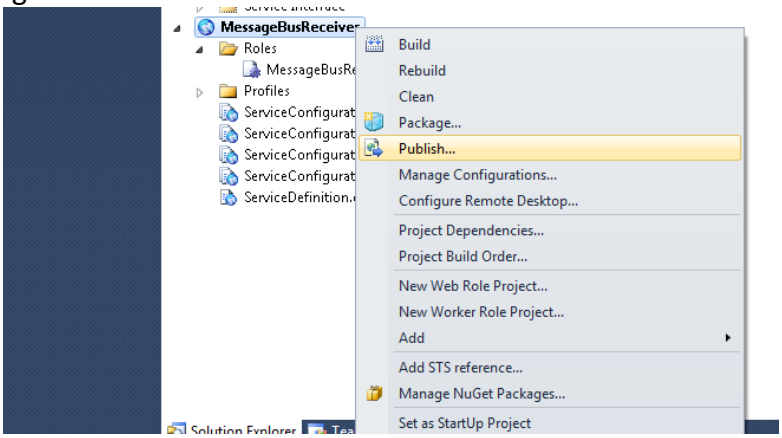

Figure 12

- The "Publish Windows Azure Application" wizard will appear.
- In the "Target Profile" dropdown select the Windows Azure profile for the environment where you want to deploy the application:
    - messagebusreceivereu1Production → West Europe
    - messagebusreceiverapac1Production → Asia Pacific

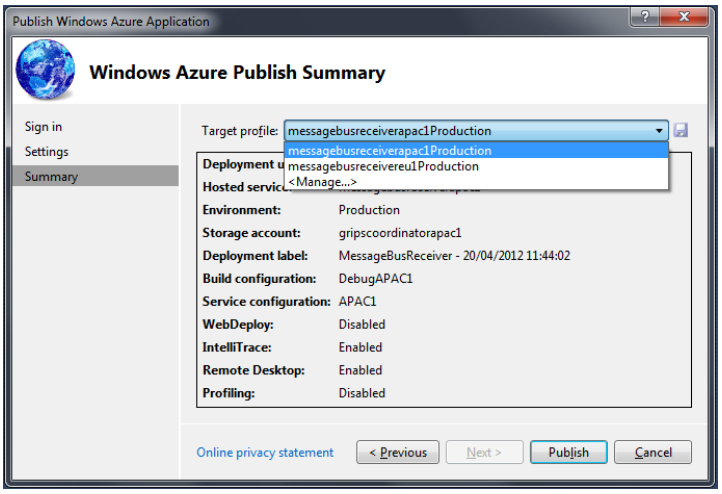

Figure 13

- If you want to enable the IntelliTrace feature, click the "Previous" button, select Advanced Settings tab and then flag the "Enable IntelliTrace" check box.

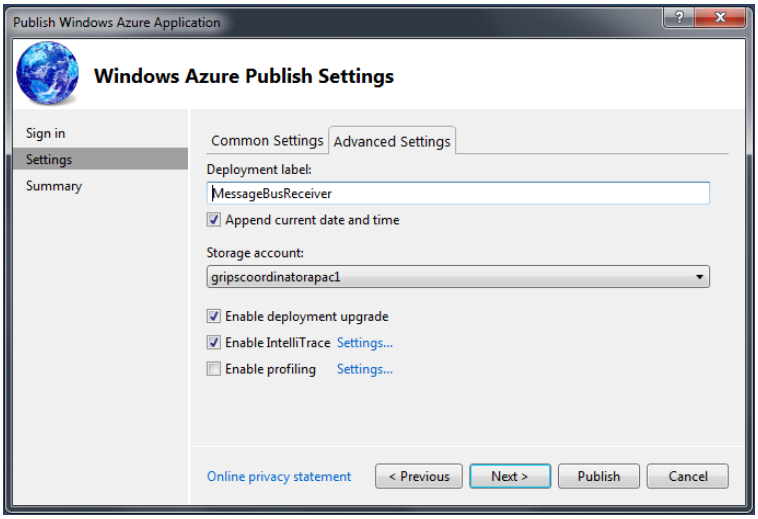

Figure 14

- Click the "Publish" button in order to deploy the solution.

3.1.3. Database SQL Azure

Here follows the list of steps needed to create the database on Microsoft SQL Azure

Creating the Database Server in the Cloud

In order to create the database in the cloud, you have to access to Azure Management Portal and follow these steps:

- Once authenticated, go to "Database".
- Choose the right subscription where you want to create the database server.
- Once identified click on "Create"
- Select the region where you want to create the database (in this POC we have to create one database per region where enterprise services are deployed, so choose from North Europe, West Europe and Southeast Asia) and click Next.
- Enter the Administrator Login and Password (we have used "gbsa" as username and Hunt4cloud as Password. If you want to change this configuration remember to change it also in web.config and app.config ConnectionStrings section for GripsCloudCoordinator and MessageBusReceiver services) and press Next.
- The firewall rules configuration window appears. Here you can enter IP Addresses and ranges that are allowed to access to the database server. To add a new Rule, press the Add button; in the Rule Name type a friendly name to identify the rule purpose; Type the IPs Addresses in "IP rage start" and "IP range end" to allow a range or if you want to allow a single IP Address type the same IP in both (remember that for test purpose, running the services on your local machine, the service will use the public IP Address to access to SQL Azure server, so you have to add this IP in the rules that you can find in the "Your current IP address"). Check the "Allow other Windows Azure services to access this server" flag and click Finish.

- Leave the Azure Management Portal opened.

Creating database and table in the new Azure Database Server

Once the Database Server is created, you have to create the database "gripcloud" following these steps:

- From Azure Management Portal, select the database server just created and click Create in the Database ribbon group.
- In the Database name type gripscloud and click ok.
- Now the database is created so we have to create the tables.
- Creating tables using Visual Studio 2010:
    - Open MessageBusReceiver or GripsCloudCoordinator or GB.POC99.Common solution
    - In Solution Explorer go to Source → Resource Access → GB.POC99.Common.DataAccess.Azure → Scripts → double click on CreateDB.sql
    - Press the Connect icon (Next to "New Query" button) and the "Connect to Database Engine" will appear.
    - In the Server name type the complete address that you can find in Azure Management Portal, clicking the database server created previously. The name is the "Fully Qualified DNS Name" in the properties column on the right.
    - Choose the SQL Server Authentication instead of Windows Authentication and enter the credentials created previously. Press Connect.
    - Now the Query is connected to the database server, choose the gripscloud database in Database dropdown box next to Connect icon.
    - Click Execute Query icon next to Database dropdown.

These steps must be followed in order to create the database server and the database for each region to be configured.

3.1.4. SQL Azure Reporting Services

In the POC SQL Azure Reporting Services have been used for testing purposes and a simple dashboard for analyzing POC performance have been created.

1. Go to Reporting
2. Click Create a New SQL Azure Reporting Services

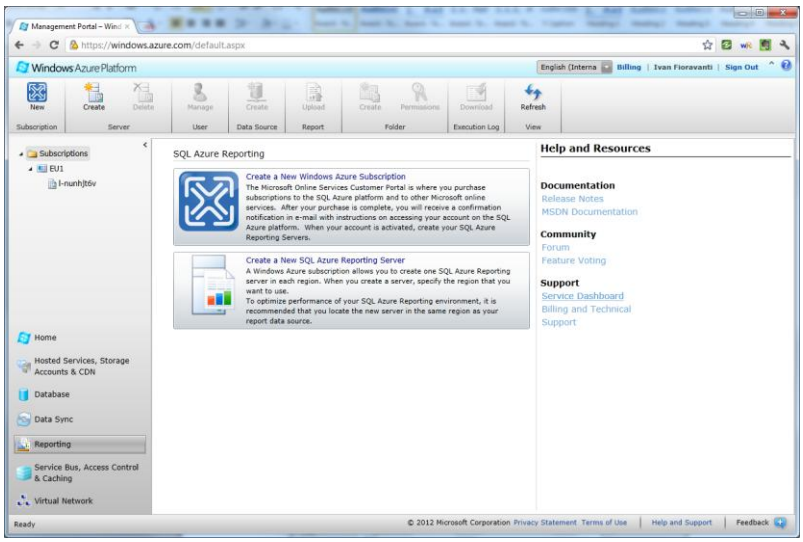

Figure 15

3. Select a subscription and the region where you want to place reports.
4. Assign Admin User and password

Once server is created you are able to upload report definitions and data sources.

1. Select the newly created server on top left part of the web interface
2. And toolbar actions above will be activated

Here you can find a useful Getting Started Guide: http://msdn.microsoft.com/en-us/library/windowsazure/gg430129.aspx

3.1.5. Service Bus & Cache

Service Bus and Cache are two additional components used in the POC.

- SB: used to integrate on-cloud and on premise data centers
- Cache: to improve performance on frontend

In order to create both of them, follow procedure below:

1. Go to Service Bus, Access Control & Caching
2. Select Service Bus in the top left part of the web interface

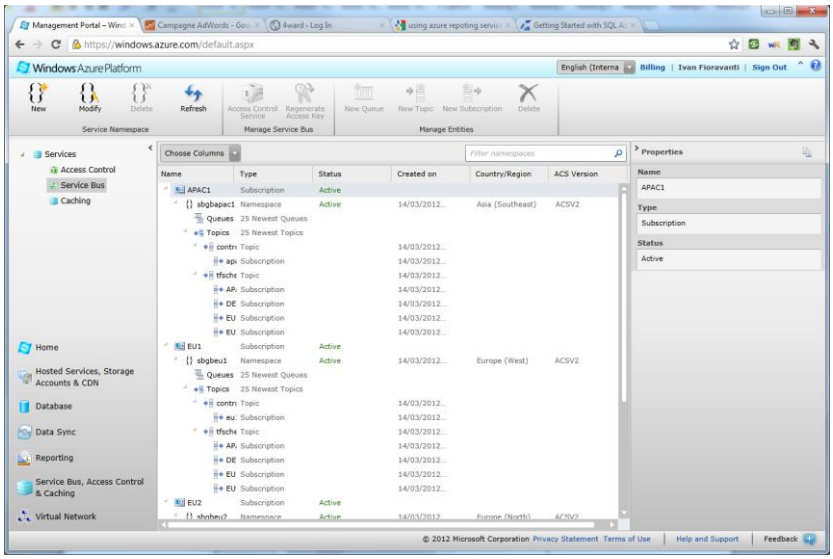

Figure 16

3. Click New
4. Insert a namespace to be used, select Region, select Cache checkbox and size

Topics and Subscriptions will be created by backend based on the configuration file.

Here you can find a useful Getting Started Guide: http://msdn.microsoft.com/en-us/library/windowsazure/gg430129.aspx

3.2. Amazon AWS

Following are described the steps in order to deploy all entities needed for Amazon AWS. In the POC everything is already configured. For more details please refer to Amazon AWS portal directly: http://aws.amazon.com/

Amazon AWS is not a full featured PaaS like Azure, therefore involvement of IT Pro is needed in order to have virtual machines correctly configured and ready for deployment before proceeding.

In the POC environment everything is already configured, in order to deploy a new infrastructure, please refer to system documentation delivered at the end of POC.

3.2.1. AWS EC2

EC2 is the AWS feature that gives you the possibility of starting new virtual machines in the cloud.

This process is handled by System & Operation team, but in case of problem you have to be aware of some possible activities:

- You are always able to connect through RDP to remote machines
- Basic software needed to run the POC is already installed in the template used to start new VM (i.e. MySql driver, AppFabric, etc.)

Autostart feature

The main issue found so far during usage of AWS is related to Autostart feature not always working out of the box.

If POC services are not working properly first of all check following things:

- Connect to Amazon AWS portal
- Select EC2 tab
- Select Instances on the left pane

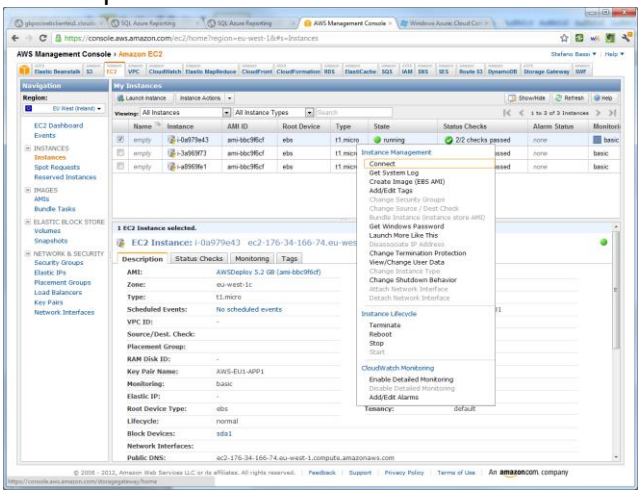

Figure 17

- Select the Backend instance on the right pane and from Context Menu select Connect
- Once connected
- Open IIS Manager

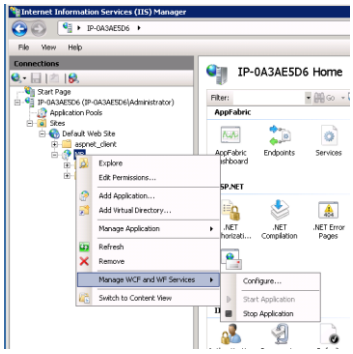

Figure 18

- From Context Menu of the web site hosting Backend select: Manage WCF and WF Services → Configure
- Enable Auto-Start and Press Yes to apply all needed changes

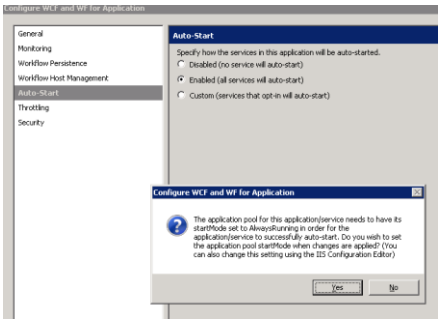

Figure 19

3.2.2. AWS RDS

On Amazon side of the POC relational database used is RDS configured with MySQL. Here follows the procedure used to initialize database.

1. From AWS Console
2. Select RDS
3. Select Launch DB Instance

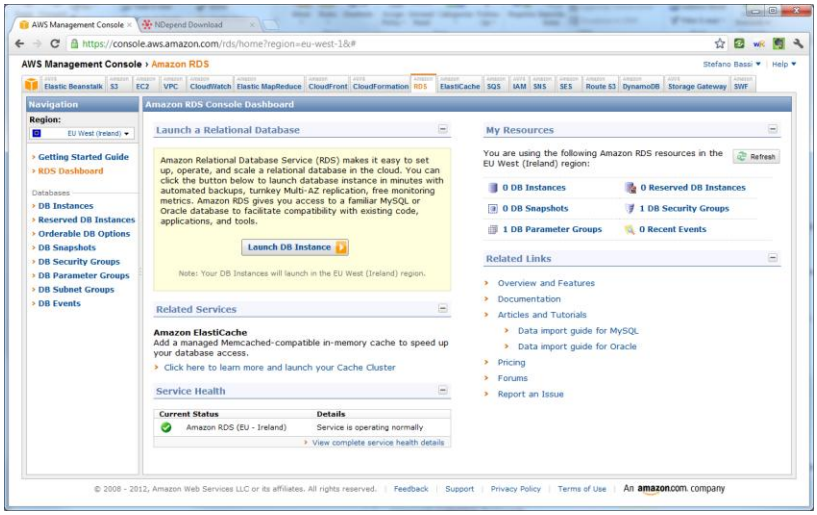

Figure 20

4. Select MySQL
5. Configure it as per following screenshot

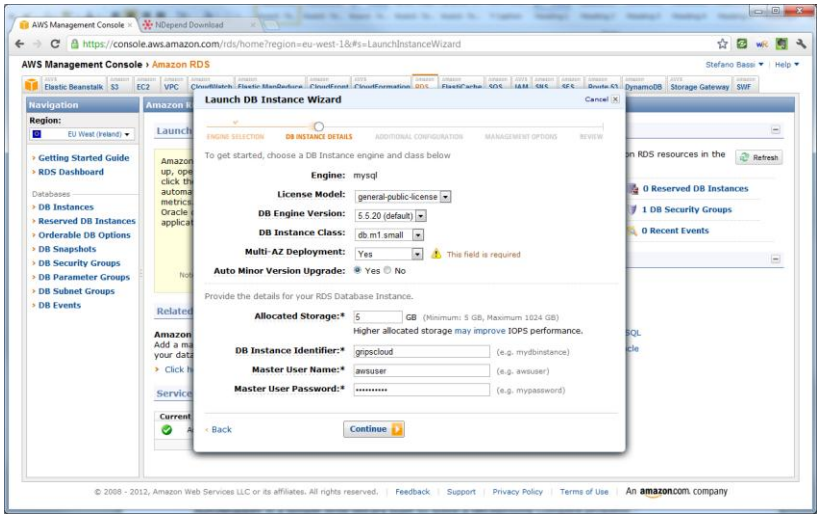

Figure 21

6. Assign database name
7. Disable backup
8. Finish configuration

Database creation

In order to create database you have to install MySQL Tools on development machine.

1. From Amazon AWS RDS select the instance created and copy the endpoint value

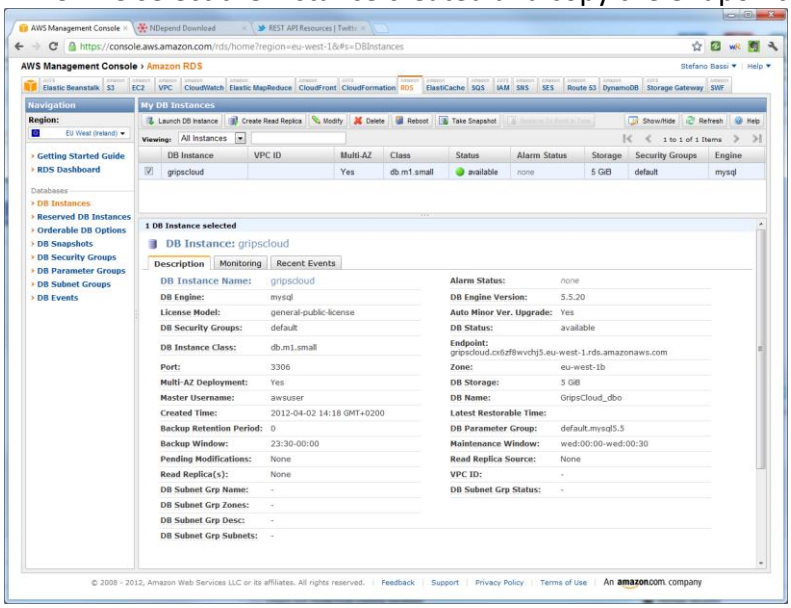

Figure 22

2. Launch MySQL Workbench
3. Click New Connection
4. Define Name, Endpoint, User and Password and click Test Connection
5. If you receive error during connection please ask to your IT Pro to configure firewall

settings on Amazon AWS RDS instance in order to allow access

6. Connect to the newly created instance double clicking it
7. Select the GripsCloud_dbo database on the left pane
8. From GB.POC99.Common.DataAccess.AWS project open the CreateDB script and copy its content
9. Paste the content in the right pane of MySQL Workbench and press Execute

3.2.3. AWS Elasticache

In the POC Elasticache has been configured only in AWS European datacenter in order to test its performance and verify code needed to use it.

To configure it properly, please follow steps below:

1. From AWS Console
2. Select Elasticache
3. Select Launch Cache Cluster

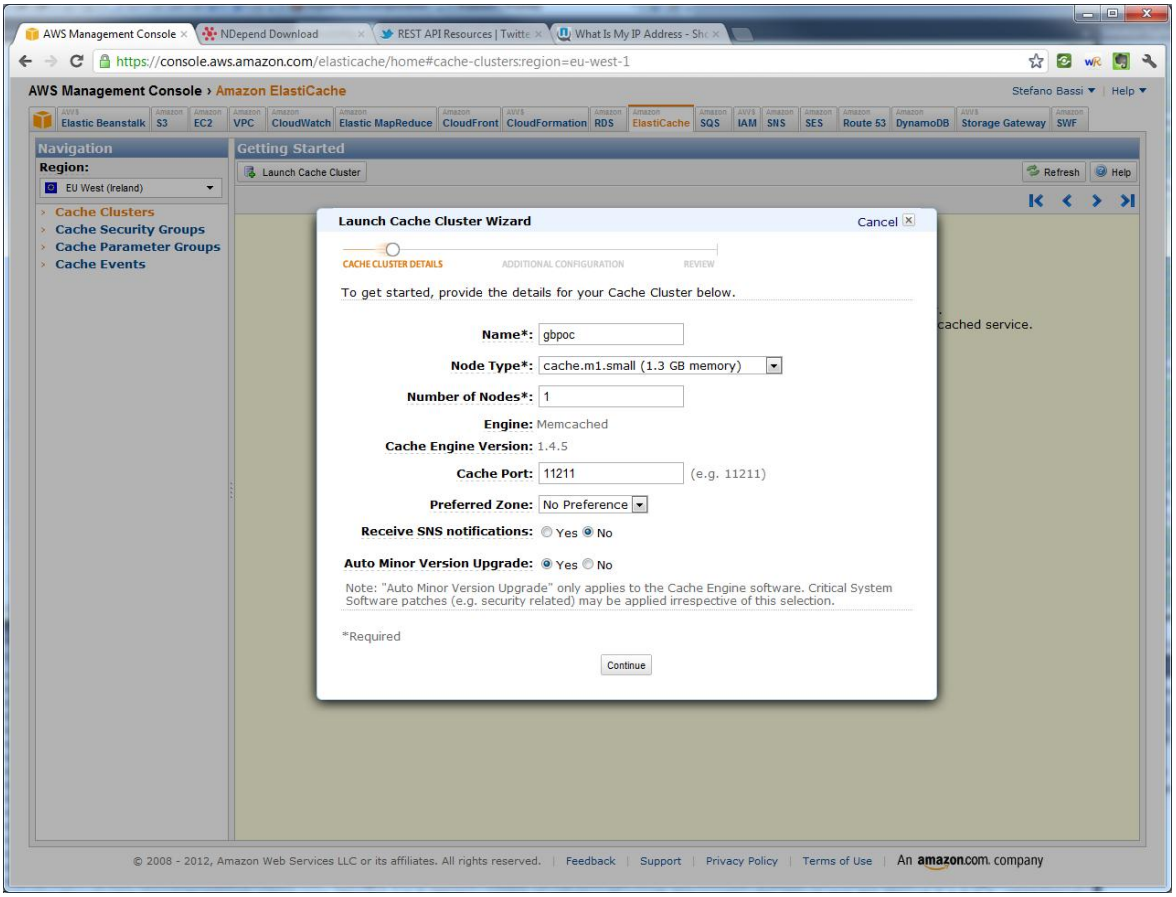

Figure 23

4. Give it a name a 1 node (for testing purpose only) continue until end of wizard
5. In order to find the IP address of the cluster node you have to connect in RDP in any of

the EC2 instance created and ping from there[1]

3.2.4. AWS SNS/SQS

On Amazon side of the POC connection between frontend – backend and the various data centers is done through SNS/SQS.

In order to create them follow the procedure below:

1. From AWS Console
2. Select RDS
3. Select Launch DB Instance

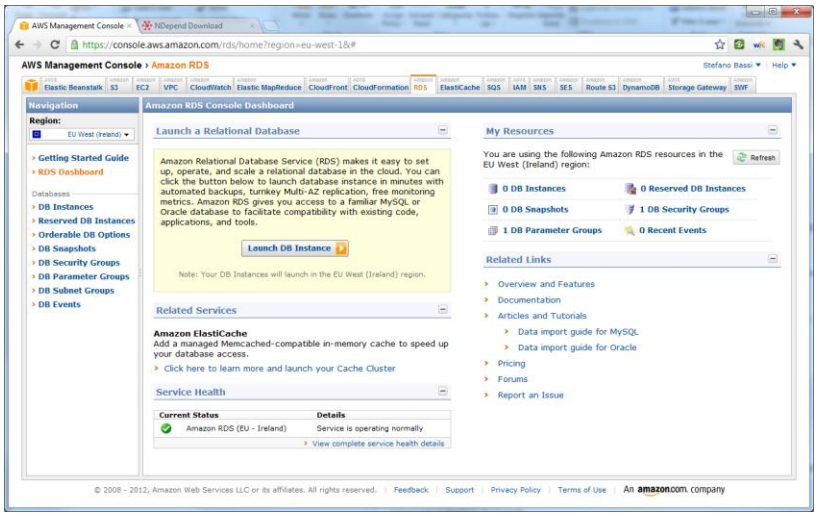

Figure 24

4. Select MySQL
5. Configure it as per following screenshot

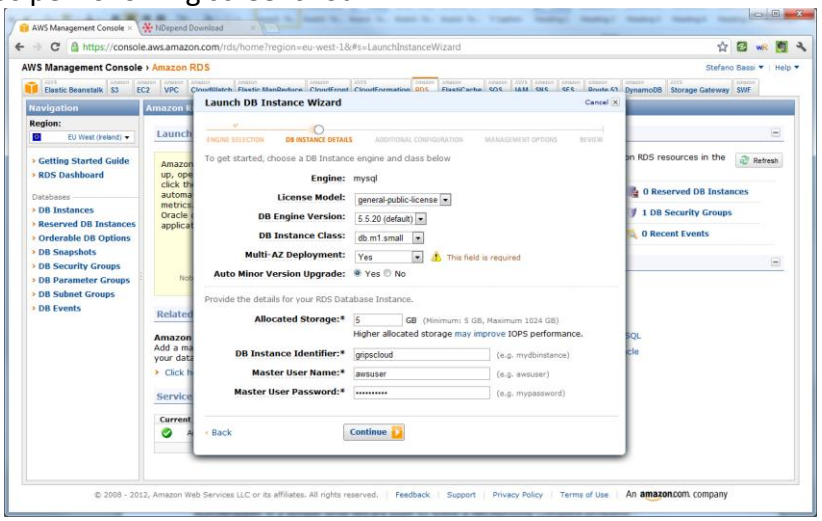

Figure 25

[1] This is needed for the Enyim .NET client used that does not support DNS name resolution

6. Assign database name
7. Disable backup
8. Finish configuration

4. Cloud Integration

4.1. Automapper

AutoMapper is a simple little library built to solve a deceptively complex problem - getting rid of code that mapped one object to another.
Initially we had some performance issues related to wrong usage of the library, the author of the library has given us many information and with community help we solved all problems.

Pros:
- Very easy to be used and customized
- Fast

Cons:
- Time consuming for complex configuration

Links:
- Home: http://automapper.org/
- Source: https://github.com/AutoMapper/AutoMapper
- NuGet: http://nuget.org/packages/AutoMapper

4.2. Entity Framework 4.2

Entity Framework is the official ORM implementation done by Microsoft. With version 4.2 it reached a high level of maturity and it currently supports 3 different usage methods:
- Database First
- Model First
- Code First

Personal comments: Code First has been introduced in the last versions and seems still immature. The main problem is related to upgrade from a version to the next one that is quite complex is some scenario. In 4ward we used both Database and Model First and both are working really well.

Pros:
- Fully supported by Microsoft

- Integrated in IDE
- Complete and useful User Interface

Cons:

- Performance hit compare to direct access to database

Links:

- Home: http://blogs.msdn.com/b/adonet/
- Source: N/A
- NuGet: http://nuget.org/packages/entityframework

4.3. Enterprise Library 5

Enterprise Library is a framework from Microsoft that helps to leverage common Enterprise Scenario. Giving to developers a set of additional building blocks built on top of .NET Framework 4 like:

- Caching
- Logging
- Exception Handling

Azure Extensions contain 2 additional building blocks:

- Wasabi: this building blocks are used to autoscale Azure instances that is a capability offered by Amazon AWS through CloudFront, but not yet available in the mainstream release of Azure. This has not been implemented in the POC, but has been tested by 4ward.
- Topaz: this is use to handle silently Transient Fault that can occur in the cloud. This has been used in the WPOC for accessing storage, cache, Service Bus and SQL Azure.

Pros:

- Easy to use
- New configurator is much improved and now easier to be used

Cons:

- Logging is slower than other libraries present on the market

Links:

- Home: http://msdn.microsoft.com/en-us/library/ff632023.aspx
- Source: http://www.microsoft.com/download/en/details.aspx?id=15104
- NuGet: http://nuget.org/packages?q=entlib

4.4. NuGet

NuGet is a Visual Studio extension that makes it easy to install and update open source libraries and tools in Visual Studio. These free developers from the manual activities related to downloading, referencing and configuring common libraries available on the market.

Pros:

- Centralized repositories of most common .NET Open Source libraries
- Fast and easy to use
- Very easy to upgrade libraries to latest version

Cons:

- Some attention must be placed in source control repositories handling with NuGet

Links:

- Home: http://www.nuget.org/
- Source: N/A

4.5. JSON Serialization ServiceStack.Text

JSON serialization is lighter than XML in terms of size and with ServiceStack.Text open source library it's also lightning fast. This gives the possibility of storing and retrieiving data using JSON format in various situations like:

- REST web service
- Storage
- NoSQL
- Cache

ServiceStack in reality is a full and very well done Opensource .NET and Mono REST Web Services framework.

Comments: in 4ward we have tested multiple JSON libraries, including the official from Microsoft, but after many tests we have verified that ServiceStack.Text is truly the most impressive from performance point of View.

Pros:

- Lightning fast
- Easy to be used

Cons:

- Not large user base

Links:

- Home: http://servicestack.net/
- Source: https://github.com/ServiceStack/ServiceStack.Text
- NuGet: http://nuget.org/packages/ServiceStack.Text

4.6. Web and App config transforms

Web config transforms have been introduced in Visual Studio 2010 and can be used to create a set of basic settings in the main web.config file and then a config transform for each

Solution/Project configuration defined (i.e. Debug/Release). This feature has been used deeply in the POC in order to create config files to be used for each environment and provider tested (3 for Azure, 2 for AWS, 3 for vCloud).

App config transforms are not present out of the box, but we have found a manual hack/workaround that gives the possibility of using this great feature also in case of Azure Worker Role or Windows Services, that do not use web.config files.
This trick has been applied to MessageBus.Host.Azure.csproj:

```
<ItemGroup>
    <None Include="app.config">
      <SubType>Designer</SubType>
    </None>
    <None Include="app.Debug.config">
      <DependentUpon>App.config</DependentUpon>
    </None>
    <None Include="app.DebugEU1.config">
      <DependentUpon>App.config</DependentUpon>
    </None>
    <None Include="app.DebugEU2.config">
      <DependentUpon>App.config</DependentUpon>
    </None>
    <None Include="app.DebugAPAC1.config">
      <DependentUpon>App.config</DependentUpon>
    </None>
    <None Include="packages.config" />
  </ItemGroup>


<UsingTask TaskName="TransformXml"
AssemblyFile="$(MSBuildExtensionsPath)\Microsoft\VisualStudio\v10.0\Web\Microsoft.Web.P
ublishing.Tasks.dll" />
  <Target Name="AfterCompile" Condition="exists('app.$(Configuration).config')">
```

```
    <!-- Generate transformed app config in the intermediate directory -->

    <TransformXml Source="app.config"
Destination="$(IntermediateOutputPath)$(TargetFileName).config"
Transform="app.$(Configuration).config" />

    <!-- Force build process to use the transformed configuration file from now on. -->

    <ItemGroup>

      <AppConfigWithTargetPath Remove="app.config" />

      <AppConfigWithTargetPath
Include="$(IntermediateOutputPath)$(TargetFileName).config">

        <TargetPath>$(TargetFileName).config</TargetPath>

      </AppConfigWithTargetPath>

    </ItemGroup>

  </Target>
```

This leads to creation of transformed app.config files in the build output directory.

In order to be able to deploy this file in Azure, additional manual changes have been applied to the Azure project MessageBusReceiver:

```
  <Target Name="CopyWorkerRoleConfigurations"
BeforeTargets="AfterPackageComputeService">

    <Copy
SourceFiles="..\MessageBusReceiver.Host.Azure\obj\$(Configuration)\MessageBusReceiver.H
ost.Azure.dll.config"
DestinationFolder="$(IntermediateOutputPath)\MessageBusReceiver.Host.Azure"
OverwriteReadOnlyFiles="true" />

  </Target>
```

Pros:

- Simplifies creation and maintenance of multiple config files

Cons:

- It takes some time to learn how to apply transform

Links:

- Home: http://msdn.microsoft.com/it-it/library/dd465326.aspx
- Source: N/A
- NuGet: N/A

4.7. WCF SOAP and REST(JSON/XML) bindings

REST is gaining a lot of traction in the last years. It's easy to be consumed by any device through simple GET or POST http requests. In the POC we have implemented all possible kind of web service interfaces using WCF:

- SOAP
- REST with XML
- REST with JSON

We have configured WCF in order to use and expose a single service contract on the same address but using different binding. To do this we added /soap and /rest at the end of it. In this way Global Blue is able to create both SOAP and REST API in a single service.

Pros:

- Easier development and maintenance
- Higher compatibility with multiple devices

Cons:

- More effort on WCF configuration side
- More attention to security (SOAP and REST uses different patterns)

Links:

- Home: http://msdn.microsoft.com/en-us/library/dd456779.aspx
  - Rest: http://msdn.microsoft.com/en-us/netframework/cc950529
- Source: N/A
- NuGet: N/A

4.8. Windows Server AppFabric

Windows Server AppFabric is a set of integrated technologies that make it easier to build, scale and manage Web and composite applications that run on IIS.

Services offered are:

- Hosting
- Caching

Hosting has been used in a simple way in the POC without advanced monitoring or persistence database for Workflow Foundation (not used in POC). The autostart feature has been used in order to replace Windows Service hosted with Appfabric hosted WCF services for vCloud and Amazon AWS backend.

Caching has been used in the vCloud POCs and initially we had some performance problems. In the last version of POC code it's now stable and faster, but it's clear that is a 1.0 version.

4.8.1. WCF Autostart feature

Autostart features are deserved a separate paragraph because of its importance. It gives the possibility of autostarting WCF services hosted in WAS and Appfabric, before receiving the first call from a client. This leads to:

- Higher speed of the service
- Automatic start of WCF Services in IIS like when hosted on Windows Service

In order to use Autostart following configuration must be applied to IIS:

- Net.pipe binding must be enable at site level and at application level
- WCF Service must be configured for autostart (Enabled)
- AppPool must be configured to support autostart (this is automatically done on the first autostart activation in an appPool)

Comments: This feature has been introduce in Windows Server AppFabric 1.0 and it's still not complete from our point of view. The service can be started, even if some changes in the code are required in order do it correctly and to initialize various components in the right way (i.e. MEF)

Pros:

- Windows Service can be replaced 1
- WCF in autostart are easier to be maintained (install, upgrade)

Cons:

- Bug on stopping services (autostart services hosted in a different appPool, requiring a dedicated web site for each of them), that will be fixed in the next Appfabric 1.1 release.

Links:

- Home: http://msdn.microsoft.com/en-us/windowsserver/ee695849
- Source: N/A
- NuGet: N/A

4.9. Managed Extensibility Framework

The Managed Extensibility Framework or MEF is a library for creating lightweight, extensible applications. It allows application developers to discover and use extensions with no configuration required. It also lets extension developers easily encapsulate code and avoid fragile hard dependencies. MEF not only allows extensions to be reused within applications, but across applications as well.

Comments:

- MEF has been used deeply in the POC, in order to be able to use a sort of plugin architectures for reducing changes of code with each cloud provider used. We have created an abstract layer for each vCloud provider element used (i.e. S3, BlobStorage, SNS, ServiceBus, etc.) and the real implementation is loaded through MEF. This gives the

possibility of extending the base POC architecture with new cloud providers plug in without having to change the business logic code.
- Even on business logic side, the processing logic of messages in the backend has been implemented through MEF. A practical example can be enabling the possibility of dynamic load of new business logic classes from remote storage (S3 or Blob Storage) without having services interruption, using dynamic background worker processes.
- Multi thread: initially we had experienced some issues with multiple threads using MEF, but at the end we found the correct implementation pattern. Creation of catalogs is the most expensive in terms of performance hit and can be done once initially or when a new dll is found. Container creation is very fast and must be done once for each thread. In this way each thread will create new objects from its own container, avoid multi thread synchronization issues.

Pros:
- Dependency Injection + Extensibility
- Very easy to be used once learnt

Cons:
- It takes some time to learn how to use it
- Does not work with generics in Import/Export (.NET 4.5 will address this limitation), but workaround can be easily used.

Links:
- Home: http://msdn.microsoft.com/en-us/library/dd460648.aspx
- Source: N/A
- NuGet: N/A

4.10. Azure Integration

Windows Azure is an open and flexible cloud platform that enables you to quickly build, deploy and manage applications across a global network of Microsoft-managed datacenters. You can build applications using any language, tool or framework. And you can integrate your public cloud applications with your existing IT environment.
In this POC we used several Windows Azure components listed below. Some features could result redundant, but it is only for test purpose in order to have a complete view of possible and best solutions.

4.10.1. Traffic Manager

Windows Azure Traffic Manager enables you to manage and distribute incoming traffic to your Windows Azure hosted services whether they are deployed in the same data center or in different centers across the world.
Traffic routing occurs as a result of policies that you define and that are based on one of the following criteria:
- **Performance:** traffic is forwarded to the closest hosted service in terms of network

latency
- **Round Robin:** traffic is distributed equally across all hosted services
- **Failover:** traffic is sent to a primary service and, if this service goes offline, to the next available service in a list

Comments:
Pros:
- It gives you possibility of creating failover scenario (i.e. Refunding Use Case) or redirect of clients to the nearest DC (i.e. Issuing)

Cons:
- With the current version Traffic Manager can be used only for services within the same Azure subscription. There are some default limits on each subscription in terms of number of cores, instances, etc. but these can be overcome contacting Microsoft Support

Links:
- http://msdn.microsoft.com/en-us/gg197529

4.10.2. Service Bus

The Service Bus provides secure messaging and relay capabilities that enable building distributed and loosely-coupled applications in the cloud, as well hybrid application across both private and public clouds. It supports multiple messaging protocols and patterns and handles delivery assurance, reliable messaging and scale for your applications.
In this POC we used this feature in order to communicate from FE to BE. Initially we have implemented the flow in order to send the entire Cheque object. The problem is that Amazon AWS does not have the same feature, therefore we have also implemented to send only the result identifier and retrieve the Cheque Result previously saved by FE.

Pros:
- Easy to use
- Fast messaging
- Enterprise oriented
- Messages can stay in queues for unlimited time

Cons:
- Maximum size of message is 256Kb, larger messages should be handled in other ways

Links:
- http://www.windowsazure.com/en-us/home/tour/service-bus

4.10.3. Blob storage

Windows Azure Blob storage is a service for storing large amounts of unstructured data that can be accessed from anywhere in the world via HTTP or HTTPS.
Common uses of Blob storage include:
- Serving images or documents directly to a browser

- Storing files for distributed access
- Streaming video and audio
- Performing secure backup and disaster recovery
- Storing data for analysis by an on-premise or Windows Azure-hosted service

You can use Blob storage to expose data publicly to the world or privately for internal application storage. The last one is the way we used this feature.
The storage service offers two types of blobs, *block blobs* and *page blobs*.
Block blobs let you upload large blobs efficiently. Block blobs are comprised of blocks, each of which is identified by a block ID. You create or modify a block blob by writing a set of blocks and committing them by their block IDs. Each block can be a different size, up to a maximum of 4 MB. The maximum size for a block blob is 200 GB, and a block blob can include no more than 50,000 blocks.
Page blobs are a collection of 512-byte pages optimized for random read and write operations. To create a page blob, you initialize the page blob and specify the maximum size the page blob will grow. To add or update the contents of a page blob, you write a page or pages by specifying an offset and a range that align to 512-byte page boundaries. A write to a page blob can overwrite just one page, some pages, or up to 4 MB of the page blob. Writes to page blobs happen in-place and are immediately committed to the blob. The maximum size for a page blob is 1 TB.
As wrote previously in 4.10.2 section, instead of sending the entire Cheque Result through Service Bus, we implemented the flow in order to save it in the Blob storage, then send only the result identifier and once the message was received from BE, it retrieves the result from Blob storage using the identifier.

Pros:
- Easy to use
- **Fault-tolerance:** Blobs on Windows Azure are replicated three times in the same data center for resiliency against hardware failure. They are also geo-replicated between two data centers 100s of miles apart from each other on the same continent, to provide additional data durability in the case of a major disaster, at no additional cost.

Cons:
- Access writes not so fast in the web role. Additional tests will have to be performed before the implementation, because worker role are able to save faster using the same code base

Links:
- http://www.windowsazure.com/en-us/home/tour/storage

4.10.4. Table Storage

Table Storage is used by applications requiring storing large amounts of data storage that need additional structure. While a table stores structured data, it does not provide any way to represent relationships between the data, sometimes called a NoSQL database.

Tables store data as collections of entities. Entities are similar to rows. An entity has a primary key and a set of properties. A property is a name, typed-value pair, similar to a column. An entity always has PartitionKey (PK), RowKey (RK) and Timestamp properties. The developer is responsible for inserting and updating the values of **PartitionKey** and **RowKey**. The server manages the value of **Timestamp**, which cannot be modified.
Tables are partitioned to support load balancing across storage nodes. A table's entities are organized by partition. A partition is a consecutive range of entities possessing the same partition key value. The partition key is a unique identifier for the partition within a given table, specified by the PartitionKey property. The partition key forms the first part of an entity's primary key. The partition key may be a string value up to 1 KB in size.
You must include the PartitionKey property in every insert, update, and delete operation.
At the application design time we considered to use it to store the entire Cheque Result, but we immediately faced to a limitation which is that you can't store complex types. So we decided to store a simple Cheque Result with main properties, leaving the complex type to SQL Azure.

Pros:

- Fast write access.
- Easy to use
- Table Storage can be partitioned. Each partition of Table Storage can be moved to a separate server by the Azure controller thereby reducing the load on any single server and improving the read access performance.
- Supports very large dataset

Cons:

- You can't use complex types and relationships
- It's a simple Key-Value NoSQL storage
- In order to perform indexing you have to store same data multiple times using differet PK, RK schemes.

Links:

- http://msdn.microsoft.com/en-us/library/windowsazure/dd179338.aspx
- http://www.windowsazure.com/en-us/home/tour/storage

4.10.5. SQL Azure

SQL Azure is a highly available and scalable cloud database service built on SQL Server technologies. With SQL Azure, developers do not have to install, setup or manage any database. High availability and fault tolerance is built-in and no physical administration is required. SQL Azure is a managed service that is operated by Microsoft and has a 99.9% monthly SLA.
We used this feature in order to store the Cheque Result and all its related child items building a complex relationship data structure.

Pros:

- No physical administration required, SQL Azure is a managed service, you do not have to install, patch or manage hardware or software. There is no need for you to create virtual machines or manually manage high availability. Every SQL Azure database has built-in high-availability, failover, and redundancy.
- SQL Azure federation makes scaling out multiple databases across 100s of nodes dramatically easier, while giving customers the flexibility to pay for only what they use.
- SQL Azure does not require new skillsets, programming models or newest tools, it works with the same SQL Server tools and APIs you use today.

Cons:

- Using Microsoft SQL Server Management Studio you can't use all features like visual design.
- Some SQL features are not available in the cloud like:
    - Full-Text search
    - Backup and Restore

Links:

- http://www.windowsazure.com/en-us/home/tour/sql-azure

4.10.6. Microsoft SQL Azure Reporting

Microsoft SQL Azure Reporting allows you to easily build reporting capabilities into your Windows Azure application. The reports can be accessed easily from the Windows Azure Portal, through a web browser, or directly from applications.
We created some Reports in order to analyze all cloud environment performance.

Pros:

- SQL Azure Reporting provides many of the features you know from SQL Server Reporting 2008 R2 to create reports with tables, charts, maps, gauges, and more and deploy them on both private and public clouds. You can literally take your report anywhere, and scale when and how you need it.
- The Service has built-in high availability and fault tolerance. SQL Azure Reporting is available across Microsoft Data centers around the world.
- You can effectively handle seasonal or fluctuating business requirements and reporting needs.
- SQL Azure Reporting scales as your requirements grow with easy self-provisioning.

Cons:

- Data are retrieved in the cloud, therefore in order to access on-premise database, they have to be connected to Azure

Link:

- http://www.windowsazure.com/en-us/home/tour/business-analytics/

4.10.7. Windows Azure AppFabric Caching

Windows Azure AppFabric Caching provides a distributed, in-memory cache, implemented as a cloud service. Caching is a managed service that is operated by Microsoft and has 99.9% monthly SLA.
We used Caching in order to save the Cheque Result in a fast way and due to its distributed capability, it makes possible to retrieve Cheque Result from it taking advantage of his fast read access capability.

Pros:

- Easily scale up or scale down by dynamically increasing or decreasing the cache size through configuration.
- Create a cache by choosing the cache size and region, and the service takes care of all the provisioning and management.
- Caching provides the flexibility to cache any type of managed object, including: XML, Binary Data, Rows and CLR Objects.
- Secured access and authorization to the cache is achieved by using security tokens from the Access Control service.
- The Caching service has a consistent development model with its Windows Server caching technology counterpart.

Cons:

- There is a variance in service performance to be kept monitored
- The cache is not cheap

Links:

- http://www.windowsazure.com/en-us/home/tour/caching

4.10.8. Enterprise Library Integration Pack for Windows Azure

The Microsoft Enterprise Library Integration Pack for Windows Azure is an extension to the Microsoft Enterprise Library 5.0 that can be used with Windows Azure. It includes the Autoscaling Application Block, the Transient Fault Handling Application Block, a protected configuration provider and the Blob configuration source.
It provides reusable components and developer's guidance designed to encapsulate recommended practices which facilitate consistency, ease of use, integration, extensibility, scalability and cost-effectiveness. Developers and IT professionals familiar with other Enterprise Library application blocks as well as those who are new to Enterprise Library will be able to benefit from the comprehensive set of content we are shipping today. You can pick and choose what matters to you and to your app, and adopt only those particular blocks/features.
In this POC we used in particular the Transient Fault Handling Application Block in order be able to manage easily the transient faults that can occur in the cloud.

Pros:

- Reusable components and developer's guidance
- Easy to be used code

Cons:

- The code becomes less readable because each instructions requiring access to the cloud, it's surrounded by the retry block.

Link:

- http://www.microsoft.com/download/en/details.aspx?id=28189
- http://blogs.msdn.com/b/agile/archive/2011/12/02/announcing-the-enterprise-library-integration-pack-for-windows-azure-with-autoscaling-transient-fault-handling-and-more.aspx

4.11. Amazon AWS Integration

Amazon Web Services provides a highly reliable, scalable, low-cost infrastructure platform in the cloud. It offers a set of services like:

- **Application Hosting:** Use reliable, on-demand infrastructure to power your applications, from hosted internal applications to Software-as-a-Service (SaaS) offerings.
- **Web Hosting:** performs dynamic web hosting with AWS's scalable infrastructure platform.
- **Databases:** variety of scalable database solutions, from hosted enterprise database software or non-relational database solutions.
- **Backup and Storage:** store data and build dependable backup solutions using AWS's inexpensive data storage services.
- **Content Delivery:** distribute content to end users worldwide, with low costs and high data transfer speeds.

4.11.1. Amazon Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. It has a simple web interface that allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment.

Comments: in the POC a base AMI image has been created with all needed software and SDKs preloaded like:

- Amazon SDKs
- MySQL .NET connector
- Windows Server AppFabric

Thi AMI has then be used as basic templates during deployment of Frontend and Backend application.

Pros:

- It enables you to increase or decrease capacity within minutes.
- Complete control of instances. You have root access to each one, and you can interact with them as you would any machine.

Cons:

- Operating System updates and configuration has still to be managed by Global Blue directly

Link:

- http://aws.amazon.com/ec2/

4.11.2. Amazon Simple Notification Service (SNS)

Amazon Simple Notification Service (Amazon SNS) is a web service that makes it easy to set up, operate, and send notifications from the cloud. It provides developers with a highly scalable, flexible, and cost-effective capability to publish messages from an application and immediately deliver them to subscribers or other applications. It is designed to make web-scale computing easier for developers.
It has a simple web services interface and browser-based Management Console that can be used to create topics you want to notify applications (or people) about, subscribe clients to these topics, publish messages, and have these messages delivered over clients' protocol of choice (i.e. HTTP, email, SMS, etc.).

Comments: we used this feature in order to send messages about operations done in FE service to the BE service taking advantage of publish subscribe pattern. SQS queues have been used as subscribers of SNS topics.

Pros:

- SNS runs within Amazon's proven network infrastructure and datacenters, so topics will be available whenever applications need them.
- Designed to meet the needs of the largest and most demanding applications, allowing applications to publish an unlimited number of messages at any time.

Cons:

- Used together with SQS it has a limit of 8KB per message

Link:

- http://aws.amazon.com/sns/

4.11.3. Amazon Simple Queue Service (Amazon SQS)

Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable, hosted queue for storing messages as they travel between computers. By using Amazon SQS, developers can simply move data between distributed components of their applications that perform different tasks, without losing messages or requiring each component to be always available.
Amazon SQS works by exposing Amazon's web-scale messaging infrastructure as a web service. Any computer on the Internet can add or read messages without any installed software or special firewall configurations.

Pros:

- SQS is high reliable cause it runs within Amazon's high-availability data centers.

- Designed to enable an unlimited number of computers to read and write an unlimited number of messages at any time.

Cons:

- Limited maximum size of message (64 KB)
- Variance in speed during the day and in different regions: APAC is faster than EU
- Messages can be retained in queues for a maximum of 14 days

Link:

- http://aws.amazon.com/sqs/

4.11.4. Amazon Simple Storage Service (S3)

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

S3 is intentionally built with a minimal feature set:

- Write, read, and delete objects containing from 1 byte to 5 terabytes of data each. The number of objects you can store is unlimited.
- Each object is stored in a bucket and retrieved via a unique, developer-assigned key.
- A bucket can be stored in one of several Regions. You can choose a Region to optimize for latency, minimize costs, or address regulatory requirements.
- Objects stored in a Region never leave the Region unless you transfer them out.
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.
- Uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

Pros:

- High availability
- Easy to use

Cons:

- Not so fast as expected

Link:

- http://aws.amazon.com/s3/

4.11.5. Amazon Relational Database Service (RDS)

Amazon Relational Database Service (RDS) is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you up to focus on your applications and business.

Pros:

- MySQL or Oracle database capabilities, so code, applications, and tools you already use

today with your existing databases can be used with Amazon RDS.
- Automatically patches the database software and backs up your database, storing the backups for a user-defined retention period and enabling point-in-time recovery.

Cons:
- For .NET development it offers a lower experience if compared to SQL Azure
- It supports Entity Framework, but lack of direct boolean support from MySQL leads to manual editing of EF 4.2 source files

Link:
- http://aws.amazon.com/rds/

4.11.6. Amazon ElastiCache

Amazon ElastiCache is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from a fast, managed, in-memory caching system, instead of relying entirely on slower disk-based databases.

Pros:
- Protocol-compliant with Memcached, so code, applications, and popular tools that you use today with existing Memcached environments will work seamlessly with the service.
- It has multiple features that enhance reliability for critical production deployments.

Cons:
- .NET client libraries available are not so mature. The most used is Enyim but it accesses AWS Elasticache through IP address instead of FQDN

Link:
- http://aws.amazon.com/elasticache/

4.11.7. Amazon SimpleDB

Amazon SimpleDB is a highly available and flexible non-relational data store that offloads the work of database administration.

Pros:
- Optimized to provide high availability and flexibility
- SimpleDB creates and manages multiple geographically distributed replicas of your data automatically to enable high availability and data durability

Cons:
- It supports only String type
- Requires more effort during development to store and retrieve numeric types (i.e. left padding for sorting)
- Very limited search capabilities

Link:
- http://aws.amazon.com/simpledb/

4.11.8. Amazon CloudFormation

AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion.
You can use AWS CloudFormation's sample templates or create your own templates to describe the AWS resources, and any associated dependencies or runtime parameters, required to run your application.

Pros:

- AWS CloudFormation supports many AWS resources, allowing you to build a highly available, reliable, and scalable AWS infrastructure for your application needs.
- Easy to organize a collection of AWS resources you want to deploy and lets you describe any dependencies or special parameters that can be passed in at runtime.
- A template can be used repeatedly to create identical copies of the same stack
- Templates are simple JSON formatted text files that can be placed under your normal source control mechanisms

Cons:

- N/A

Link:

- http://aws.amazon.com/cloudformation/

5. CUSTOM APPLICATION SETTINGS

5.1. Azure configuration files

This paragraph describes all custom sections defined in the standard Web.config file for the Azure implementation. Below follows a brief description of the structure of ServiceDefinition.cscfg.

Structure of ServiceDefinition.cscfg

The service configuration file specifies the number of role instances to deploy for each role in the service, the values of any configuration settings, and the thumbprints for any certificates associated with a role.

```
<ServiceConfiguration serviceName="<service-name>" osFamily="[1|2]" osVersion="<os-version>">

  <Role name="<role-name>">
```

```
    <Instances count="<number-of-instances>" />

    <ConfigurationSettings>

      <Setting name="<setting-name>" value="<setting-value>" />

    </ConfigurationSettings>

    <Certificates>

      <Certificate name="<certificate-name>" thumbprint="<certificate-thumbrint>"
thumbprintAlgorithm="<algorithm>" />

    </Certificates>

    <OsImage href="<vhd_image_name>" />

  </Role>

</ServiceConfiguration>
```

Role element:

The Role element describes the number of instances, configuration settings, and certificate thumbprints for a role defined by the service.

- *name*: The name of the role. The name must match the name provided for the role in the service definition file. This field is mandatory.

Instances element:

The Instances element specifies the number of instances of this role to deploy as part of the service deployment. Instances is a required optional element of the configuration file.

- *count:* The required number of role instances. This field is mandatory.

Configurationsettings element:

The ConfigurationSettings element describes the collection of configuration settings for the role. ConfigurationSettings is an optional element of the configuration file.

Setting element:

The Setting element describes a name-value pair that provides a value for a configuration setting for an instance of a role.

- *name*: A unique name for the configuration setting. The name must match the name declared for the configuration setting in the service definition file. This field is mandatory.
- *value*: The value for the configuration setting. This field is mandatory.

Certificates element:

The Certificates element describes the collection of certificates for the role. Certificates is an optional element of the configuration file.

Certificate element:

The Certificate element describes the thumbprint and algorithm associated with a certificate defined for this role. The certificate that is used for the remote desktop connection.

- *name*: the name of the certificate, as given in the service definition file. This field is mandatory.
- *thumbprint*: The certificate thumbprint. This value cannot be an empty string. The certificate thumbprint is a string of hexadecimal numbers containing no spaces. The hexadecimal numbers must be represented using digits and uppercase alpha characters. This field is mandatory.
- *thumbprintAlgorithm*: The algorithm used to generate the certificate thumbprint. This field is mandatory.

5.1.1. MessageBus Receiver

Topic section

```
<topicsSectionAzure>

<servicebuses>

<add key="EU1servicebus" namespaceAddress="sbgbeu1" issuerName="owner"
issuerKey="fwb+mIFPONWJyZQVV30qpkfYfgyUr+qoqlpIO8GfZMc="/>

<add key="EU2servicebus" namespaceAddress="sbgbeu2" issuerName="owner"
issuerKey="2tTNeX26Hjd6yFT3xA656GQ/NbglY3DwUo2/6b396bE="/>

<add key="APAC1servicebus" namespaceAddress="sbgbapac1" issuerName="owner"
issuerKey="a8nD9NhmAlI3g7JNqpOVA/mzMSg+i0heJ9sDEt1c6T8="/>

</servicebuses>

<topics>

<add key="APAC1" value="tfscheque" servicebusKey="APAC1servicebus"/>
```

```
<add key="EU2" value="tfscheque" servicebusKey="EU2servicebus"/>
<add key="CEU1" value="controlbus" servicebusKey="EU1servicebus" isControlBus="true"/>
<add key="EU1" value="tfscheque" servicebusKey="EU1servicebus"/>
</topics>
<subscriptions>
<add key="EU1" value="DEVIssuedAndVoided" topicKey="EU1"/>
<add key="EU2" value="DEVIssuedAndVoided" topicKey="EU2"/>
<add key="APAC1" value="DEVIssuedAndVoided" topicKey="APAC1"/>
<add key="CEU1" value="devcontrolbus" topicKey="CEU1"/>
</subscriptions>
<filters>
<add key="EU11" subscriptionKey="EU1" field="Operation" value="Issue"/>
<add key="EU12" subscriptionKey="EU1" field="Operation" value="IssueWithObject" operator="or"/>
<add key="EU13" subscriptionKey="EU1" field="Operation" value="Void" operator="or"/>
<add key="EU14" subscriptionKey="EU1" field="Operation" value="VoidWithObject" operator="or"/>
<add key="EU21" subscriptionKey="EU2" field="Operation" value="Issue"/>
<add key="EU22" subscriptionKey="EU2" field="Operation" value="IssueWithObject" operator="or"/>
<add key="EU23" subscriptionKey="EU2" field="Operation" value="Void" operator="or"/>
<add key="EU24" subscriptionKey="EU2" field="Operation" value="VoidWithObject" operator="or"/>
<add key="APAC11" subscriptionKey="APAC1" field="Operation" value="Issue"/>
<add key="APAC12" subscriptionKey="APAC1" field="Operation" value="IssueWithObject" operator="or"/>
<add key="APAC13" subscriptionKey="APAC1" field="Operation" value="Void" operator="or"/>
<add key="APAC14" subscriptionKey="APAC1" field="Operation" value="VoidWithObject" operator="or"/>
</filters>
</topicsSectionAzure>
```

The “topicsSectionAzure” section is used to configure the ServiceBus custom settings. The <servicebuses> group of settings defines which servicebuses are managed by the current instance. For each servicebus (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related servicebus.
- *nameSpaceAddress*: the name of the Service Namespace created. This field is mandatory.
- *issuerName*: the Name of the created Storage Account. This field is mandatory.

- *issuerKey*: the Primary Access Key recorded during Storage Account creation. This field is mandatory.

The <topics> group of settings defines which topics are defined for each service bus. For each topic (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related service bus.
- *value*: the name of the topic. This field is mandatory.
- *isControlBus*: this field indicates if the topic is that one used to monitor when the region applications (frontends and backend) should be stopped. This field is not mandatory and the default value is false.
- *serviceBusKey*: the key of the father service bus. This field is mandatory.

The <subscriptions> group of settings defines which subscriptions are defined for each topic. For each subscription (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related topic. This field is mandatory.
- *value*: the name of the subscription. This field is mandatory.
- *topicKey*: the key of the father topic. This field is mandatory.

The <filters> group of settings defines which filters are defined for each subscription. For each filter (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related subscription. This field is mandatory.
- *field*: the message property used to define the filter in subscription. This field is mandatory.
- *value*: the value of the message property defined with equals operator. This field is mandatory.
- *operator*: the logical operator used to concatenate multiple filters. Possible values are “and” and “or”. This field is not mandatory in case of only one filter is defined.
- *subscriptionKey*: the key of the father topic. This field is mandatory.

In case of subscription with “Issue”, “Void” and “IssueWithObject” as possible values for “Operation” message property, following <add> elements should be added to filter element:

```
<add key="key1" subscriptionKey="subscriptionKey" field="Operation" value="Issue"/>

<add key="key2" subscriptionKey="subscriptionKey" field="Operation" value="IssueWithObject" operator="or"/>

<add key="key3" subscriptionKey="subscriptionKey" field="Operation" value="Void" operator="or"/>
```

Note: In case of a mandatory field is not set in the config file, a *System.Configuration.ConfigurationErrorsException* is thrown with the message: *Required attribute 'field' not found.*

AppSettings

```
<appSettings>

<add key="Receiver" value="AZUREEU1"/>

<add key="LocalDataConnectionString" value="DataConnectionStringEU1"/>

<add key="microsoft.visualstudio.teamsystems.aspnetdevserver:/" value="1337;True;6308;1;-8588791831116947398"/>

<add key="microsoft.visualstudio.teamsystems.backupinfo" value="1;web.config.backup" />

<add key="CacheService.TableStorage.Provider" value="AppFabric" />

<add key="CacheService.TableStorage.EnableCache" value="False" />

<add key="CacheService.TableStorage.ExpirationType" value="D" />

<add key="CacheService.TableStorage.ExpirationValue" value="30" />

<add key="CacheService.BlobStorage.Provider" value="AppFabric" />

<add key="CacheService.BlobStorage.EnableCache" value="False" />

<add key="CacheService.BlobStorage.ExpirationType" value="D" />

<add key="CacheService.BlobStorage.ExpirationValue" value="30" />

<add key="CacheService.Provider" value="AppFabric" />

<add key="CacheService.EnableCache" value="TRUE" />

<add key="CacheService.ExpirationType" value="D" />

<add key="CacheService.ExpirationValue" value="30" />

</appSettings>
```

The standard <appSettings> section defines all <add> items for instance that need to be configured depending upon the environment:

- Receiver: The identity of current system. This field is compared with related “Sender” field in GripsCloudCoordinator in order to decide if data are coming or not from the same environment (EU1, EU2, APAC1 …).
- *LocalDataConnectionString*: this field represents the name of *DataConnectionString* defined in Serviceconfiguration.cscfg for the current (local) instance.
- *microsoft.visualstudio.teamsystems.aspnetdevserver:/*: XXXXXXXXXXXX
- *microsoft.visualstudio.teamsystems.backupinfo*:XXXXXXXXXXXXXXX

- *CacheService.TableStorage.EnableCache*: this field indicates if the cache layer above the Table Storage is enabled. If the value is set to true, all parameters with prefix *CacheService.TableStorage* are mandatory. If the value is set to false, the layer will simply manage the Table Storage, without using the cache.
- *CacheService.TableStorage.Provider*: this field set which cache provider will be used for managing the cache layer. Two different values for the setting are recognized, one for the Windows Azure AppFabric Caching (specify APPFABRIC as value) and another one for the default in-memory cache provider offered by the .NET Framework 4 (specify INMEMORY as value).
- *CacheService.TableStorage.ExpirationType*: this field indicates the time measurement unit for the parameter *CacheService.TableStorage.ExpirationValue*. Possible values are D (days), H (hours) and M (minutes).
- *CacheService.TableStorage.ExpirationValue*: this field indicates how many time should elapse before the cache expire.
- *CacheService.BlobStorage.EnableCache*: this field indicates if the cache layer above the Blob Storage is enabled. If the value is set to true, all parameters with prefix *CacheService.BlobStorage* are mandatory. If the value is set to false, the layer will simply manage the Blob Storage, without using the cache.
- *CacheService.BlobStorage.Provider*: this field sets which cache provider will be used for manage the cache layer. Two different values for the setting are recognized, one for the Windows Azure AppFabric Caching (specify APPFABRIC as value) and another one for the default in-memory cache provider offered by the .NET Framework 4 (specify INMEMORY as value).
- *CacheService.BlobStorage.ExpirationType*: this field indicates the time measurement unit for the parameter *CacheService.BlobStorage.ExpirationValue*. Possible values are D (days), H (hours) and M (minutes).
- *CacheService.BlobStorage.ExpirationValue*: this field indicates how much time should elapse before the cache expires.
- *CacheService.EnableCache*: this field indicates if the cache layer is enabled. If the value is set to true, all parameters with prefix *CacheService* are mandatory. If the value is set to false, the cache layer will be disabled.
- *CacheService.Provider*: this field sets which cache provider will be used for manage the cache layer. Two different values for the setting are recognized, one for the Windows Azure AppFabric Caching (specify APPFABRIC as value) and another one for the default in-memory cache provider offered by the .NET Framework 4 (specify INMEMORY as value).
- *CacheService.ExpirationType*: this field indicates the time measurement unit for the parameter *CacheService.ExpirationValue*. Possible values are D (days), H (hours) and M (minutes).
- *CacheService.ExpirationValue*: this field indicates how much time should elapse before the cache expires.

Settings in ServiceConfiguration.cscfg

```
<ConfigurationSettings>
<Setting name="DataConnectionStringEU1"
value="DefaultEndpointsProtocol=https;AccountName=gripscoordinatoreu1;AccountKey=u46aNrubQw24rO5AeqIeQCZc6HRpaiZk48WKo4b5
aECW4J3aMaVGcYLD6EHjlfApFB7dpTBftcaoZD/k3n312g==" />
<Setting name="DataConnectionStringEU2"
value="DefaultEndpointsProtocol=https;AccountName=gripscoordinatoreu2;AccountKey=RWPWELYls66dJ9Ebh2AF+J7ux+pN19VGDzaxoTRq
2bxBjRWhmLnmwLfXIEJifL9TK3VyWASE7rwRcR4W1f6a9w==" />

<Setting name="DataConnectionStringAPAC1"
value="DefaultEndpointsProtocol=https;AccountName=gripscoordinatorapac1;AccountKey=3aqcmg3dfuwj2LJPxMjLqriUR1FVGxkUcZDTct
HxusbN7cTL8+Uojso0cBIGY84C8pHWGG86We1R4PIzD97q0A==" />

<Setting name="DataConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=gripscoordinatoreu1;AccountKey=u46aNrubQw24rO5AeqIeQCZc6HRpaiZk48WKo4b5
aECW4J3aMaVGcYLD6EHjlfApFB7dpTBftcaoZD/k3n312g==" />

<Setting name="DiagnosticsConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=gripscoordinatoreu1;AccountKey=u46aNrubQw24rO5AeqIeQCZc6HRpaiZk48WKo4b5
aECW4J3aMaVGcYLD6EHjlfApFB7dpTBftcaoZD/k3n312g==" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.Enabled" value="true" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountUsername" value="4ward" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountEncryptedPassword"
value="MIIBnQYJKoZIhvcNAQcDoIIBjjCCAYoCAQAxggFOMIIBSgIBADAyMB4xHDAaBgNVBAMME1dpbmRvd3MgQXp1cmUgVG9vbHMCEGcbqTuyypWsQdmZXr
lISJMwDQYJKoZIhvcNAQEBBQAEggEANHdc4wPtEREVodRCiEPchPWVdYFnxEGQsSKwJOo+1hXOZLMrSC1YrJBjyz9Y7I0HNHIgcRyO/lIrHYKIGdaYGrwnVi/
Q9XCi6c4Axjc8wixS4zXFfH0A6PeJVwM574BtrCE3sKjLGU4G/Ze7o4hMehXqVy2OTXXrSt0c0kGv3/cJ/UA6QE+xGej91Po68ytJguIOwVNkYhrHzQ19yb26
kA/hd/t/Aqv4FFi1yfFk5mH6IQKP7usdUeIfvfbTESO7rmSm1npLb9A41ik+90y0RirPanFM+vKa7WBWZ3qNrOYpLd+RBSDbkaxCMUEZomDe5PWWILynO2tp9
6rbS3STtTAzBgkqhkiG9w0BBwEwFAYIKoZIhvcNAwcECHyYDkm4SqvNgBC74UlPv/44ykRfNk7XP/p0" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountExpiration" value="2012-12-21T23:59:59.0000000+01:00"
/>

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteForwarder.Enabled" value="true" />

<Setting name="poisonMessageThreshold" value="3" />

<Setting name="receiverThreadNumber" value="5" />

<Setting name="poisonReceiverThreadNumber" value="3" />

<Setting name="poisonPrefetchCount" value="20" />

<Setting name="subscriptionPrefetchCount" value="200" />

</ConfigurationSettings>
```

- *DataConnectionStringEU1*: This is the connection string to the Window Azure storage account for EU1 region. It is used to programmatically access data storage and other functionalities in Windows Azure. *AccountName* is the name of the Windows Azure storage account and *AccountKey* is the access key for the storage account.
- *DataConnectionStringEU2*: This is the connection string to the Window Azure storage account for EU2 region. It is used to programmatically access data storage and other functionalities in Windows Azure. *AccountName* is the name of the Windows Azure storage account and *AccountKey* is the access key for the storage account.
- *DataConnectionStringAPAC1*: This is the connection string to the Window Azure storage

account for APAC1 region. It is used to programmatically access data storage and other functionalities in Windows Azure. *AccountName* is the name of the Windows Azure storage account and *AccountKey* is the access key for the storage account.

- *DataConnectionString*: This is the connection string to the Window Azure storage account for the current region. It is used to programmatically access data storage and other functionalities in Windows Azure. *AccountName* is the name of the Windows Azure storage account and *AccountKey* is the access key for the storage account.
- *DiagnosticsConnectionString*: this field contains Windows Azure storage account information to transfer diagnostic data. *AccountName* is the name of the Windows Azure storage account and *AccountKey* is the access key for the storage account.
- *Microsoft.WindowsAzure.Plugins.RemoteAccess.Enabled*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. Set true to enable a remote desktop connection for a role; otherwise, false.
- *Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountUsername*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. This field represents the name of the user account that can access the instance. You must use this account when remotely accessing the role instance.
- *Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountEncryptedPassword*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. This field represents the encrypted password for the user account.
- *Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountExpiration*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. This field represents the expiration date of the remote desktop connection in ISO 8601 “yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fffffffK” format. For example, the expiration date could be "2011-12-17T23:59:59.0000000-08:00".
- *Microsoft.WindowsAzure.Plugins.RemoteForwarder.Enabled*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. Set true to enable forwarding of remote desktop connection for the service; otherwise, false.
- *poisonMessageThreshold*: this field represents the threshold for put a message in the deadletter queue. A message that has been received (but not deleted) a number of times greater than the PoisonMessageThreshold is put in the deadletter queue.
- *receiverThreadNumber*: this field indicates how many threads are instanced for processing poison messages.
- *poisonReceiverThreadNumber*: this field indicates how many threads are instanced for processing poison messages.
- *PoisonPrefetchCount:* this field represents the number of messages that the message receiver can simultaneously receive from the server (deadletter queue) in the same round-trip.
- *SubscriptionPrefetchCount:* this field represents the number of messages that the

message receiver can simultaneously receive from the server in the same round-trip.

5.1.1. GripsCloudCoordinator

AppSettings

```
<appSettings>
<add key="AccountName" value="gripscoordinatoreu1" />
<add key="ChequeTopicName" value="tfscheque" />
<add key="ControlBusTopicName" value="controlbus"/>
<add key="Sender" value="AZUREEU1" />
<add key="microsoft.visualstudio.teamsystems.aspnetdevserver:/"
  value="1337;True;6308;1;-8588791831116947398" />
<add key="microsoft.visualstudio.teamsystems.backupinfo" value="1;web.config.backup" />
<add key="CacheService.TableStorage.Provider" value="InMemory" />
<add key="CacheService.TableStorage.EnableCache" value="False" />
<add key="CacheService.TableStorage.ExpirationType" value="D" />
<add key="CacheService.TableStorage.ExpirationValue" value="30" />
<add key="CacheService.BlobStorage.Provider" value="InMemory" />
<add key="CacheService.BlobStorage.EnableCache" value="False" />
<add key="CacheService.BlobStorage.ExpirationType" value="D" />
<add key="CacheService.BlobStorage.ExpirationValue" value="30" />
<add key="CacheService.Provider" value="AppFabric" />
<add key="CacheService.EnableCache" value="TRUE" />
<add key="CacheService.ExpirationType" value="D" />
<add key="CacheService.ExpirationValue" value="30" />
</appSettings>
```

The standard <appSettings> section defines all <add> items for instance that need to be configured depending upon the environment:

- *AccountName*: this field indicates is the Azure account name.
- *ChequeTopicName:* this field is the name of the specific topic for cheques configured on Windows Azure AppFabric Service Bus.
- *ControlBusTopicName:* this field is the name of the specific topic for controlbus

configured on Windows Azure AppFabric Service Bus.

- *Sender*: The identity of current system. This field is compared with related "Receiver" field in GripsCloudCoordinator in order to decide if data are coming or not from the same environment (EU1, EU2, APAC1 …).
- *microsoft.visualstudio.teamsystems.aspnetdevserver:/*: XXXXXXXXXXXX
- *microsoft.visualstudio.teamsystems.backupinfo*:XXXXXXXXXXXXXXX
- *CacheService.TableStorage.EnableCache*: this field indicates if the cache layer above the Table Storage is enabled. If the value is set to true, all parameters with prefix *CacheService.TableStorage* are mandatory. If the value is set to false, the layer will simply manage the Table Storage, without using the cache.
- *CacheService.TableStorage.Provider*: this field set which cache provider will be used for managing the cache layer. Two different values for the setting are recognized, one for the Windows Azure AppFabric Caching (specify APPFABRIC as value) and another one for the default in-memory cache provider offered by the .NET Framework 4 (specify INMEMORY as value).
- *CacheService.TableStorage.ExpirationType*: this field indicates the time measurement unit for the parameter *CacheService.TableStorage.ExpirationValue*. Possible values are D (days), H (hours) and M (minutes).
- *CacheService.TableStorage.ExpirationValue*: this field indicates how many time should elapse before the cache expire.
- *CacheService.BlobStorage.EnableCache*: this field indicates if the cache layer above the Blob Storage is enabled. If the value is set to true, all parameters with prefix *CacheService.BlobStorage* are mandatory. If the value is set to false, the layer will simply manage the Blob Storage, without using the cache.
- *CacheService.BlobStorage.Provider*: this field sets which cache provider will be used for manage the cache layer. Two different values for the setting are recognized, one for the Windows Azure AppFabric Caching (specify APPFABRIC as value) and another one for the default in-memory cache provider offered by the .NET Framework 4 (specify INMEMORY as value).
- *CacheService.BlobStorage.ExpirationType*: this field indicates the time measurement unit for the parameter *CacheService.BlobStorage.ExpirationValue*. Possible values are D (days), H (hours) and M (minutes).
- *CacheService.BlobStorage.ExpirationValue*: this field indicates how much time should elapse before the cache expires.
- *CacheService.EnableCache*: this field indicates if the cache layer is enabled. If the value is set to true, all parameters with prefix *CacheService* are mandatory. If the value is set to false, the cache layer will be disabled.
- *CacheService.Provider*: this field sets which cache provider will be used for manage the cache layer. Two different values for the setting are recognized, one for the Windows Azure AppFabric Caching (specify APPFABRIC as value) and another one for the default in-memory cache provider offered by the .NET Framework 4 (specify INMEMORY as value).
- *CacheService.ExpirationType*: this field indicates the time measurement unit for the

parameter *CacheService.ExpirationValue*. Possible values are D (days), H (hours) and M (minutes).

- *CacheService.ExpirationValue*: this field indicates how much time should elapse before the cache expires.

Settings in ServiceConfiguration.cscfg

```
<ConfigurationSettings>

<Setting name="DataConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=gripscoordinatoreu1;AccountKey=u46aNrubQw24rO5AeqIeQCZc6HRpaiZk48WKo4b5
aECW4J3aMaVGcYLD6EHjlfApFB7dpTBftcaoZD/k3n312g==" />

<Setting name="DiagnosticsConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=gripscoordinatoreu1;AccountKey=u46aNrubQw24rO5AeqIeQCZc6HRpaiZk48WKo4b5
aECW4J3aMaVGcYLD6EHjlfApFB7dpTBftcaoZD/k3n312g==" />

<Setting name="issuerKey" value="fwb+mIFPONWJyZQVV30qpkfYfgyUr+qoqlpIO8GfZMc=" />

<Setting name="issuerName" value="owner" />

<Setting name="namespaceAddress" value="sbgbeu1" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.Enabled" value="true" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountUsername" value="4ward" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountEncryptedPassword"
value="MIIBnQYJKoZIhvcNAQcDoIIBjjCCAYoCAQAxggFOMIIBSgIBADAyMB4xHDAaBgNVBAMME1dpbmRvd3MgQXp1cmUgVG9vbHMCEDBtByk5smi2TdoyzY
TKM0IwDQYJKoZIhvcNAQEBBQAEggEAnreZRnIfnO78CSzumMYvGn2mnV/gfKSsVff9jV8855cjEMnQEKqiRK5gA4klawPMUwRqu+DgFbMmUXhk6rjNRAwb8Qz
dWChMEv37cheWj9TD8cpddaqkRmI9AgZzFjnHJB60fmpEbQpUMkb3Z7cP0swZdzw9/szwnFHJh7bl3Au9r7NCimmlgp+WJ/k3qp+0VZRtO+Z4/1ZhcfagjV9i
45mcvVDETw6YHwjvx0sV6CtsJnsQE1cR02hcvMm/AkBjeyxaXokdqYrW4Sge/iwsEyhQ9qPU2O51+ad8jNOwQ8Hqzsy6LNNnglVhcl8VxOzZW4Y8lKRThW3jD
wamdUm1TjAzBgkqhkiG9w0BBwEwFAYIKoZIhvcNAwcECKd2AsIkvXvegBBWRg9v+V27NbCPH/eyGYYE" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountExpiration" value="2012-12-2T23:59:59.0000000+01:00" />

<Setting name="Microsoft.WindowsAzure.Plugins.RemoteForwarder.Enabled" value="true" />

<Setting name="ServiceBus" value="AZURE" />

</ConfigurationSettings>
```

- *DataConnectionString*: This is the connection string to the Window Azure storage account for the current region. It is used to programmatically access data storage and other functionalities in Windows Azure. *AccountName* is the name of the Windows Azure storage account and *AccountKey* is the access key for the storage account.
- *DiagnosticsConnectionString*: this field contains Windows Azure storage account information to transfer diagnostic data. *AccountName* is the name of the Windows Azure storage account and *AccountKey* is the access key for the storage account.
- *issuerKey*: the issuer key of the application namespace created in Windows Azure AppFabric Service Bus, needed in authentication with Access Control.
- *issuerName*: the issuer name of the application namespace created in Windows Azure AppFabric Service Bus, needed in authentication with Access Control.

- *namespaceAddress:* the namespace address of the namespace application created in Windows Azure AppFabric Service Bus.
- *Microsoft.WindowsAzure.Plugins.RemoteAccess.Enabled*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. Set true to enable a remote desktop connection for a role; otherwise, false.
- *Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountUsername*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. This field represents the name of the user account that can access the instance. You must use this account when remotely accessing the role instance.
- *Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountEncryptedPassword*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. This field represents the encrypted password for the user account.
- *Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountExpiration*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. This field represents the expiration date of the remote desktop connection in ISO 8601 “yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fffffffK” format. For example, the expiration date could be "2011-12-17T23:59:59.0000000-08:00".
- *Microsoft.WindowsAzure.Plugins.RemoteForwarder.Enabled*: When the RemoteAccess module is imported into the service definition, this configuration setting is automatically added for the role. Set true to enable forwarding of remote desktop connection for the service; otherwise, false.
- *ServiceBus:* This field indicates which pattern will be used in storing cheques. Possible values are: “AZURE” and “AZUREFULL”. In the “AZURE” mode, GripsCloudCoordinator saves cheque in local BlobStorage and sends to ServiceBus all information needed to retrieve data from Blob Storage and store data where requested (remote Blob Staorage and SQL Azure). In the “AZUREFULL” mode, GripsCloudCoordinator sends to ServiceBus the full cheque and the MessageBusReceiver will store it directly where requested.
-

5.2. AWS Web and App config

This paragraph describes all custom sections defined in the Web.config file for the AWS implementation.

5.2.1. MessageBus Receiver

Topic section

```
<topicsSectionAWS>

<regions>

<add key="APAC" value="Asia Pacific (Singapore)" isLocal="false" sqsUrl="https://ap-southeast-1.queue.amazonaws.com"
```

```
snsUrl="https://sns.ap-southeast-1.amazonaws.com" s3Url="s3-ap-southeast-1.amazonaws.com" ec2Url="http://ec2.ap-
southeast-1.amazonaws.com"/>

<add key="EU" value="EU West" isLocal="true" sqsUrl="https://eu-west-1.queue.amazonaws.com" snsUrl="https://sns.eu-west-
1.amazonaws.com" s3Url="s3-eu-west-1.amazonaws.com" ec2Url="http://ec2.eu-west-1.amazonaws.com"/>

</regions>

<buckets>

<add key="chequeapac" value="tfschequeapac" regionKey="APAC" />

</buckets>

<topics>

<add key="APAC" value="tfschequeapac" regionKey="APAC" />

<add key="EU" value="tfschequeeu" regionKey="EU"/>

<add key="CEU" value="controlbuseu" regionKey="EU" isControlBus ="true"/>

</topics>

<subscriptions>

<add key="APAC11" value="IssuedAndVoidedAPAC" topicKey="APAC" />

<add key="APAC12" value="IssuedAndVoidedAPAC" topicKey="EU"/>

<add key="CEU11" value="devcontrolbuseu" topicKey="CEU"/>

</subscriptions>

</topicsSectionAWS>
```

The “topicsSectionAWS” section is used to configure the ServiceBus custom settings.

The <regions> group of settings defines which regions are managed by the current instance. For each region (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related service bus. This field is mandatory.
- *value*: this field is not used at all by the application and it should be a meaning description of the region. This field is not mandatory.
- *isLocal*: this field indicates if the region is the one of the current installation.
- *sqsUrl*: the URL of Amazon Simple Notification Service (Amazon SNS) of the current region.
- *snsUrl*: the URL of Amazon Simple Queue service (Amazon SQS) for the current region.
- *s3Url*: the URL of Amazon Simple Storage Service (Amazon S3) for the current region.
- *ec2Url*: the URL of Amazon Elastic Compute Cloud (Amazon EC2) for the current region.

The <buckets> group of settings defines which buckets are defined in Amazon S3 for each

region. For each bucket (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related region.
- *value*: the name of the bucket. This field is mandatory.
- *regionKey*: the key of the father region. This field is mandatory.

The <topics> group of settings defines which topics are defined in Amazon SNS for each region. For each topic (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related region.
- *value*: the name of the topic. This field is mandatory.
- *regionKey*: the key of the father region. This field is mandatory.

The <subscriptions> group of settings defines which subscriptions are defined for each topic in Amazon SNS. For each subscription (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related topic. This field is mandatory.
- *value*: the name of the subscription. This field is mandatory.
- *topicKey*: the key of the father topic. This field is mandatory.

Note: In case of a mandatory field is not set in the config file, a *System.Configuration.ConfigurationErrorsException* is thrown with the message: *Required attribute 'field' not found.*

AppSettings

```
<appSettings>

<add key="Receiver" value="AWSEU1" />

<add key="ReceiverThreadNumber" value="10" />

<add key="ReceiverMessageNumber" value="10" />

<add key="MessageVisibilityTimeout" value="120" />

<add key="MessageReceivingTimeOutSeconds" value="5" />

<add key="MessagePollIntervalSeconds" value="2" />

<add key="PoisonReceiverThreadNumber" value="3" />

<add key="PoisonMessageThreshold" value="3" />

<add key="ClientSettingsProvider.ServiceUri" value="" />
```

```
<add key="AWSAccessKey" value="AKIAJCPYEWYOIX4ZVR2Q"/>

<add key=" AWSSecretKey" value="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"/>

<add key="CacheService.EnableCache" value="TRUE" />

</appSettings>
```

The standard <appSettings> section defines all <add> items for instance that need to be configured depending upon the environment:

- Receiver: The identity of current system. This field is compared with related "Sender" field in GripsCloudCoordinator in order to decide if data are coming or not from the same environment (EU1, EU2, APAC1 …).
- *ReceiverThreadNumber*: this field indicates how many threads are instanced for processing dequeued messages.
- *ReceiverMessageNumber*: this field indicates the maximum number of messages simultaneously returned by SQS. The value must be included between 1 and 10. Amazon SQS never returns more messages than this value but may return fewer.
- *MessageVisibilityTimeout*: The duration (in seconds) that the received messages from SQS queue are hidden from subsequent retrieve requests.
- *MessageReceivingTimeOutSeconds*: this field represents the timeout in seconds during polling SQS queue.
- *MessagePollIntervalSeconds*: the polling interval between each receives from SQS queue.
- *PoisonReceiverThreadNumber*: this field indicates how many threads are instanced for processing poison messages.
- *PoisonMessageThreshold*: this field represents the threshold for put a message in the poison queue. A message that has been received (but not deleted) a number of times greater than the *PoisonMessageThreshold* is put in the poison queue.
- *ClientSettingsProvider.ServiceUri*: XXXXXXXXXXXXXXXXXX
- *AWSAccessKey*:XXXXXXXXXXXXXXXXXXXXXXXXXXXX
- *AWSSecretKey*:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
- *CacheService.EnableCache*: this field indicates if the Memcached client that manages the Amazon ElastiCache is enabled.

5.2.2. GripsCloudCoordinator

Topic section

```
<topicsSectionAWS>

<regions>

<add key="EU" value="EU West" isLocal="true" sqsUrl="https://eu-west-1.queue.amazonaws.com" snsUrl="https://sns.eu-west-
```

```
1.amazonaws.com" s3Url="s3-eu-west-1.amazonaws.com" ec2Url="http://ec2.eu-west-1.amazonaws.com"/>

</regions>

<buckets>

<add key="cheque" value="tfschequeeu" regionKey="EU" />

</buckets>

<topics>

<add key="EU" value="tfschequeeu" arn="arn:aws:sns:eu-west-1:648522325131:tfschequeeu" regionKey="EU" />

<add key="CEU" value="controlbuseu" arn="arn:aws:sns:eu-west-1:648522325131:controlbuseu" regionKey="EU"
isControlBus="true"/>

</topics>

</topicsSectionAWS>
```

The "topicsSectionAWS" section is used to configure the ServiceBus custom settings. This section is the same described in paragraph "MessageBus Receiver – AWS Topic Section". The only difference is that in GripsCloudoordinator only the local region (with "isLocal" set to true) need to be configured.

AppSettings

```
<appSettings>

<add key="microsoft.visualstudio.teamsystems.aspnetdevserver:/" value="1337;True;6308;1;-8588791831116947398" />

<add key="microsoft.visualstudio.teamsystems.backupinfo" value="1;web.config.backup" />

<add key="CacheService.EnableCache" value="TRUE" />

<add key ="Sender" value="AWSEU1"/>

<add key="AWSAccessKey" value="AKIAJCPYEWYOIX4ZVR2Q"/>

<add key="AWSSecretKey" value="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"/>

<add key="DeployStack" value="GCAWSLB"/>

</appSettings>
```

The standard <appSettings> section defines all <add> items for instance that need to be configured depending upon the environment:

- *microsoft.visualstudio.teamsystems.aspnetdevserver:/*: XXXXXXXXXXXX
- *microsoft.visualstudio.teamsystems.backupinfo*:XXXXXXXXXXXXXXX
- *CacheService.EnableCache*: this field indicates if the Memcached client that manages the Amazon ElastiCache is enabled.
- *Sender*: The identity of current system. This field is compared with related "Receiver" field in GripsCloudCoordinator in order to decide if data are coming or not from the

same environment (EU1, EU2, APAC1 …).

- *AWSAccessKey*:XXXXXXXXXXXXXXXXXXXXXXXXXXX
- *AWSSecretKey*:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
- *DeployStack*:XXXXXXXXXXXXXXXXXXXXXXXX (è il prefisso usato per cercare le public DNS instances su EC2 per fare lo start/stop dei frontend).

5.3. vCloud Web and App config

This paragraph describes all custom sections defined in the Web.config file for the vCloud implementation.

5.3.1. MessageBus Receiver

AWS Topic section

```
<topicsSectionAWS>

<regions>

<add key="APAC" value="Asia Pacific (Singapore)" isLocal="false" sqsUrl="https://ap-southeast-1.queue.amazonaws.com"
snsUrl="https://sns.ap-southeast-1.amazonaws.com" s3Url="s3-ap-southeast-1.amazonaws.com" ec2Url="http://ec2.ap-
southeast-1.amazonaws.com"/>

<add key="EU" value="EU West" isLocal="true" sqsUrl="https://eu-west-1.queue.amazonaws.com" snsUrl="https://sns.eu-west-
1.amazonaws.com" s3Url="s3-eu-west-1.amazonaws.com" ec2Url="http://ec2.eu-west-1.amazonaws.com"/>

</regions>

<buckets>

<add key="chequeapac" value="tfschequeapac" regionKey="APAC" />

</buckets>

<topics>

<add key="APAC" value="tfschequeapac" regionKey="APAC" />

<add key="EU" value="tfschequeeu" regionKey="EU"/>

<add key="CEU" value="controlbuseu" regionKey="EU" isControlBus ="true"/>

</topics>

<subscriptions>

<add key="APAC11" value="IssuedAndVoidedAPAC" topicKey="APAC" />

<add key="APAC12" value="IssuedAndVoidedAPAC" topicKey="EU"/>

<add key="CEU11" value="devcontrolbuseu" topicKey="CEU"/>

</subscriptions>

</topicsSectionAWS>
```

The “topicsSectionAWS” section is used to configure the ServiceBus custom settings.

The <regions> group of settings defines which regions are managed by the current instance. For each region (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related service bus. This field is mandatory.
- *value*: this field is not used at all by the application and it should be a meaning description of the region. This field is not mandatory.
- *isLocal*: this field indicates if the region is the one of the current installation.
- *sqsUrl*: the URL of Amazon Simple Notification Service (Amazon SNS) of the current region.
- *snsUrl*: the URL of Amazon Simple Queue service (Amazon SQS) for the current region.
- *s3Url*: the URL of Amazon Simple Storage Service (Amazon S3) for the current region. This field is mandatory also if it is not used in vCloud implementation.
- *ec2Url*: the URL of Amazon Elastic Compute Cloud (Amazon EC2) for the current region. This field is mandatory also if it is not used in vCloud implementation.

The <buckets> group of settings defines which buckets are defined in Amazon S3 for each region. For each bucket (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related region.
- *value*: the name of the bucket. This field is mandatory.
- *regionKey*: the key of the father region. This field is mandatory.

The <topics> group of settings defines which topics are defined in Amazon SNS for each region. For each topic (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related region.
- *value*: the name of the topic. This field is mandatory.
- *regionKey*: the key of the father region. This field is mandatory.

The <subscriptions> group of settings defines which subscriptions are defined for each topic in Amazon SNS. For each subscription (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related topic. This field is mandatory.
- *value*: the name of the subscription. This field is mandatory.
- *topicKey*: the key of the father topic. This field is mandatory.

Note: In case of a mandatory field is not set in the config file, a *System.Configuration.ConfigurationErrorsException* is thrown with the message: *Required attribute 'field' not found.*

Azure Topic section

```
<topicsSectionAzure>

<servicebuses>

<add key="EU1servicebus" namespaceAddress="sbgbeu1" issuerName="owner"
issuerKey="fwb+mIFPONWJyZQVV30qpkfYfgyUr+qoqlpIO8GfZMc="/>

<add key="EU2servicebus" namespaceAddress="sbgbeu2" issuerName="owner"
issuerKey="2tTNeX26Hjd6yFT3xA656GQ/NbglY3DwUo2/6b396bE="/>

<add key="APAC1servicebus" namespaceAddress="sbgbapac1" issuerName="owner"
issuerKey="a8nD9NhmAlI3g7JNqpOVA/mzMSg+i0heJ9sDEt1c6T8="/>

</servicebuses>

<topics>

<add key="APAC1" value="tfscheque" servicebusKey="APAC1servicebus"/>

<add key="EU2" value="tfscheque" servicebusKey="EU2servicebus"/>

<add key="CEU1" value="controlbus" servicebusKey="EU1servicebus" isControlBus="true"/>

<add key="EU1" value="tfscheque" servicebusKey="EU1servicebus"/>

</topics>

<subscriptions>

<add key="EU1" value="DEVIssuedAndVoided" topicKey="EU1"/>

<add key="EU2" value="DEVIssuedAndVoided" topicKey="EU2"/>

<add key="APAC1" value="DEVIssuedAndVoided" topicKey="APAC1"/>

<add key="CEU1" value="devcontrolbus" topicKey="CEU1"/>

</subscriptions>

<filters>

<add key="EU11" subscriptionKey="EU1" field="Operation" value="Issue"/>

<add key="EU12" subscriptionKey="EU1" field="Operation" value="IssueWithObject" operator="or"/>

<add key="EU13" subscriptionKey="EU1" field="Operation" value="Void" operator="or"/>

<add key="EU14" subscriptionKey="EU1" field="Operation" value="VoidWithObject" operator="or"/>

<add key="EU21" subscriptionKey="EU2" field="Operation" value="Issue"/>

<add key="EU22" subscriptionKey="EU2" field="Operation" value="IssueWithObject" operator="or"/>

<add key="EU23" subscriptionKey="EU2" field="Operation" value="Void" operator="or"/>

<add key="EU24" subscriptionKey="EU2" field="Operation" value="VoidWithObject" operator="or"/>
```

```
<add key="APAC11" subscriptionKey="APAC1" field="Operation" value="Issue"/>

<add key="APAC12" subscriptionKey="APAC1" field="Operation" value="IssueWithObject" operator="or"/>

<add key="APAC13" subscriptionKey="APAC1" field="Operation" value="Void" operator="or"/>

<add key="APAC14" subscriptionKey="APAC1" field="Operation" value="VoidWithObject" operator="or"/>

</filters>

</topicsSectionAzure>
```

The “topicsSectionAzure” section is used to configure the ServiceBus custom settings. The <servicebuses> group of settings defines which servicebuses are managed by the current instance. For each servicebus (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related servicebus.
- *nameSpaceAddress*: the name of the Service Namespace created. This field is mandatory.
- *issuerName*: the Name of the created Storage Account. This field is mandatory.
- *issuerKey*: the Primary Access Key recorded during Storage Account creation. This field is mandatory.

The <topics> group of settings defines which topics are defined for each service bus. For each topic (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related service bus.
- *value*: the name of the topic. This field is mandatory.
- *isControlBus*: this field indicates if the topic is that one used to monitor when the region applications (frontends and backend) should be stopped. This field is not mandatory and the default value is false.
- *serviceBusKey*: the key of the father service bus. This field is mandatory.

The <subscriptions> group of settings defines which subscriptions are defined for each topic. For each subscription (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related topic. This field is mandatory.
- *value*: the name of the subscription. This field is mandatory.
- *topicKey*: the key of the father topic. This field is mandatory.

The <filters> group of settings defines which filters are defined for each subscription. For each filter (<add> element), following information has to be set:

- *key*: the key identifier for the add element. This is an internal information used by the application to connect a child element to the related subscription. This field is mandatory.
- *field*: the message property used to define the filter in subscription. This field is mandatory.
- *value*: the value of the message property defined with equals operator. This field is mandatory.
- *operator*: the logical operator used to concatenate multiple filters. Possible values are "and" and "or". This field is not mandatory in case of only one filter is defined.
- *subscriptionKey*: the key of the father topic. This field is mandatory.

In case of subscription with "Issue", "Void" and "IssueWithObject" as possible values for "Operation" message property, following <add> elements should be added to filters element:

```
<add key="key1" subscriptionKey="subscriptionKey" field="Operation" value="Issue"/>

<add key="key2" subscriptionKey="subscriptionKey" field="Operation" value="IssueWithObject" operator="or"/>

<add key="key3" subscriptionKey="subscriptionKey" field="Operation" value="Void" operator="or"/>
```

Note: In case of a mandatory field is not set in the configuration file, a *System.Configuration.ConfigurationErrorsException* is thrown with the message: *Required attribute 'field' not found.*

AppSettings

```
  <appSettings>

    <add key=" PrefetchCount " value="200" />

    <add key="Receiver" value="VCLOUDEU2" />

    <add key="ReceiverThreadNumber" value="10" />

    <add key="ReceiverMessageNumber" value="10" />

    <add key="MessageVisibilityTimeout" value="120" />

    <add key="MessageReceivingTimeOutSeconds" value="5" />

    <add key="MessagePollIntervalSeconds" value="2" />

    <add key="PoisonReceiverThreadNumber" value="3" />

    <add key="PoisonMessageThreshold" value="3" />

    <add key="AWSAccessKey" value="AKIAJCPYEWYOIX4ZVR2Q" />

    <add key="AWSSecretKey" value="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" />
```

```
    <add key="CacheService.EnableCache" value="FALSE" />
    <add key="CacheService.Provider" value="AppFabric" />
    <add key="CacheService.ExpirationType" value="D" />
    <add key="CacheService.ExpirationValue" value="30" />
    <add key="ClientSettingsProvider.ServiceUri" value="" />
    <add key="ServiceBus.Azure.Enabled" value="TRUE" />
    <add key="ServiceBus.AWS.Enabled" value="TRUE" />
    <add key="StoragePath" value="\\localhost\Temp\"/>
    <add key="ChequeFolderName" value="cheques"/>
  </appSettings>
```

The standard <appSettings> section defines all <add> items for instance that need to be configured depending upon the environment:

- *PrefetchCount:* this field represents the number of messages that the message receiver can simultaneously receive from the server in the same round-trip.
- Receiver: The identity of current system. This field is compared with related "Sender" field in GripsCloudCoordinator in order to decide if data are coming or not from the same environment (EU1, EU2, APAC1 …).
- *ReceiverThreadNumber*: this field indicates how many threads are instanced for processing dequeued messages.
- *ReceiverMessageNumber*: this field indicates the maximum number of messages simultaneously returned by SQS. The value must be included between 1 and 10. Amazon SQS never returns more messages than this value but may return fewer.
- *MessageVisibilityTimeout*: The duration (in seconds) that the received messages from SQS queue are hidden from subsequent retrieve requests.
- *MessageReceivingTimeOutSeconds*: this field represents the timeout in seconds during polling SQS queue.
- *MessagePollIntervalSeconds*: the polling interval between each receives from SQS queue.
- *PoisonReceiverThreadNumber*: this field indicates how many threads are instanced for processing poison messages.
- *PoisonMessageThreshold*: this field represents the threshold for put a message in the poison queue. A message that has been received (but not deleted) a number of times greater than the *PoisonMessageThreshold* is put in the poison queue.
- *ClientSettingsProvider.ServiceUri*: XXXXXXXXXXXXXXXXX
- *AWSAccessKey*:XXXXXXXXXXXXXXXXXXXXXXXXXXXX
- *AWSSecretKey*:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
- *CacheService.EnableCache*: this field indicates if the cache layer is enabled. If the value is

set to true, all parameters with prefix *CacheService* are mandatory. If the value is set to false, the cache layer will be disabled.

- *CacheService.Provider*: this field sets which cache provider will be used for manage the cache layer. Two different values for the setting are recognized, one for the Windows Azure AppFabric Caching (specify APPFABRIC as value) and another one for the default in-memory cache provider offered by the .NET Framework 4 (specify INMEMORY as value).
- *CacheService.ExpirationType*: this field indicates the time measurement unit for the parameter *CacheService.ExpirationValue*. Possible values are D (days), H (hours) and M (minutes).
- *CacheService.ExpirationValue*: this field indicates how much time should elapse before the cache expires.
- *ClientSettingsProvider.ServiceUri*: XXXXXXXXXXXXXXXXXX
- *ServiceBus.Azure.Enabled*: Setting this field to true, the Azure ServiceBus will be enabled while setting it to false it will be disabled. The Azure ServiceBus can be enabled with AWS ServiceBus.
- *ServiceBus.AWS.Enabled*: Setting this field to true, the AWS ServiceBus will be enabled while setting it to false it will be disabled. The AWS ServiceBus can be enabled with Azure ServiceBus.
- *StoragePath*: the network share path of the storage.
- *ChequeFolderName*: the specific cheque folder for storing cheques.

5.3.2. GripsCloudCoordinator

AppSettings

```
<appSettings>

<add key="ChequeTopicName" value="vcloudtfscheque"/>

<add key="Sender" value="VCLOUDEU1"/>

<add key="namespaceAddress" value="sbgbeu1"/>

<add key="issuerName" value="owner"/>

<add key="issuerKey" value="fwb+mIFPONWJyZQVV30qpkfYfgyUr+qoqlpIO8GfZMc="/>

<add key="microsoft.visualstudio.teamsystems.aspnetdevserver:/"

value="1337;True;6308;1;-8588791831116947398" />

<add key="microsoft.visualstudio.teamsystems.backupinfo" value="1;web.config.backup" />

<add key="CacheService.Provider" value="AppFabric" />

<add key="CacheService.EnableCache" value="FALSE" />

<add key="CacheService.ExpirationType" value="D" />
```

```
<add key="CacheService.ExpirationValue" value="30" />

<add key="AWSAccessKey" value="AKIAJCPYEWYOIX4ZVR2Q" />

<add key="AWSSecretKey" value="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" />

<add key="AWSSNSServiceURL" value="https://sns.eu-west-1.amazonaws.com" />

<add key="ServiceBus" value="AZURE" />

<add key="StoragePath" value="\\localhost\Temp\"/>

<add key="StoragePathHttp" value="http://localhost/Temp/"/>

<add key="ChequeFolderName" value="cheques"/>

</appSettings>
```

- *ChequeTopicName:* this field is the name of the specific topic for cheques configured on Windows Azure AppFabric Service Bus.
- *Sender*: The identity of current system. This field is compared with related "Receiver" field in GripsCloudCoordinator in order to decide if data are coming or not from the same environment (EU1, EU2, APAC1 …).
- *issuerKey*: the issuer key of the application namespace created in Windows Azure AppFabric Service Bus, needed in authentication with Access Control.
- *issuerName*: the issuer name of the application namespace created in Windows Azure AppFabric Service Bus, needed in authentication with Access Control.
- *namespaceAddress:* the namespace address of the namespace application created in Windows Azure AppFabric Service Bus.
- *microsoft.visualstudio.teamsystems.aspnetdevserver:/*: XXXXXXXXXXXX
- *microsoft.visualstudio.teamsystems.backupinfo*:XXXXXXXXXXXXXXX
- *CacheService.EnableCache*: this field indicates if the cache layer is enabled. If the value is set to true, all parameters with prefix *CacheService* are mandatory. If the value is set to false, the cache layer will be disabled.
- *CacheService.Provider*: this field sets which cache provider will be used for manage the cache layer. Two different values for the setting are recognized, one for the Windows Azure AppFabric Caching (specify APPFABRIC as value) and another one for the default in-memory cache provider offered by the .NET Framework 4 (specify INMEMORY as value).
- *CacheService.ExpirationType*: this field indicates the time measurement unit for the parameter *CacheService.ExpirationValue*. Possible values are D (days), H (hours) and M (minutes).
- *CacheService.ExpirationValue*: this field indicates how much time should elapse before the cache expires.
- *AWSAccessKey*:XXXXXXXXXXXXXXXXXXXXXXXXXXXX
- *AWSSecretKey*:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
- *AWSSNSServiceURL*:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
- *ServiceBus:* Possible values for this field are: "AZURE", "AZUREFULL" and "AWS". In the

"AZURE" and "AZUREFULL" modes, Azure AppFabric ServiceBus will be used as ServiceBus while in the "AWS" mode Amazon Simple Notification Service will be used. In the "AZUERE" mode GripsCloudCoordinator will save all cheques in the local BlobStorage and it will send to the ServiceBus all information needed to retrieve data from Blob Storage and store data where requested (remote Blob Staorage and SQL Azure). In the "AZUREFULL" mode, GripsCloudCoordinator will send to ServiceBus the full cheque and the MessageBusReceiver will store it directly where requested.

- *StoragePath*: the network share path of the storage.
- *StoragePathHttp:* the http address for the network share path of the storage.
- *ChequeFolderName*: the specific cheque folder for storing cheques.

AWS TopicSection

```
<topicsSectionAWS>

<regions>

<add key="EU" value="EU West" isLocal="true" sqsUrl="https://eu-west-1.queue.amazonaws.com" snsUrl="https://sns.eu-west-1.amazonaws.com" s3Url="s3-eu-west-1.amazonaws.com" ec2Url="http://ec2.eu-west-1.amazonaws.com"/>

</regions>

<topics>

<add key="EU" value="vcloudtfschequeeu" arn="arn:aws:sns:eu-west-1:648522325131:vcloudtfschequeeu" regionKey="EU" />

</topics>

</topicsSectionAWS>
```

The "topicsSectionAWS" section is used to configure the ServiceBus custom settings. This section is the same described in paragraph "MessageBus Receiver – AWS Topic Section". The only difference is that in GripsCloudoordinator only the local region (with "isLocal" set to true) need to be configured.

CONCLUSION

In this project we have used different technologies in order to produce a service which would be available to different clients around the world. We have used Azure form Microsoft, AWS from amazon and VMware Cloud. But the real competition was between Microsoft and Amazon.

The Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) markets are growing faster than ever. But sometimes it seems like startups base their infrastructure decisions on a popularity contest or random selection. Among the two gorillas in the cloud space, some developers swear by Amazon, others think Azure is the best, but often the details are sparse as to why one option is better than the other.

The real choice is between IaaS and PaaS. If we were interested in an IaaS cloud, we would have definitely considered AWS.

However, the real win for us was getting access to an environment that was as managed as possible, so we are spending our time and money focused on solving the business problem instead of creating and managing our various environments for AWS.

Since time to market was important and we had .NET skills at the time, it didn't make sense to gamble on other technologies.

It is possible to build a sufficiently redundant failover strategy on either cloud with comparable overall uptime statistics, but it's just a lot easier to do it on Azure.

It is easier for the average engineer to accomplish high availability on Azure because the tools push us down that path and make it pretty easy to build stateless applications.

From performance point of view, we have seen that AWS had better results. In the read/write case, it is far simpler to deploy very fast high-transaction data-stores like Apache Cassandra, or to use Amazon SimpleDB, which is built on top of Amazon's Dynamo data store. In other words, we don't have to setup our own NoSQL implementation based on CQRS and Azure blobs or tables.

And also working with big data tools, Since we built out our own NoSQL implementation based on MongoDB and Azure blobs and tables, we are on our way there, but the lack of a fast, horizontally scalable lookup service that is native to the architecture is something they need to fix.

There is an outlet valve though. It's probably possible to stand up our own OS instances in Azure and to install and manage our own MongoDB implementation if push came to shove. The only issue is that with that, we would be crossing the line from PaaS to IaaS and weakening the value proposition of our cloud choice. At the end we have chosen to use SQL Azure for our Azure Solution. On the other hand Amazon does the big data thing really well. Extra-Large Memory instances, High-CPU instances, Cluster Compute instances. They can't be beat right now. Amazon definitely provides greater variability for different big data applications. Every big data problem requires different underlying capabilities — this is definitely a

case where a one-size-fits-all solution might not be a solution at all.

At the end, recalling that our priority was to make this service highly available and performance was our second concern, our proof of concept in this project came to this conclusion that Microsoft Azure is a better solution to this problem.

REFERENCES

1. Programming WCF Services: Mastering WCF and the Azure AppFabric Service Bus (book), Author : Juval Lowy,publisher: o'reilly, published : August 30, 2010
2. Cloud Computing with the Windows Azure Platform(book), Author : Roger Jenings, publisher: Wrox, published : October 5, 2009
3. Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB (book), Author: James Murty, publisher: o'reilly, published: April 1, 2008
4. Automapper, http://automapper.org/
5. ADO.net, http://blogs.msdn.com/b/adonet/
6. Enterprise Library 5, http://msdn.microsoft.com/en-us/library/ff632023.aspx
7. Service Stack, http://servicestack.net/
8. Web.config Transformation Syntax for Web Application Project Deployment http://msdn.microsoft.com/it-it/library/dd465326.aspx
9. Windows Communication Foundation(.NET4): http://msdn.microsoft.com/en-us/library/dd456779.aspx
10. REST in Windows Communication Foundation(WCF): http://msdn.microsoft.com/en-us/netframework/cc950529
11. Managed Extensibility Framework http://msdn.microsoft.com/en-us/windowsserver/ee695849
12. blob storage: http://www.windowsazure.com/en-us/home/tour/storage
13. amazon elastic compute cloud(EC2): http://aws.amazon.com/ec2/
14. amazon simple notification service(SNS): http://aws.amazon.com/sns/
15. amazon simple queue service(SQS): http://aws.amazon.com/sqs/
16. amazon simple storage service(S3): http://aws.amazon.com/s3/
17. Amazon relational database service(RDS): http://aws.amazon.com/rds/
18. amazon ElastiCache: http://aws.amazon.com/elasticache/
19. amazon simpleDB: http://aws.amazon.com/simpledb/
20. amazon cloudFormation: http://aws.amazon.com/cloudformation/