Version: 0.1.0. Draft

Author:

Date: August 23, 2017

Contents

[Glossary, definition of terms 4](#_Toc483400728)

[Introduction 5](#_Toc483400729)

[Prerequisites 6](#_Toc483400730)

[System integration schema 7](#_Toc483400731)

[Installation 9](#_Toc483400732)

[Database 9](#_Toc483400733)

[SQL Server 9](#_Toc483400734)

[MongoDB 10](#_Toc483400735)

[Website installation 10](#_Toc483400736)

[Identity server 11](#_Toc483400737)

[Digital Edge API 12](#_Toc483400738)

[Digital Edge Web 17](#_Toc483400739)

[Digital Edge WebAdminAPI 18](#_Toc483400740)

[Digital Edge Web Admin 18](#_Toc483400741)

[Taskhost windows service 19](#_Toc483400742)

[Installation 19](#_Toc483400743)

[Configuration 19](#_Toc483400744)

[Upgrading the existing installation 19](#_Toc483400745)

[Development installation 19](#_Toc483400746)

[Test/Production environment 19](#_Toc483400747)

[Testing 19](#_Toc483400748)

[POSTMAN 21](#_Toc483400749)

[DE Web site test 23](#_Toc483400750)

[Appendix 1: installations 24](#_Toc483400751)

[Windows application server roles and features installation 24](#_Toc483400752)

[Network load balancing service 27](#_Toc483400753)

[Web Platform Installer 5 31](#_Toc483400754)

[ARR (Application Request Routing module) 31](#_Toc483400755)

[Install ARR 32](#_Toc483400756)

[Configure ARR 32](#_Toc483400757)

[Service Bus 1.1 35](#_Toc483400758)

[Service Bus 1.1 Installation 35](#_Toc483400759)

[Service Bus 1.1. Configuration 35](#_Toc483400760)

[APP Fabric Installation 41](#_Toc483400761)

[AppFabric Installation 41](#_Toc483400762)

[SQL Server installation 47](#_Toc483400763)

[MongoDB installation 48](#_Toc483400764)

# Glossary, definition of terms

API – Application programming Interface

WebAPI – Application programming interface exposed as REST services over http/https

REST - representational state transfer

BCS – Banking Core System

DE – Digital Edge product

DE Web – Digital Edge web front end application

DE WebAPI – Digital Edge REST API services

DE WebAdmin – Digital Edge web management tool

DE WebAdminAPI – Digital Edge web management REST API services

DE Identity – Identity server REST API services for token based access

IIS – Internet informational services, Microsoft Web server

ARR – Application request routing services, part of Microsoft’s IIS

DMBS – Database management system

CI – Continuous Integration - development environment where every code change is deployed and tested automatically.

# Introduction

DE is set of web applications backed by SQL Server, and optionally MongoDB, database engine. Functionalities, offered by BCS and DE infrastructure itself, are hosted in IIS as REST services which are used to standardize functionalities between frontend client applications and backend services. System use Identity server for token based access which is integrated in infrastructure and its functionalities are also available as REST services.

Digital Edge Web site is frontend client for Digital Edge infrastructure (Digital Edge API). Other client may be a mobile application, which requires that DE API and DE Identity server are available over Internet.

Besides Identity, API and Web, DE exposes administration REST set of services for controlling and maintaining DE along with WebAdmin web user interface application.

DE is developed in.NetFramework 4.5 C#, using WebAPI 2.0 and MVC Razor technology communicating by JSON objects. DE service implementations are using EntityFramework, JSON and BSon object types and automapper primarily.

# Prerequisites

Prerequisites are listed below. Depending on environment and projected load, DE host system may have none, one or two web gates, one or two web servers, one or more WebAPI servers including all backing services required for operation such as AppFabric and ServiceBus, one (SQL Server only installation) or more database servers with one SQL Server and one or more MongoDB servers.

* Windows Server 2012R2 or newer operating system installed on dedicated web and database server(s). Number and power of servers depend on each installation and projected load.
* .Net framework 4.5 should be installed on all servers.
* IIS 7 or newer including application server from features with all.Net framework versions installed on dedicated web servers.

Initial setup assume that DE WebAPI and DE Web sites are hosted on separate servers. More powerful web server should host application middle layer contained of following applications DEWebAPI, DEIdentity, DeWebAdmin and DEWebAdminAPI, along with DE Task service. AppFabric 1.1 Caching service and Service bus 1.1 should be installed also. If multiple servers are used for middle tier, machines should form NLB cluster, and AppFabric and Service bus services should be installed all middle tier machines, and configured as own service cluster.

Less powerful server should host DE Web application. Installation can have multiple web servers in which case additional low power web server can be used as gate with ARR support server

* Database management system. Dedicated database server for hosting each DBMS is recommended.
  + SQL Server 2012 or better(Express edition is sufficient if accompanied with separate MongoDB machine, otherwise Standard edition with 24Gb RAM available for SQL Server)
  + MongoDB with 24Gb of RAM available for Mongos process, or 3 node configuration each having at least 12Gb RAM
* At least 4Gb RAM available for each worker process on each server (WebSite, WebAPI)
* SSL certificate for https access
* Client machine with W7 or better with Chrome and Postman extension for testing purposes

# System integration schema

Two approaches are planned for DE environment, and these configurations depend only on performance requirements. First recommended approach is to have two DBMS, for separating configuration from operational data. Such scenario is presented on schema below.



One or more servers can be used in each server group. Actual number of servers depends on system requirements.

Please see following table for recommended server group configuration.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **System name** | **OS** | **vCPU count** | **vRAM size** | **Storage size** | **Storage Category** | **Category** | **Network** |
| 1 | hub.ods | hub.db1 | Windows 2012 R2 Standard or Linux  MongoDB Write node | 4 | 16 Gb | 1 TB  At least four separate LUNs  ??? | HDD, 15k drives or better | High availability | 2 x 1 Gbps |
| hub.db2 | Windows 2012 R2 Standard or Linux  MongoDB read node | 4 | 16 Gb |
| hub.db3 | Windows 2012 R2 Standard or Linux  MongoDB read node | 4 | 16 Gb |
| 2 | hub.db | hub.db | Windows 2012 R2 Standard  MS SQL 2012 SQL Express or better | 8 | 32 Gb | Active / Passive  ??? | 2 x 1 Gbps or better |
| 2 | hub.gw | hub.gw1 | Windows 2012 R2 Standard  IIS 7.5 or better & ARR | 2 | 4 Gb | 80Gb | HDD | NLB HW or MS NLB sw | 1Gbps or better |
| hub.gw2 | Windows 2012 R2 Standard  IIS 7.5 or better & ARR | 2 | 4 Gb | 80Gb | HDD | 1Gbps or better |
| 3 | hub.web | hub.web1 | Windows 2012 R2 Standard  IIS 7.5 or better | 4 | 8 Gb | 80Gb | HDD | MS NLB sw? | 1Gbps or better |
| hub.web2 | Windows 2012 R2 Standard  IIS 7.5 or better | 4 | 8 Gb | 80Gb | HDD | 1Gbps or better |
| 4 | hub.app | hub.app1 | Windows 2012 R2 Standard  IIS 7.5 or better  AppFabric 1.1  ServiceBus 1.1 | 8 | 16 Gb | 80Gb | HDD | MS NLB sw | 1Gbps or better |
| hub.app2 | Windows 2012 R2 Standard  IIS 7.5 or better  AppFabric 1.1  ServiceBus 1.1 | 8 | 16 Gb | 80 Gb | HDD | 1Gbps or better |
| 5 | hub.metric | hub.profile | Ubuntu Linux 12.04  Piwik | 2 | 4 GB | 300 GB | HDD | **Optional** | 1 Gbps |

# Installation

All DE application components can be installed on single or multiple servers. Installation has same procedure for all servers, and later those servers are joined in NLB for optimal performance

## Database

### SQL Server

MCHUB database contains configuration and operational data. Additionally, when MongoDB is not used, SQL Server can be used to store synchronized data as well.

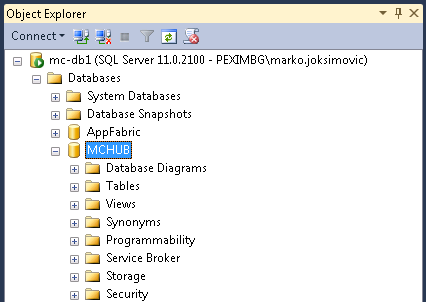
#### Installation highlights

During SQL Server installation, choose Latin 1 General collation which is accent sensitive and case insensitive. Make sure that at least 3 drives/LUNs are used, each for temp, log and data file store. Also, create at least 4 temp data files, adjust Max Degree of Parallelism to number which is half of the number of logical processors available to SQL Server and verify that power options are set to performance setting. All these general guidelines can be found in Microsoft best practices regarding SQL Server.

More information regarding SQL Server installation can be found in [Appendix 1: SQL Server Installation](#_SQL_Server_installation).

#### Database deployment

MCHUB database is required for DE WebAPI and Identity server operations. It can be deployed by simple database restore, or by using migrations or prepared SQL scripts. Required DBMS are SQL Server or Oracle DB.



Once MCHUB database is deployed, please make sure that it can be accessed from web server hosting the Identity and/or MCAPI.

### MongoDB

MongoDB is intended to hold operational synchronized data only.

#### Installation higlights

Please install MongoDB on dedicated server (or servers depending of usage scenario and projected load) using MongoDB documentation available online. More information regarding MongoDB installation can be found in [Appendix 1: MongoDB installation](#_MongoDB_installation_1).

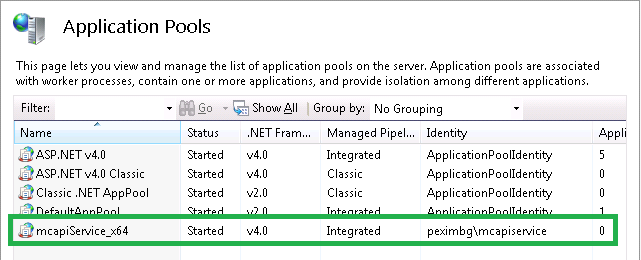
#### Database deployment

No actual deployment is done except data store location specified during installation. All required collections will be generated automatically on first use.

## Website installation

Websites (on both API and WEB servers) are installed in IIS. More information regarding required windows features required for web server can be found in [Appendix 1: Windows server features installation](#_Windows_server_features).

Installation will require a person who is proficient in IIS. Please create separate application pool for each web site (DE Web, DE WebAPI, DE Identity, DEWebAdminAPI and DEWebAdmin ) on each dedicated server. It is recommended that you choose more powerful server as “application server” which will host DE WebAPI, DE Indentity, DE WebAdmin and DE WebAdminAPI, along with AppFabric cache, Service bus and DETaskHost service. Each , two application pools, one for Identity, one for MCAPI. Try to use names which will distinct them easily during application creation.



Process model for Identity application pool can use ApplicationPoolIdentity credentials, so create new pool exactly the same as ASP.NET v4.0 application pool.

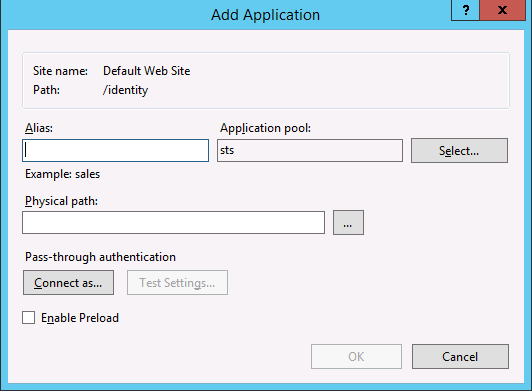
If there is an issue of adding Network service account to have sufficient user rights on SQL Server or using SQL authentication in general, MCAPI application pool can use domain user name with sufficient user rights on MCHUB database. We recommend that account used for access has granted db\_owner role membership. SQL server authorization is **not** recommended.

### Identity server

Extract Identity server package into folder designated for web services (C:\Websites for example).

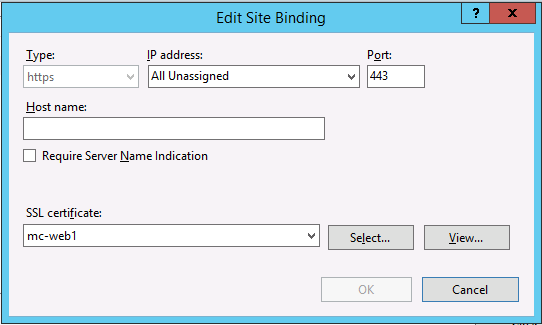
Start IIS Management console and navigate to Default web site node.

Create new application with name Identity, set Identity application pool and choose folder where you have extracted the Identity package as its physical path.



If production deployment is required, https is required for Identity access. This step requires SSL certificate to be installed on web server prior to executing this step.

Right-click Default Web Site node and choose edit bindings. Add or update the https binding and make sure SSL certificate is selected.



Update the web.config file database connection string settings to MCHUB database. Consider granting db\_owner role membership for MCHUB database to account used to run application pool with or <domain>\IIS APPPOLL<PoolName> if application pool identity is used. SQL authorization can be used, however it is considered obsolete and is less secure. Password is not encrypted in configuration file.

### Digital Edge API

Extract API package to folder where web applications are stored (for example C:\Websites\mcapi).

Using IIS management console, create new application pool (for example mcapi) and application within Default Web Site, name it mcapi, and choose its physical path and assign newly created (mcapi) application pool.

In web.config file, please update database connection strings to proper database instances. Once again, consider windows authentication and check if account used for application pool has sufficient user rights on server. We recommend that this user account used for application pool is granted db\_owner role for MCHUB database or <domain>\IIS APPPOLL<PoolName> if application pool identity is used as application pool user.

Besides connection strings, appsettings, unity and additional custom sections stated below needs to be checked.

#### appSettings

Keys of interest are highlighted yellow.

<!-- SXS -->

<add key="sxs.username" value="admin" />

<add key="sxs.password" value="admin" />

<add key="SkipPINChange" value="true" />

<add key="HideTraceDetails" value="63" />

<add key="DemoFolder" value="c:\Demo" />

<add key="DemoTrace" value="False" />

<add key="Identity" value="http://mc-web1/identity/" />

<add key="UploadTempFolder" value="UploadTemp"/>

<add key="EnableResponseSaver" value="true"/>

<!--SMS OTP-->

<add key="SMSOTP.ExpirationInMinutes" value="2" />

<add key="SMSOTP.MaximumResendCount" value="5" />

<add key="SMSOTP.Show" value="true"/>

<!--ORACLE-->

<add key="AccountingApiPackage" value="ACCOUNTINGAPI" />

<add key="AccountDataApiPackage" value="IF6ACCOUNTDATAAPI\_PKG" />

<!--NOTIFICATION HUB-->

<add key="NotificationHubConnectionString" value="Endpoint=sb://pushdemonamespace.servicebus.windows.net/;SharedAccessKeyName=DefaultFullSharedAccessSignature;SharedAccessKey=48K3jiG0hWnmJ2CBZkejOibqA2CJTaZOlWh8ueY+KUE=" />

<add key="NotificationHubPath" value="RaiffPoC" />

<!--Sts app setings-->

<add key="sts-api-url" value="http://mc-web1/mcapi/" />

<add key="sts-url" value="http://mc-web1/identity/connect/token" />

<add key="sts-client-id" value="multichannelclientcredentials"/>

<add key="sts-client-secret" value="myrandomclientsecret"/>

<add key="sts-preferred-client-culture" value="en-us"/>

<add key="sts-scope" value="multichannelmanagement"/>

<!--SYNCHRONIZATION-->

<add key="sts.url" value="http://mc-web1/identity/connect/token" />

<add key="api.url" value="http://mc-web1/mcapi/" />

<add key="sts.customer.username" value="<*username*>" />

<add key="sts.customer.password" value="<*password*>" />

<add key="sts.client.id" value="multichannelropc" />

<add key="sts.client.secret" value="myrandomclientsecret" />

<add key="sts.client.scope" value="openid multichannelmanagement" />

<add key="sts.client.prefered-culture" value="en-US" />

<!-- Mongo & Elastic -->

<add key="mongo-db.connection-string" value="mongodb://172.16.88.249:27017" />

<add key="mongo-db.ods-database" value="MC-ODS" />

<add key="mongo-db.account-data-database" value="MC-ODS-AccountData" />

<add key="mongo-db.arrangement-database" value="MC-ODS-Arrangement" />

<add key="mongo-db.party-database" value="MC-ODS-Party" />

<add key="mongo-db.payment-database" value="MC-ODS-Payment" />

<add key="mongo-db.product-database" value="MC-ODS-Product" />

<add key="mongo-db.profile-database" value="MC-ODS-Profile" />

<add key="api-base-url" value="http://mc-web1/mcapi/" />

<add key="arrangement-api-path" value="v1/arrangement" />

<add key="accounting-api-path" value="v1/accounting" />

Highlighted setting represents values which are related to the current DE development installation and may differ on other target environment. Identity key URL casing MUST match to webserver URL.

Other keys may exist, depending on custom service implementations which may use them, however those settings will be explicitly described in “How to” document supplied along with implementation release.

In case of CoreAPI installation it is required to set context key in order to accept trust credentials form other identity.

<add key="CoreApi.ContextOverride" value="true" />

#### Unity

Other important configuration block is Unity. It contains definitions on assemblies which needs to be loaded in application domain and specific implementations to be used for each service. In case that new service and implementation is added to the WebAPI, this section needs to be updated.

Example of Unity’s block first part which specifies assemblies to be loaded:

<unity xmlns="http://schemas.microsoft.com/practices/2010/unity">

<assembly name="Experience.Contracts" />

<assembly name="Experience.WebApi.Extensions" />

<assembly name="Experience.Services.Classifications" />

<assembly name="Experience.EntityFramework.Extensions" />

<assembly name="Experience.Synchronization.Common" />

<assembly name="Experience.Synchronization.ODS" />

<assembly name="Experience.AccountDataAPI.Contracts" />

<assembly name="Experience.AccountDataAPI.Services.Oracle" />

<assembly name="Experience.AccountDataAPI.Services.Hub" />

<assembly name="Experience.AccountDataAPI.Controllers" />

<assembly name="Experience.AccountDataAPI.Synchronization" />

<assembly name="Experience.AccountDataAPI.Services.Proxy" />

…

Within this add new assemblies if required.

Other, larger part of Unity block specifies implementation of each service. If new service is added, this is the place where its implementation is specified.

<assembly name="Experience.UserManagementAPI.Contracts" />

<assembly name="Experience.UserManagementAPI.Services" />

<assembly name="Experience.UserManagementAPI.Controllers" />

<container>

<!--Translation Service-->

<register type="Experience.Contracts.ServiceInterfaces.ITranslationService" mapTo="Experience.Services.Classifications.ClassificationTranslationService"></register>

<!--EventLoggers(s)-->

<register type="Experience.Contracts.Events.Interfaces.IEventLogWriter" mapTo="Experience.EntityFramework.Extensions.Events.EventLogger"></register>

<register name="EventsAutomapperConfigurationProvider" type="Experience.Contracts.ServiceContracts.IAutomapperConfigurationProvider" mapTo="Experience.EntityFramework.Extensions.Events.AutoMapper.AutomapperConfigurationProvider"></register>

<!--EventsConfigurationProvider(s)-->

<register type="Experience.Contracts.EventsManagement.Interfaces.IEventsSpecificationsProvider" mapTo="Experience.EntityFramework.Extensions.EventsManagement.EventsSpecificationsProvider"></register>

<register name="EventsManagementAutomapperConfigurationProvider" type="Experience.Contracts.ServiceContracts.IAutomapperConfigurationProvider" mapTo="Experience.EntityFramework.Extensions.EventsManagement.AutoMapper.AutomapperConfigurationProvider"></register>

<!-- ODS AccountDataAPI -->

<register type="Experience.AccountDataAPI.Contracts.V1.ServiceContracts.IAccountDataAPIQueryService" mapTo="Experience.AccountDataAPI.Synchronization.V1.Implemetations.AccountDataAPIQueryServiceOds"></register>

<register type="Experience.AccountDataAPI.Contracts.V1.ServiceContracts.IAccountDataAPICommandService" mapTo="Experience.AccountDataAPI.Synchronization.V1.Implemetations.AccountDataAPICommandServiceOds"></register>

<register name="AccountDataAutomapperConfigurationProvider" type="Experience.Contracts.ServiceContracts.IAutomapperConfigurationProvider" mapTo="Experience.AccountDataAPI.Services.Hub.V1.AccountDataAPIAutomapperConfigurationProvider"></register>

The “type” keyword represents interface providing methods to be implemented, and “mapTo” is service implementation class which actually contains code which will perform desired action.

#### Connection strings

Example of section containing connection strings is below.

<connectionStrings>

<!--HUB-->

<add name="Events" connectionString="Data Source=MC-DB1;Initial Catalog=<MCHUB database name>;Integrated Security=False;User ID=sa;Password=*<Password>*;MultipleActiveResultSets=True;Application Name=MC" providerName="System.Data.SqlClient" />

<add name="EventsManagement" connectionString="Data Source=MC-DB1;Initial Catalog=<MCHUB database name>;Integrated Security=False;User ID=sa;Password=*<Password>*;MultipleActiveResultSets=True;Application Name=MC" providerName="System.Data.SqlClient" />

<add name="Arrangement" connectionString="Data Source=MC-DB1;Initial Catalog=<MCHUB database name>;Integrated Security=False;User ID=sa;Password=*<Password>*;MultipleActiveResultSets=True;Application Name=MC" providerName="System.Data.SqlClient" />

Please note that example is using SQL Server authorization which may not be allowed by target site implementation policy. We recommend integrated security to be used.

#### Custom ConfigSections

Configuration file contains two custom configuration sections used for content and ODS.

ContentAPI security settings default values are presented below.

<ContentSettings DisableContentSettings="false" RestrictToUserFolders="true" PublicContentFolders="/public;/temp;/products;/statements" SuperUsers ="DU1234;" UploadTempFolder="UploadTemp" />

Feature can be disabled if entire config section is omitted or “DisableContentSettings” is set to true. Restricting to user folders enforces the rule that all users can use only folder below their home folder. Home folder for user is consider to be root folder named as customer name. Superusers are excluded from this behavior, as well as declared public folders. Upload temp folder is homefolder on webserver where actual uploaded file is stored during upload. ApplicationPool’s account need to have sufficient user rights to manage files in this folder.

ODS config is part where default ODS configuration is. It features ODS operation mode, timeout in ms for QuasiOnline mode and ODS and Core implementation. ODS implementation can be MongoDB service implementation or HUB service implementation depending on environment configuration, and it refers to synchronization data store. Core implementation is service implementation which handles BCS. Example of ApiODSConfig is presented below.

<ApiOdsConfig>

<add name = "PartyApiQuery" >

<ods-configuration timeout="500" mode="Offline" ods-implementation = "Experience.PartyAPI.Services.Hub.V1.PartyAPIQueryServiceMongo, Experience.PartyAPI.Services.Hub"  core-implementation = "Experience.PartyAPI.Services.Hub.V1.PartyAPIQueryService, Experience.PartyAPI.Services.Hub"/>

</add>

Currently it is possible to specify different ODS or Core implementation for interface, however this semantically is not correct and should not be done.

### Digital Edge Web

Extract WebSite package to folder where web applications are stored (for example C:\Websites\mcweb).

Using IIS management console, create new application pool, and application within Default Web Site with name mcweb, and choose its physical path and newly created application pool (for example mcweb application pool).

In web.config file, please update database connection strings to proper database instances. There is a group for each core system, HUB, Experience, BAPO or PUB, and try to keep these groups separated for easier navigation in the future.

#### AppSettings

Following node key values represent general settings for DE Web configuration.

<add key="Multichannel.API" value="http://mc-web1/mcapi/" />

<add key="Multichannel.STS.Origin" value="http://mc-web1/identity" />

<add key="Theme" value="Standard" />

<add key="ConfigurationPath" value="C:\websites\mcweb\" />

<add key="LocalCurrency" value="EUR" />

<add key="LocalCountry" value="RS" />

<add key="Languages" value="en-US" />

<add key="RSSFeedLink" value="https://www.b92.net/info/rss/vesti.xml" />

<add key="AppMarCacheName" value="default" />

Digital Edge Web uses AppFabric Cache service as session and cache store. This setting is mandatory in all multiple web servers hosting DE Web scenarios.

#### Custom configSections

To make it work, make sure that following section is added to configSection group node:

<section name="dataCacheClient"

 type="Microsoft.ApplicationServer.Caching.DataCacheClientSection, Microsoft.ApplicationServer.Caching.Core, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"

 allowLocation="true"

 allowDefinition="Everywhere"/>

In addition, dataCacheClient block should look like:

<dataCacheClient requestTimeout="15000" channelOpenTimeout="3000" maxConnectionsToServer="100">

<localCache isEnabled="false" sync="TimeoutBased" ttlValue="300" objectCount="10000" />

<clientNotification pollInterval="5" maxQueueLength="10000" />

<hosts>

<host name="mc-web1" cachePort="22233" />

</hosts>

<securityProperties mode="None" protectionLevel="None" />

<transportProperties connectionBufferSize="131072" maxBufferPoolSize="2147483647" maxBufferSize="2147483647" maxOutputDelay="10" channelInitializationTimeout="60000" receiveTimeout="600000" />

</dataCacheClient>

Host name and port values are dependant on environment.

#### System.web node and Unity section

Following setting will enable session to be stored in AppFabric cahce. This is mandatory for web server farm installation.

<!-- <sessionState mode="InProc" cookieless="false" timeout="20" cookieName="ibSession" /> -->

<sessionState mode="Custom" customProvider="AppFabricCacheSessionStoreProvider">

<providers>

<!-- specify the named cache for session data -->

<add

  name="AppFabricCacheSessionStoreProvider"

  type="Microsoft.ApplicationServer.Caching.DataCacheSessionStoreProvider"

  cacheName="session"

  sharedId="SC"/>

</providers>

</sessionState>

Commented line represent “in memory” session storage for single web server installation scenario only.

Following keys in Unity block switch “in memory caching” or “AppFabric caching”. Again, AppFabric caching is mandatory for web farm installation scenario. More details how to [install AppFabric caching](#_APP_Fabric_Installation) service can be found in Installation appendix.

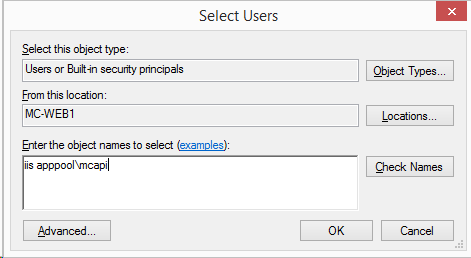
To use AppFabric caching client in web application, user account set for application pool must be added into Performance Monitor Users local security group on server hosting the web application using the client. If special “ApplicationPoolIdentity” service account is used, then open security group “Performance Monitor Users”, change Locations to current server and manually add account with following name:

IIS APPPOOL\{app pool name}

where {app pool name} needs to be replaced with real application pool name used. For example, if pool is called mcapi, try to add following account into this group:

IIS APPPOOL\mcapi

Like on below image.



Click on “Check Names” and then on “Ok” to add application pool identity to “Performance Monitor Users” group.

<!--register type="AssecoSEE.IBank.UI.Infrastructure.Caching.ICacheProvider" mapTo="AssecoSEE.IBank.UI.Infrastructure.Caching.CacheProvider" />-->

<register type="AssecoSEE.IBank.UI.Infrastructure.Caching.ICacheProvider" mapTo="AssecoSEE.IBank.UI.Infrastructure.Caching.AppFabricCacheProvider" />

Commented line represent in memory cache provider for single web server scenario only.

### Digital Edge WebAdminAPI

ConfigurationSetttings is only part that needs to be updated in WebAdmin API configuration and settings must match DE WebAPI configuration.

Please refer to DE WebAPI [connection string settings](#_Connection_strings) for more details.

### Digital Edge Web Admin

Web admin web site uses DE WebAdmin API and direct access to the database. All previously explained regarding application pool account and database user rights here apply also. Please refer to DE WebAPI [connection string settings](#_Connection_strings) for more details.

Application settings node, as in DE WeAPI, has URLs to API and identity server.

<!--Admin-->

<add key="Admin.ClientId" value="multichanneladminhybrid" />

<add key="Admin.Uri" value="http://mc-web1/mcwebadmin/" />

<!--Admin.Api-->

<add key="Admin.API.Uri" value="http://mc-web1/mcwebadminapi" />

<!--IdentityServer-->

<add key="IdentityServer.Uri" value="https://mc-web1/identity" />

## Taskhost windows service

Service is designed to be configurable single service which can manage running tasks and report status or activities of DE. Also, multiple instances of same service with different configuration handling specific user scenarios is also possible and some grouping of tasks is recommended.

### Installation

Service is distributed as zip archive and deployment is straightforward. Extract the archive and run install.bat file which will install the service.

### Configuration

Configuration description pending

## Upgrading the existing installation

Two types of installation will be referred here: Development environment and Test/Production environment where data loss is not allowed.

### Development installation

Development version installation like publish of stage or demo version within premises is done by backup and restore of the MCHUB database to new name (for example MCHUB\_Staging).

Second step is to delete existing versions of application on target folders (for example, all folders containing staging version of applications have suffixes “\_Staging”) then copy from CI application version folders to new folders (for example, to appropriate \_Staging or demo folders) and configuration update to reflect all changes made.

For database backup restore, please check [appropriate section](#_Database) above.

For configuration updates/changes also please check [respective sections](#_Website_installation) above. If folders are not available, please create folders with appropriate naming and then [start configuring web applications as described above](#_Website_installation).

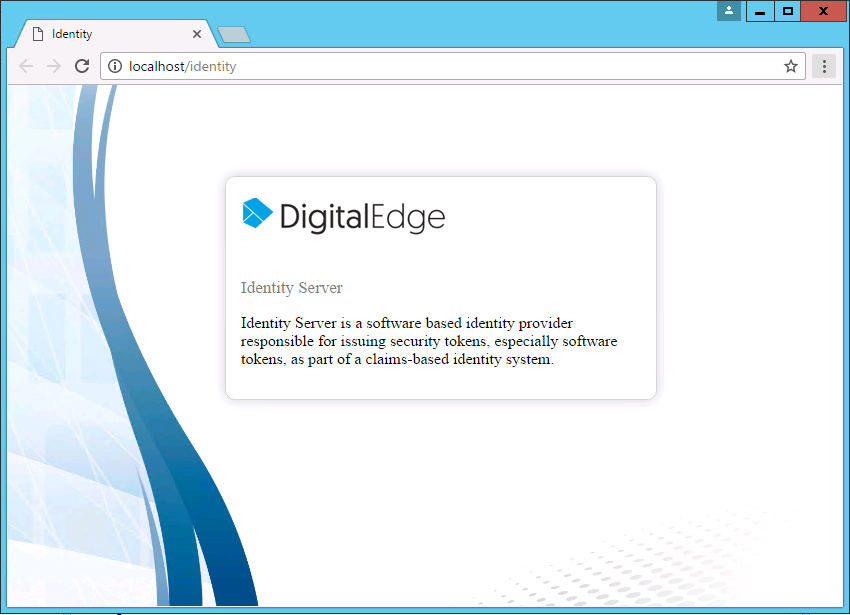
In special cases, we may provide additional scripts to be executed, but this will be covered in HowTo documents accompanying the release.

### Test/Production environment

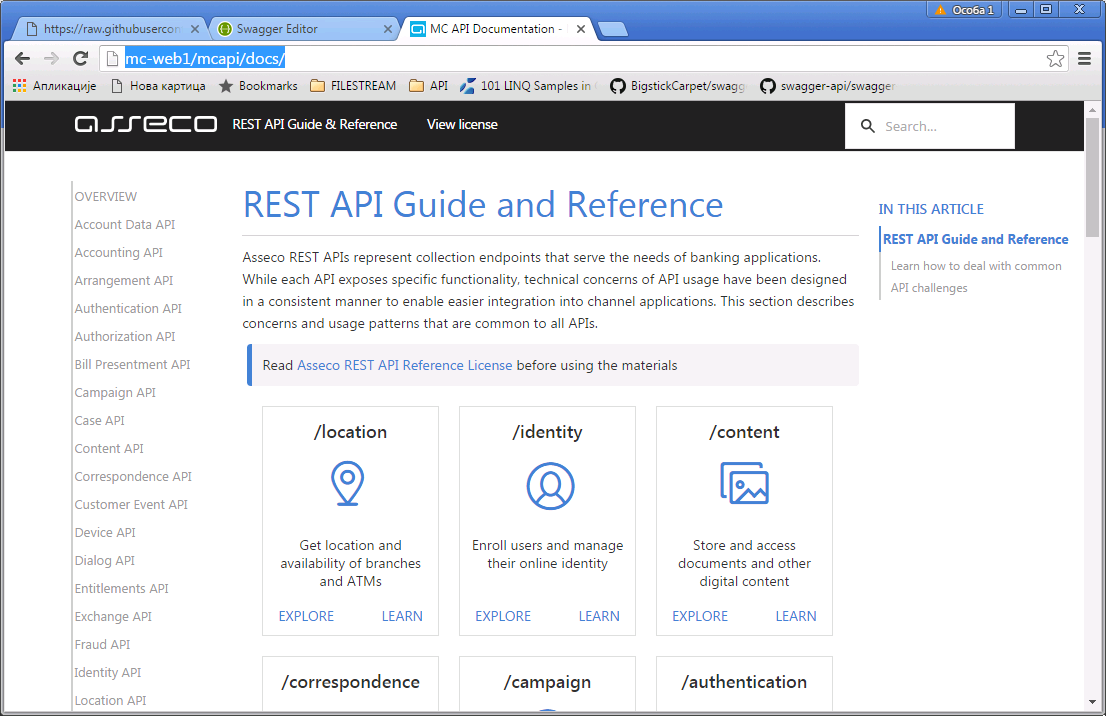
Description pending

# Testing

Successful installation is tested by POSTMAN, Chrome add-on tool for REST access. Prior to postman, initial testing of installed services can be done using simple browser access. From IIS management console expand Default Web Site node and locate Identity node. Right-click Identity web application node and choose Manage application / Browse from context menu. Identity server web site should display page similar to one on the image below.



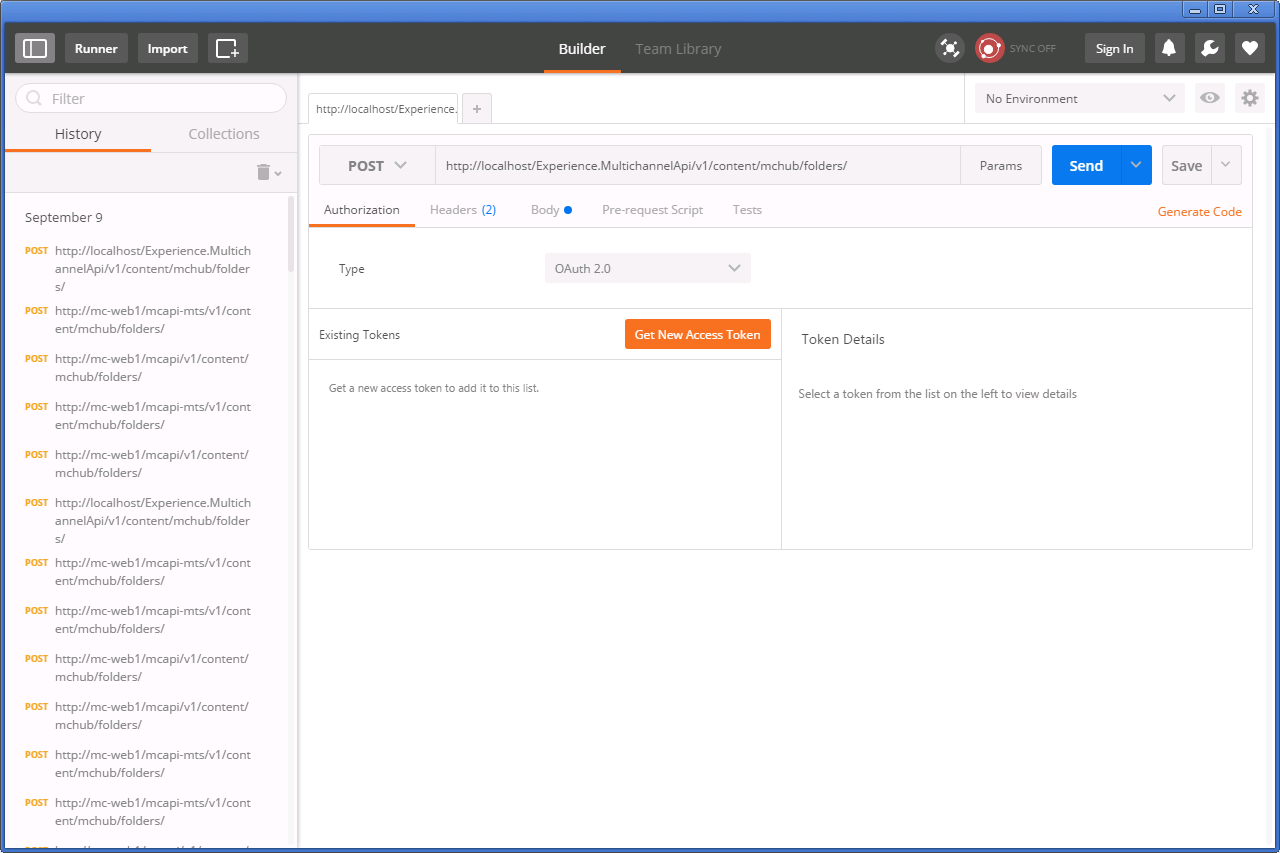
Using steps above, navigate to mcapi web site, which should display page similar to image below.



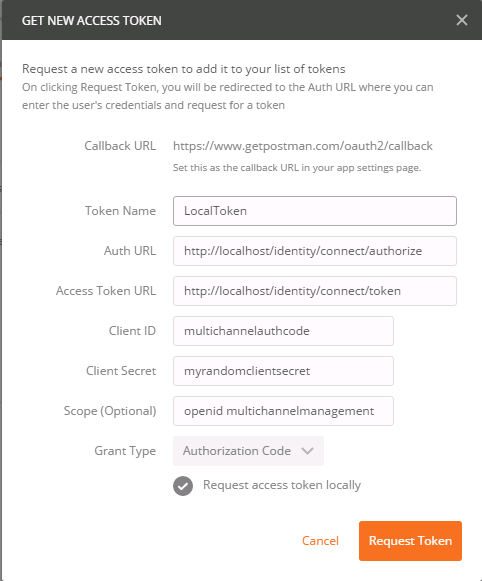
You may need to add “/docs” to the end of URL to navigate to document pages. Also, different browsers may have different fonts installed which can make display slightly differs to image above.

## POSTMAN

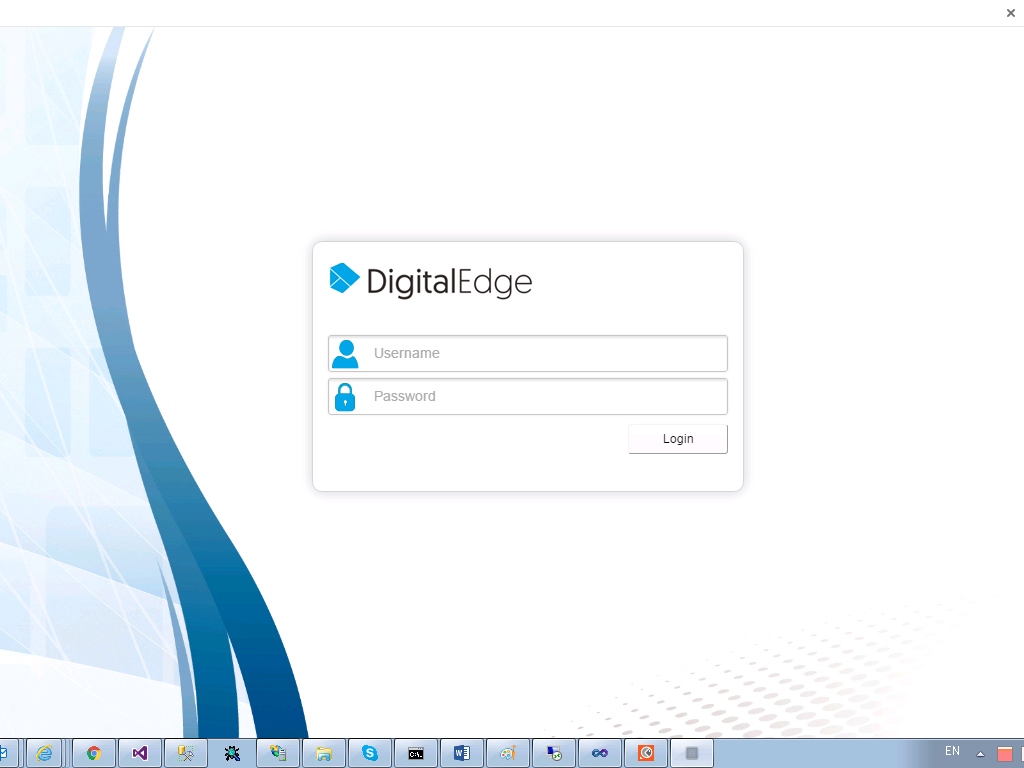
Postman is most commonly used tool for REST services access. It supports authentication and JSON objects posting and getting, including rest of popular methods such as PUT, DELETE and PATCH.



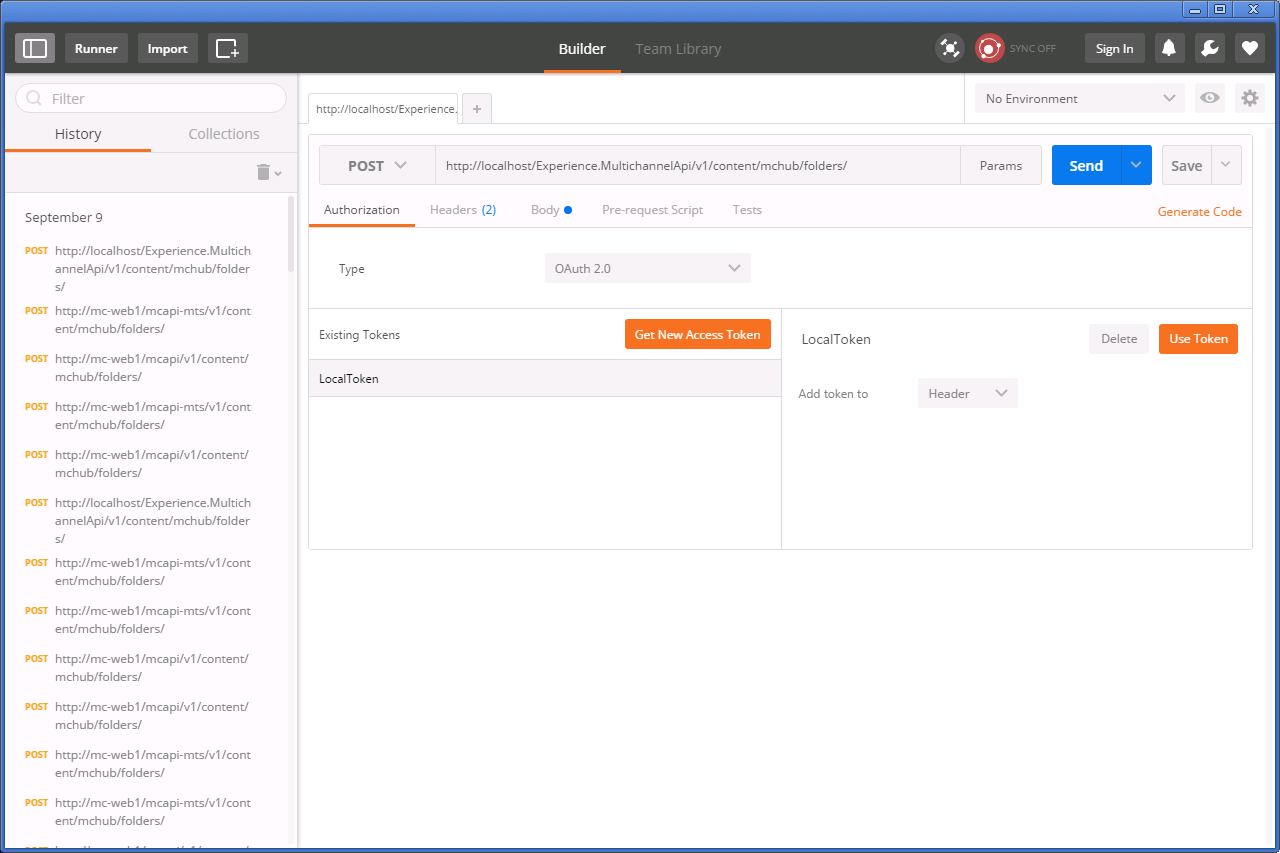
To receive token from Identity server, switch to Authorization tab, choose OAuth2.0 method and click on get new token. Enter data as shown on the image.



Login prompt will as you for credentials after which token will be received.

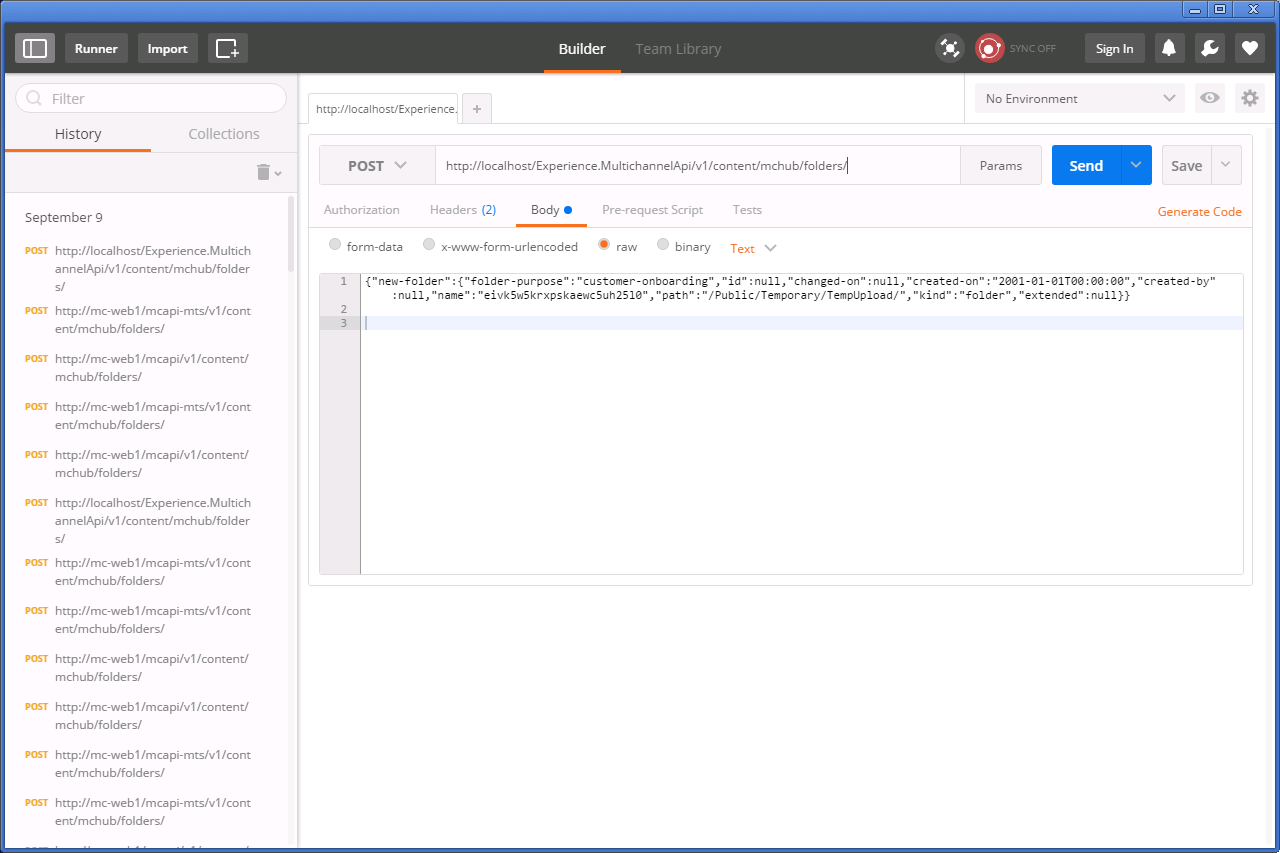


Choose to add token to header.



Along with Token (Authorization header), add header Content-Type with value “application/json”.

Now you can call api method you need to test. Following image shows posting JSON to URL:



Result will be displayed below in window below JSON payload within POSTMAN.

## DE Web site test

# Appendix 1: installations

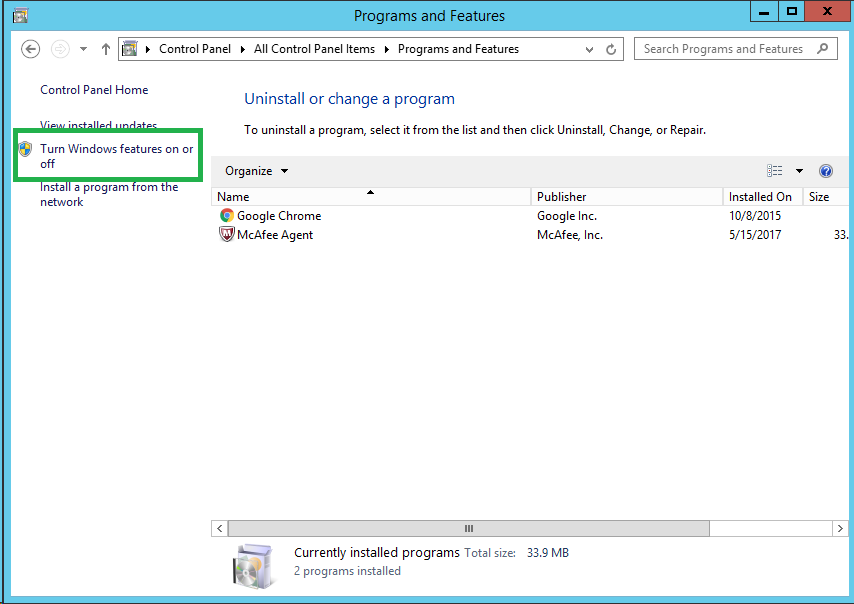
There are 4 logical server cluster required for DE environment. First one is plain web server having ARR installed with a purpose of being web gate and web cache. Single server is required, but business availability scenarios may require 2 hosts (nodes). This server cluster may be avoided if there is rooter with similar capabilities (load balancing and static content and images web caching).

Second cluster of servers is Web servers hosting the web application. It is recommended to have 2 nodes in cluster to help reduce downtimes during application maintenance and upgrade.

Third cluster is API web servers. Those servers handle API services as well as backend event driven processes, and hosts the infracture services such as AppFabric caching service and Service Bus.

## Windows application server roles and features installation

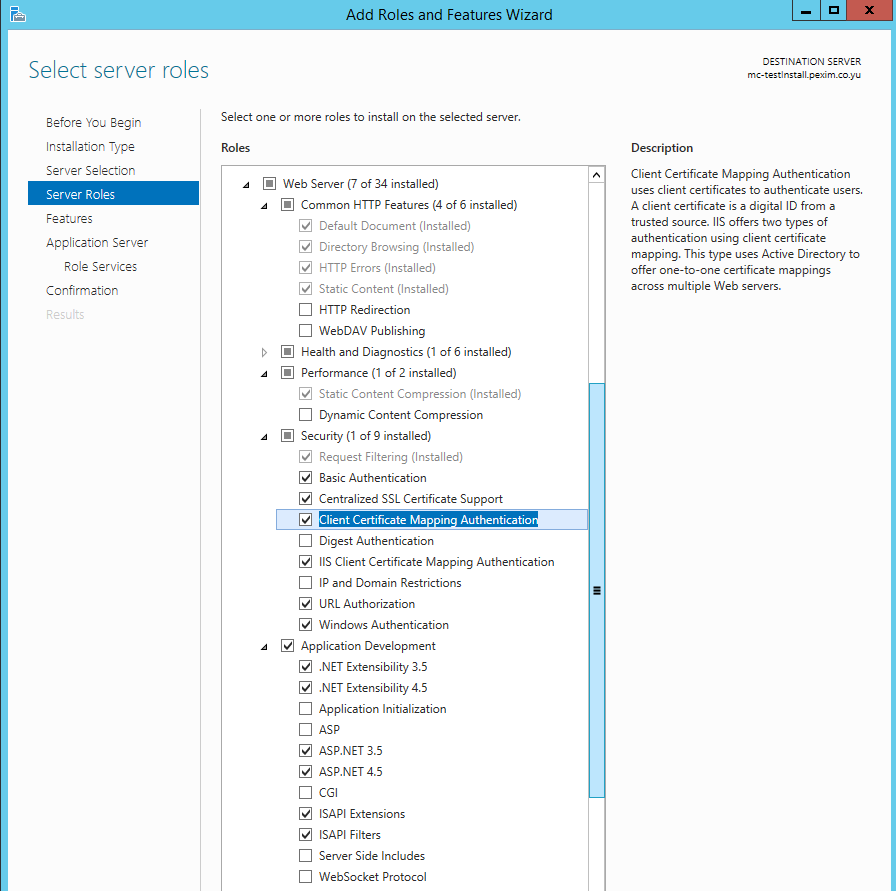
This step is required on application server only. From control panel choose programs and features, and from left pane choose “Turn windows features on or off”.



Installation wizard starts, first screen is informational. Click on Next to continue and on following screen choose “Role-based or Feature-based installation” and again click on Next. Following screen offers choice server on which installation will occur, please select current server and click on Next.

On server roles selection screen, please choose following roles (confirm additional features required for selection if necessary when prompted):

* Application server
* Web Server (IIS)
  + Common features
    - Default Document
    - Directory Browsing
    - HTTP errors
    - Static Content
  + Health and diagnostics (Http logging only)
  + Performance (Static Content Compression only)
  + Security
    - Basic
    - Centralized SSL certificate Support
    - Client Certificate Mapping Authentication
    - IIS Client Certificate Mapping Authentication
    - URL Authorization
    - Windows Authentication
  + Application development (select all .NET options)
    - ASP.NET 3.5 (this will automatically include required .NET Framework 3.5 feature along with .NET 3.5 Extensibility and ISAPI filters and extensions)
    - ASP.NET 4.5



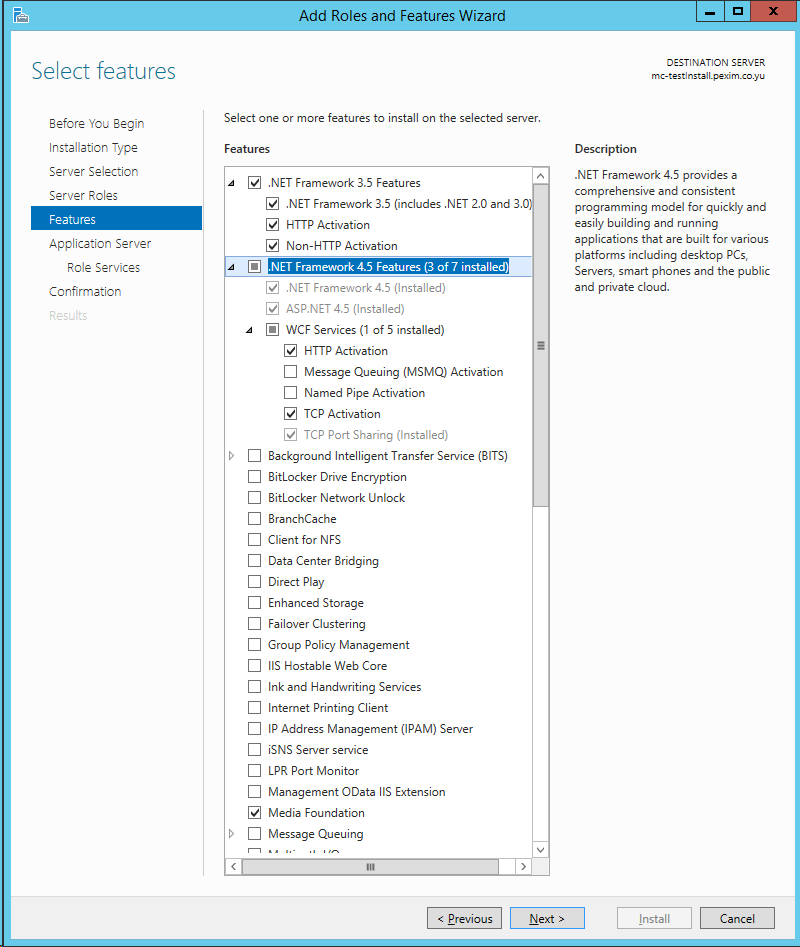
Click on Next to continue.

On features selection screen some of features may have been added during previous selection of roles, however those needs to be reviewed.

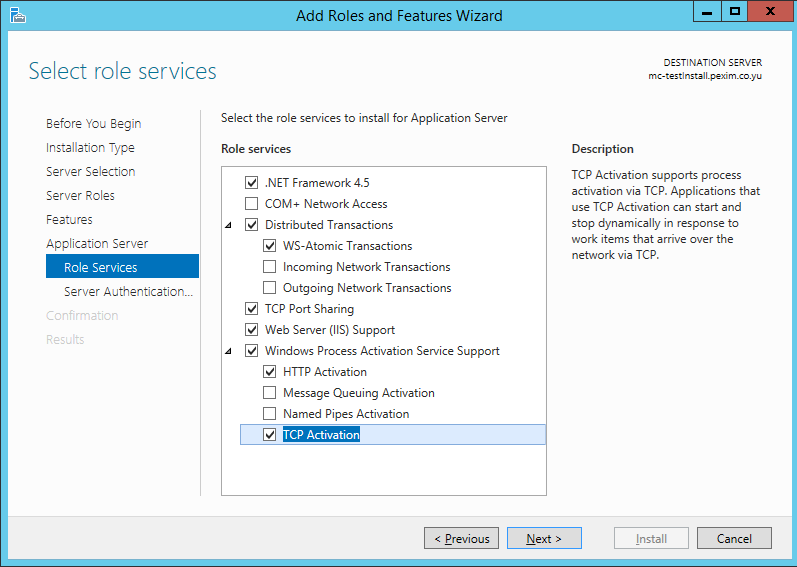
Required will be noted, and if prompted that additional features are required, please enable those as well.

* >NET Framework 3.5 (Enable all sub settings & confirm all prompted features)
* .NET Framework 4.5
  + .NET Framework 4.5
  + ASP.NET 4.5
  + WCF Services
    - HTTP Activation
    - TCP activation
    - TCP Port Sharing
* Media foundation (Required for chat-bot only on WebApp servers, not on API servers if multiple servers are used)
* Network Load Balancing (NLB - Required on API servers if multiple servers are used, will prompt for Remote Server Administration Tools/Network Load Balancing Tools feature)

Please note that setting above may have enable additional features, and confirm them when prompted. Example screenshot of selected features is presented below.



Clicking on Next role services needs to be selected for application server. Once again, when propted confirm additional related features for each service requesting them. Check following image for proper services selection:



Click on Next will prompt for certificate for SSL Encryption, please choose “Choose a certificate for SSL encryption later” option and click on Next for final installation confirmation and proceed to end of wizard by clicking on Install.

### Network load balancing service

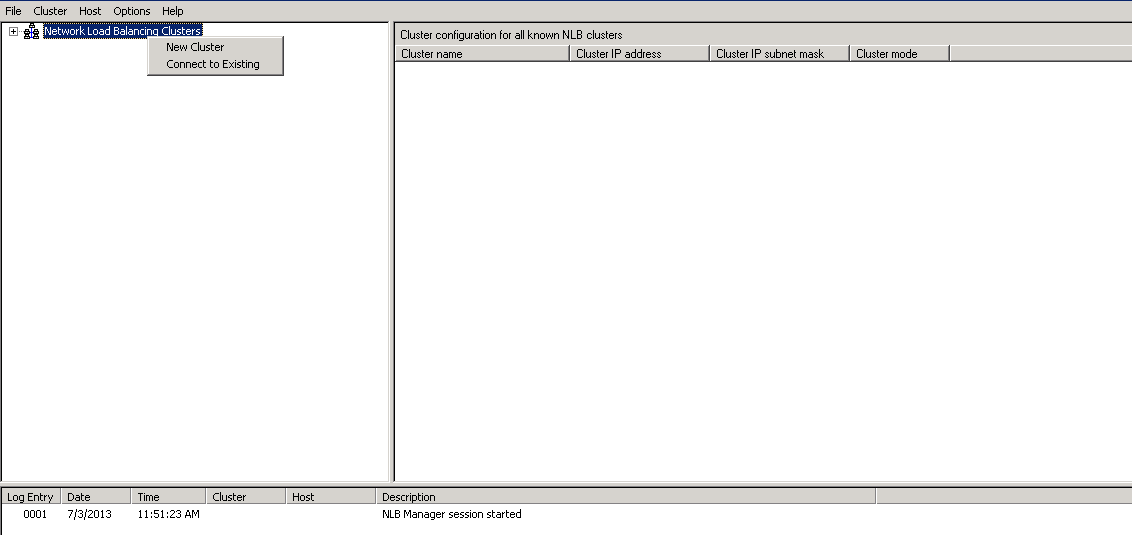
To scale performance and set environment with multiple serviers in cluster which raises avilability and minimizes downtime during application update, network loading balance service (NLB) must be set. Do not take values from screenshots as they may not be suited for current installation.

Prior to NLB setup you have to have static IP addresses without DHCP (address assigning) for all computers participating in the cluster, and one static address for NLB cluster. Make sure that all static IPs used in NLB scenarios are excluded from DHCP to prevent other machines receiving those address while cluster nodes are restarting or down for maintenance.

#### Adding New Cluster

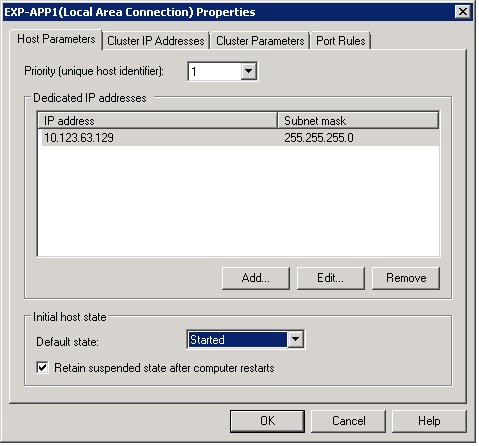
Search for “Nlb” on windows start menu and start “Network Load Balancing Manager”.

On first server which will become a node in NLB cluster, right-click on ***Network Load Balancing Clusters > New Cluster***

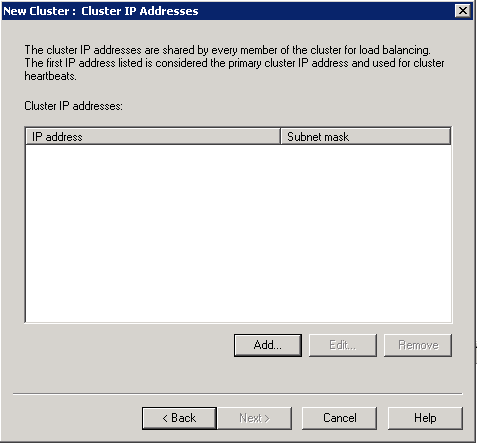


In the field Host, add address, then click on the **[Connect]** button, then **[Next]**

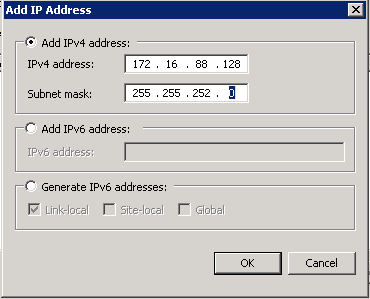
Set default state to ***Started***, and check option ***‘Retain suspended state after computer restarts’***. Click on **[Next]**



On the next screen, click **[Add]**

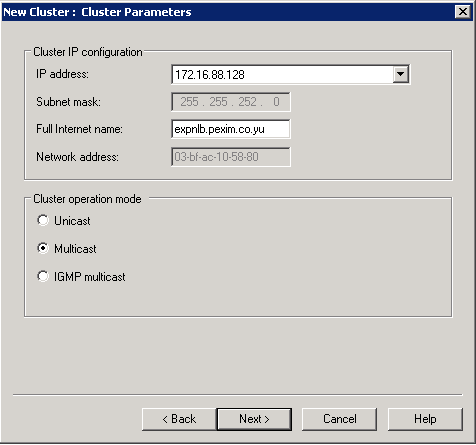


On the next screen, set IPv4 address for the cluster and subnet mask:



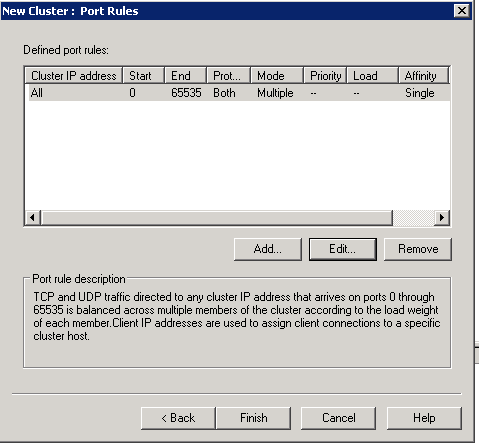
Click **[OK]**, then **[Next]**.

On the next screen, add **Full internet name** and set **Cluster operation** mode to ***Multicast***:



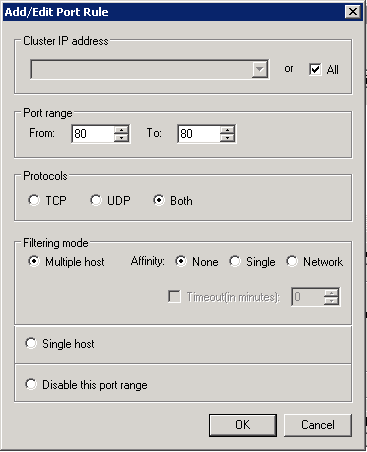
If server has 2 hardware network cards, Unicast can be used, however it will fully converge adapter to cluster and this adapter/IP will be unavailable for other processes. Click **[Next]**.

On the next screen, click **[Edit]**



By default all ports will be shared on cluster, however we recommend only ports in use should be shared.

Set **Port range** and **Filtering mode** (all ports for cache):



*Note: Every server from cluster has its own IP address, this one and static as well. If* ***multiple host*** *is set, then it sees both addresses. If* ***Affinity*** *is set to* ***None****, job will be taken over by available server, and if set to* ***Single****, then one from cluster will take the job. It should be set as None for API functionalities.*Click **[OK]**, then **[Add...]**

Repeat this step until all of following ports are added:

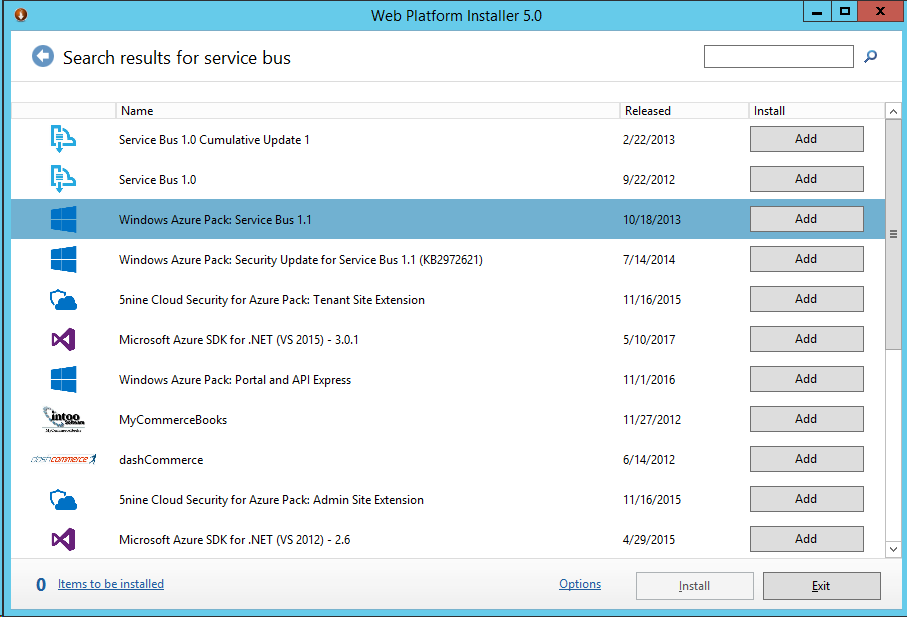
* 80 http
* 808 net.tcp
* 443 https
* 9354 Service Bus
* 9355 Service Bus
* 22233 AppFabric cache
* 22234 AppFabric cache
* 22235 AppFabric cache
* 22236 AppFabric cache

#### Connect to existing cluster

On other servers, start NLB manager as described [above](#_Adding_New_Cluster), and choose “Connect to Existing” option from “Cluster” menu. After converging is done, all settings from cluster will be the same of all nodes participating.

## Web Platform Installer 5

Download Web platform installer (WPI) launcher from <https://www.microsoft.com/web/downloads/platform.aspx> and start the downloaded executable which will trigger automatic download and install of WPI. Below image indicates example of searching for products installations using WPI.

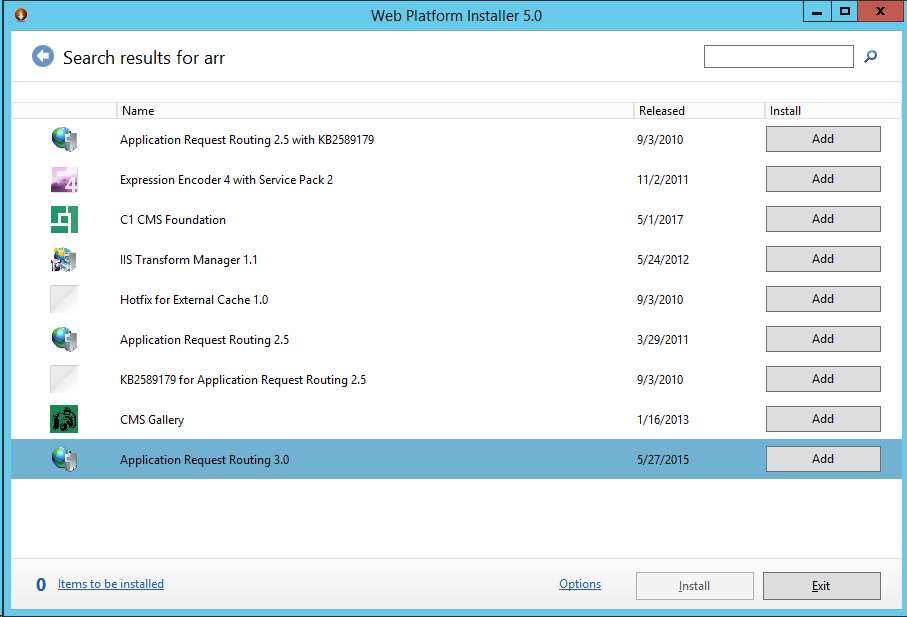


## ARR (Application Request Routing module)

This module is required only on web gateway server.

### Install ARR

Using Web platform installer (WPI) search for “arr” which stands for Advanced Routing and Rewrite module. Select version 3.0 as shown on below image.

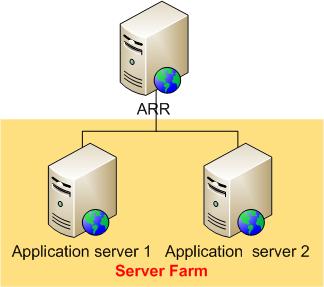


Click on “Add” and then on “Install”. Accept the license terms so installation process can start.

After installation succeeds, confirmation dialog is shown.

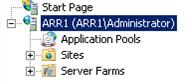
### Configure ARR

To define and configure a server farm in Application Request Routing Version 1 on IIS as shown below:

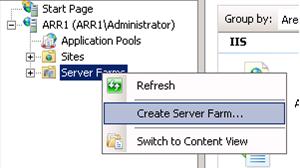
[](https://docs.microsoft.com/en-us/iis/extensions/configuring-application-request-routing-arr/define-and-configure-an-application-request-routing-server-farm/_static/image4.jpg)

On above image ARR is installed on web gateway server, and referred as ARR1 in later documentation, which may not be required if there is hardware solution which can support all ARR features.

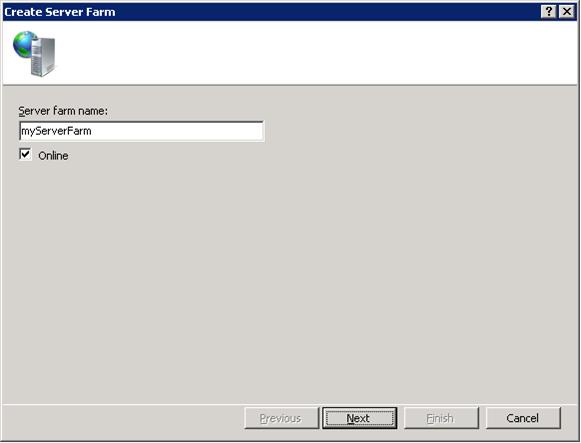
Search for “IIS” in Windows start menu and launch IIS Manager. Find Application Request Routing (ARR) in a server level feature. Select and expand the root of the server (ARR1 is just name used on the image, in fact this should be DE gateway server or server cluster, depending on desired level of scalability).

[](https://docs.microsoft.com/en-us/iis/extensions/configuring-application-request-routing-arr/define-and-configure-an-application-request-routing-server-farm/_static/image6.jpg)

To create a server farm, right-click **Server Farms**, and then select **Create Server Farm...**.

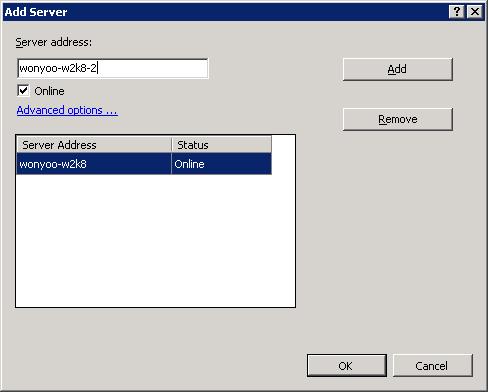
[](https://docs.microsoft.com/en-us/iis/extensions/configuring-application-request-routing-arr/define-and-configure-an-application-request-routing-server-farm/_static/image8.jpg)

Enter a name for the server farm. In the example below, **myServerFarm** is the name of the server farm. For DE purposes 2 server farms are required, DEWEB and DEAPI. Click **Next** to continue.

[](https://docs.microsoft.com/en-us/iis/extensions/configuring-application-request-routing-arr/define-and-configure-an-application-request-routing-server-farm/_static/image10.jpg)

The next step is to add servers to a server farm.

On the Add Server page of the wizard, add as many application servers as needed. For DEWEB server group add web servers which will host DEWeb application, and for DEAPI server group add all servers which are intended to host DE WebAPI, and which are previously added to NLB API cluster. Click **Finish** to create the server farm with the entered application servers as the server farm members. Process needs to be repeated for each group by adding required servers to it.

*Advanced options...* allow you to configure the HTTP and HTTPS ports to non-standard (80 for HTTP and 443 for HTTPS) ports: [](https://docs.microsoft.com/en-us/iis/extensions/configuring-application-request-routing-arr/define-and-configure-an-application-request-routing-server-farm/_static/image12.jpg)

## Service Bus 1.1

### Service Bus 1.1 Installation

--edit

Offline installation:

Use bat in sbInstall.zip

C:\Tools\WebPIOfflineFiles

From command prompt navigate to this folder on target computer and type:

bin\webpicmd /install /Products:ServiceBus\_1\_1 /xml:feeds\latest\webproductlist.xml

--edit

Service bus installation can be installed using downloaded installation or by using Web platform installer (WPI). Here we will describe process of installation and configuration of Service Bus 1.1 service.

Service Bus is enterprise bus service manly used by event driven systems to decouple functionalities and do distribute the processing.

Service Bus is installed on API web servers cluster (cluster can have at least one host or may be scaled to many hosts if business process requires heavy load & processing).

Start WPI and search for “Service Bus 1.1” product. Select Windows Azure Pack: Service Bus 1.1 and add it to install selection, choose Install and accept the license terms to continue. When prompted always make sure that you use Microsoft update when checking for updates.

After Service Bus 1.1 files are downloaded and installed, confirmation window is shown.

### Service Bus 1.1. Configuration

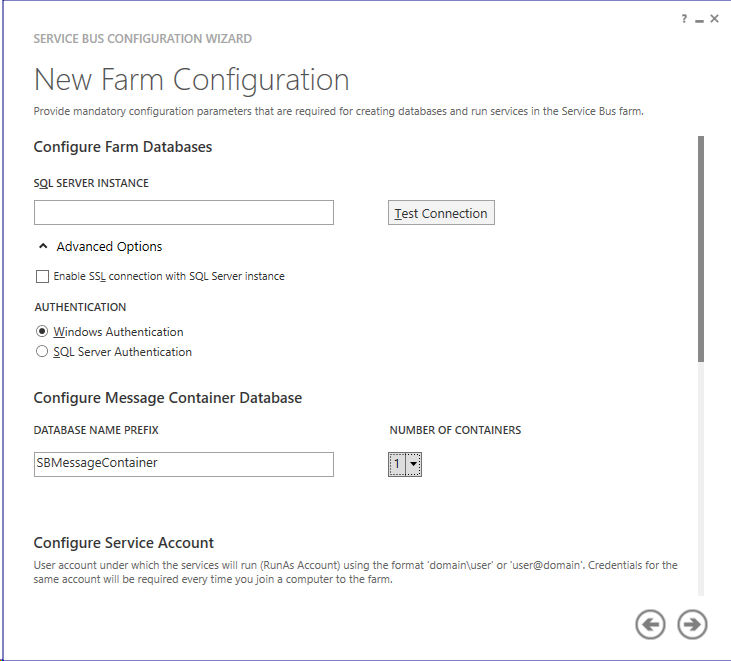
Press Windows key and type “service” in Start menu to find and start Service Bus Configuration.

From this point on forward, it is required to distinct hosts within cluster if there is any.

When configuring service bus for the first time on API web servers cluster, Service Bus cluster does not exist, and within Service Bus Configuration Wizard it is required to “Create New Farm”. If installation is performed on second host (node) of API web server cluster, it is required to choose “Join an existing farm”.

#### Create New Farm

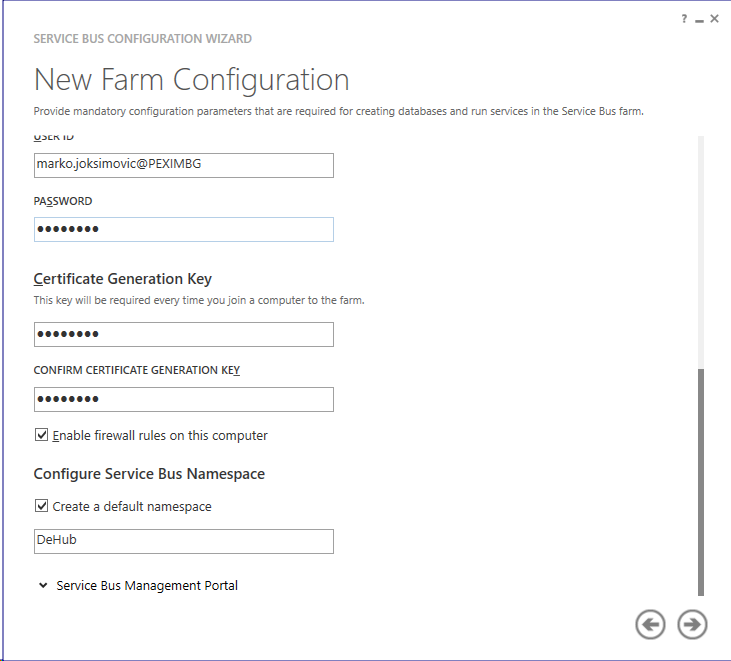
Click on “Using Default Settings (recommended)” option. On following screen choose SQL Server instance (SQL Server needs to be installed prior to Service Bus Installation) which will host Service Bus databases, select 1 container (1 + 2 Databases will be created). Configure windows account which will have sufficient user rights on selected SQL Server instance. It is possible to configure SQL Server SA authentication for database access using advanced options, however SQL Server Authentication is not recommended for production environments.



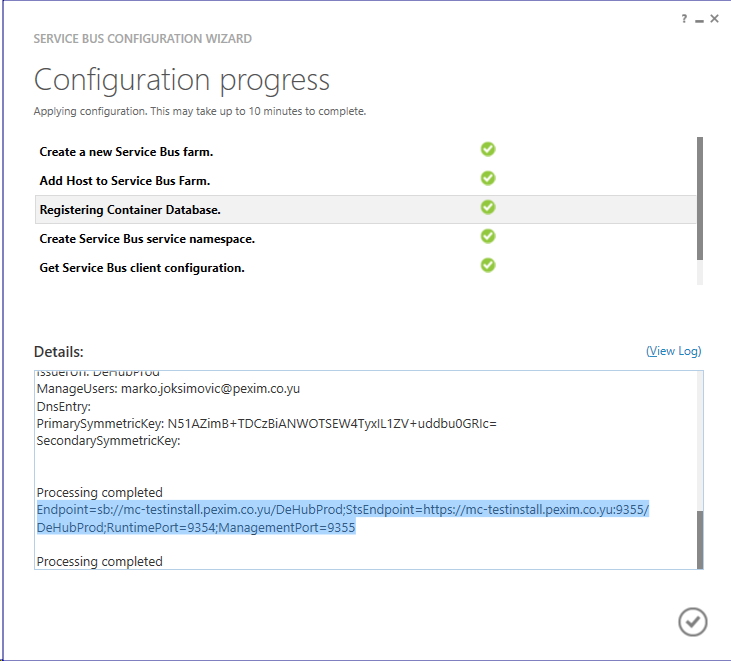
Service account is account under credentials Service Bus service will run. This account needs to have sufficient privileges on host system to be able to run as service, access to network, manage SSL Certificate stores. Reconsider creating/using special service accounts which are not covered with password change policy, as policy may require periodical password change, or even password lock which can introduce unplanned service downtime.

Certificate generation key(Welcome1)is important then cluster is managed, so make sure that this key is available to future administrators of Service Bus.

Last setting is related to a Namespace which will be used for DE. By default, “ServiceBusDefaulltNameSpace” is presented as default choice, but it can be changed to simpler, more desirable name such as “DeHubNameSpace”. Please note that this name is used within API and DE Services configuration, so make sure that chosen namespace is applied throughout DE services.



Confirmation screen is displayed after successful installation and setup will advance to creation of new farm, after which configuration progress will be shown for final review.

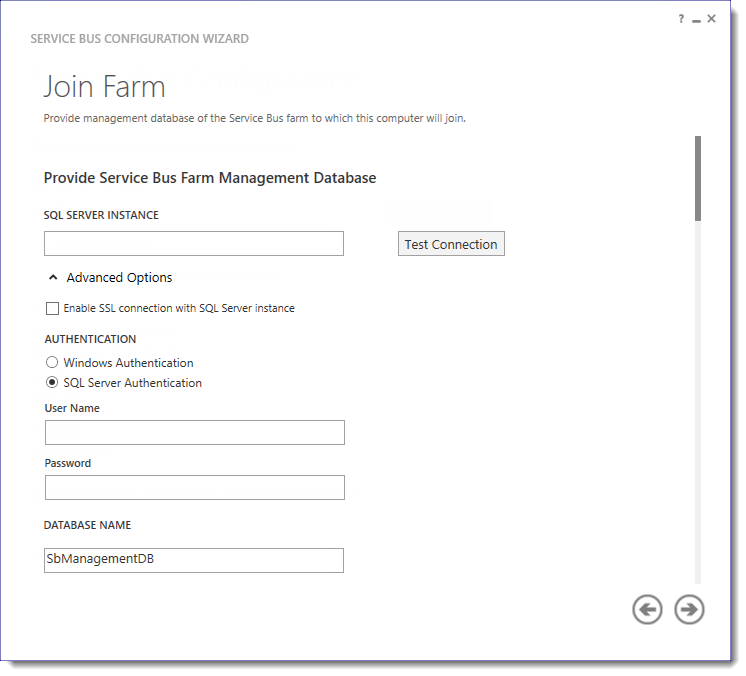


Please note that required Service Bus connection string for Service Buss access is highlighted in above screenshot.

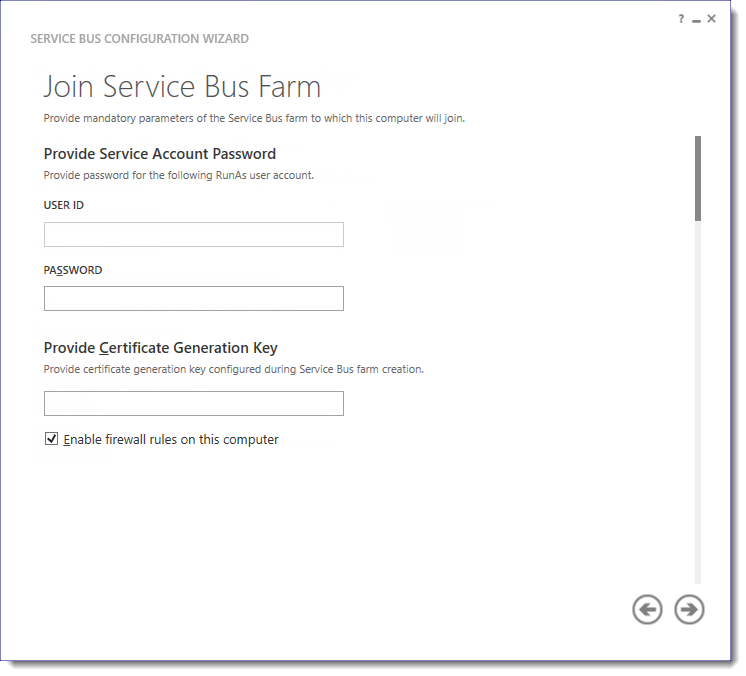
#### Join an Existing Farm

If there are more than one server node in API servers cluster, it is required to join server to Service Bus farm installed on first node.

Now it is necessary to enter SQL Server instance and configuration database used for creating the cluster in [Create New Farm](#_Create_New_Farm) step.



And to provide user account for Service Bus service and Certificate generated Key on next step.



And to advance the wizard until confirmation of successful joining to Service Bus cluster is received.

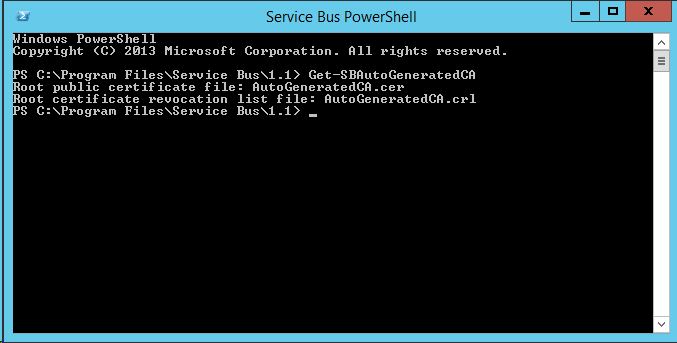
Repeat this step on all server nodes in the API cluster.

#### Client Certificate and user access

To access Service Bus, it is required to have client certificate installed. Service Bus creates auto generated certificate which can be downloaded using Service Bus Power Shell Console.

To Export a client certificate, type “Service” in Windows Start Menu Search, and run “Service Bus PowerShell” with administrative permissions

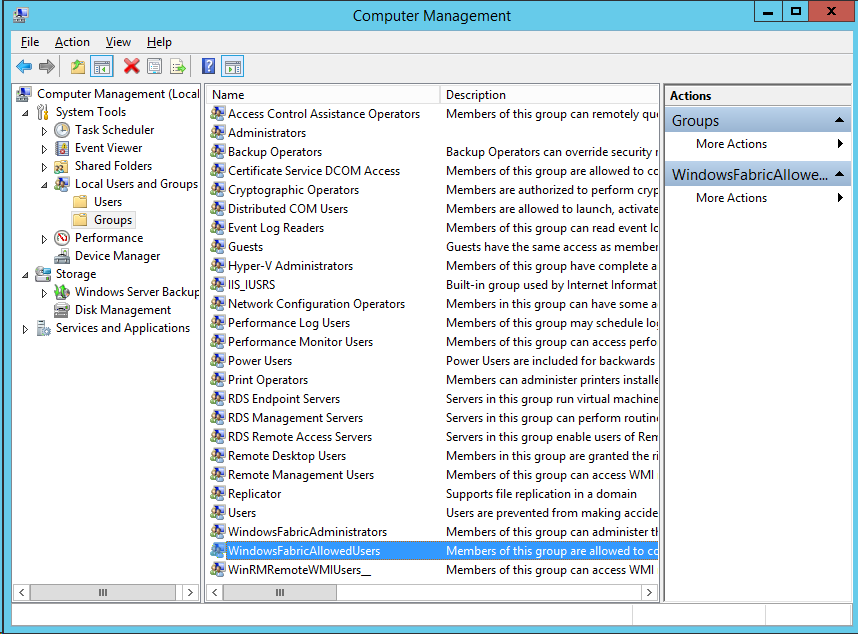
Type Get-SBAutoGeneratedCA which will export certificates to current power shell folder (if command above is copied from rich text editor like Word, make sure it does not have special formatting characters which can prevent execution in power shell).



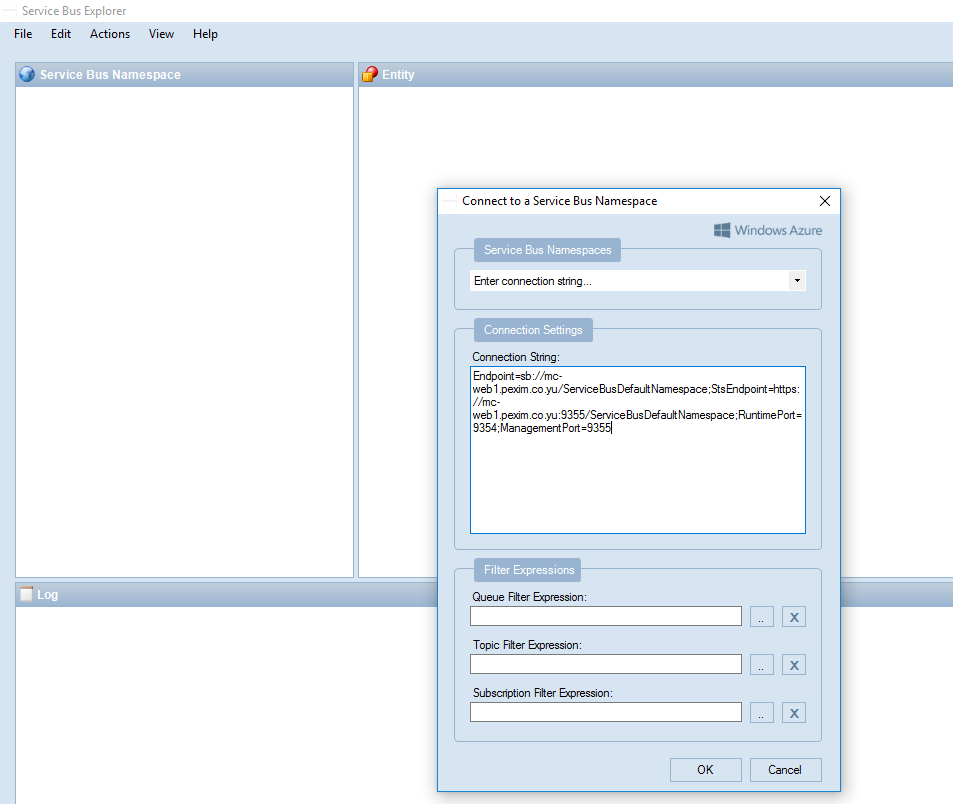
Install those certificates on all machines to be used as Service Bus clients. In standard case, this certificate needs to be installed on all API servers as services working with Service Bus are installed there only.

If there is third party integration with Service Bus, this certificate is also required on all client machines.

To allow additional users to access Service Bus, please manage users in WindowsFabricAdministrators and WindowsFabricAllowedUsers groups, as shown on below image.



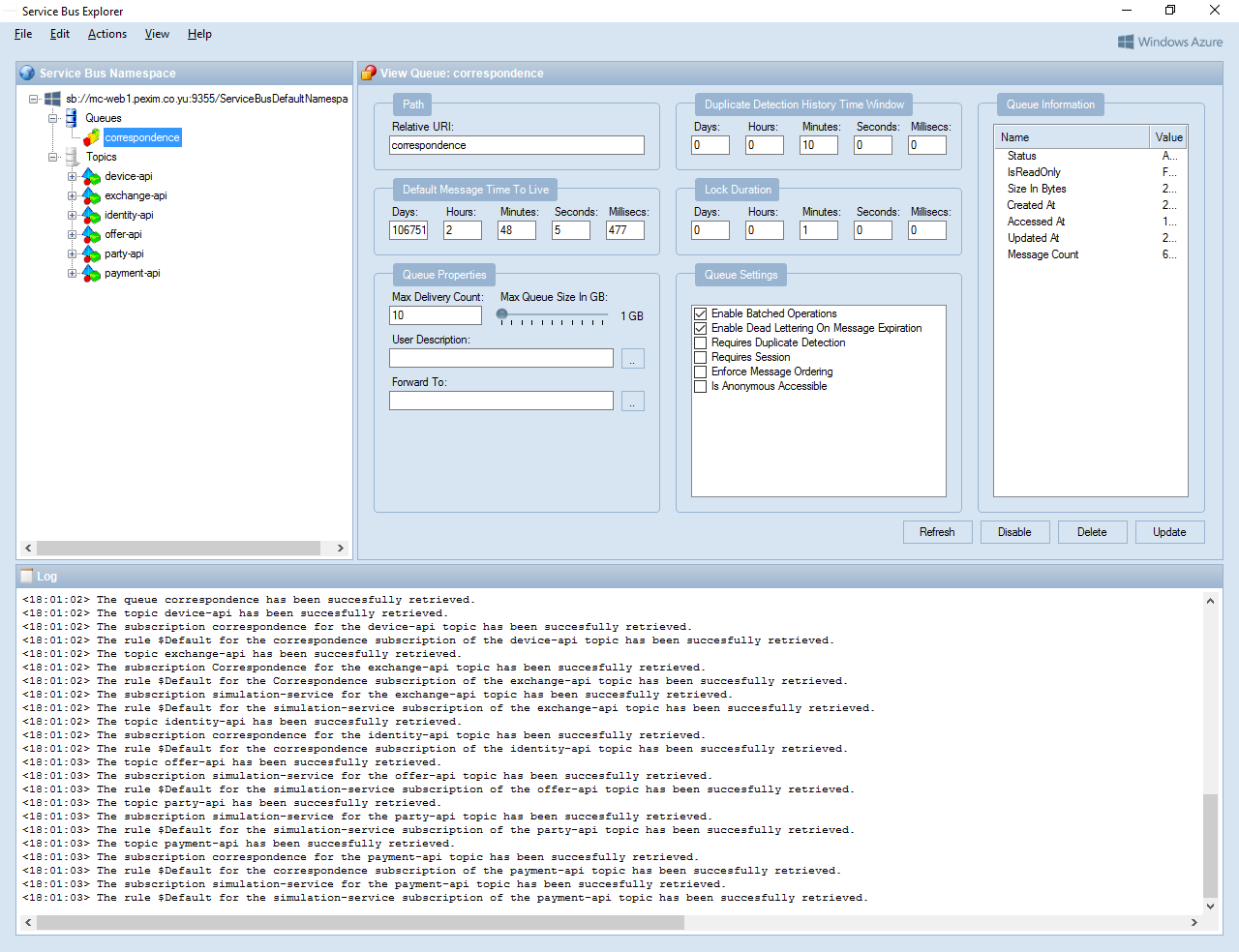
After configuration, Service Bus Explorer (third party app) can be used to verify Service Bus availability and access. Start the application and select File/Connect from menu and enter [connection string highlighted in step of installation and farm setup](#_Create_New_Farm) as connection string, like shown on below image.



If errors regarding security token are displayed, verify that client access certificate is added to Trusted Certificate Root store. Also, if unauthorized access error is shown, make sure that user trying to connect to Service Bus is added to appropriate groups as described in this section.

On successful connection, Service Bus topics and Queues are shown if any.

Please note that services will create required topics and queues for normal DE operation.



Above image shows how connected and set environment looks like, but server name and Service Bus namespace may differ from current installation.

## APP Fabric Installation

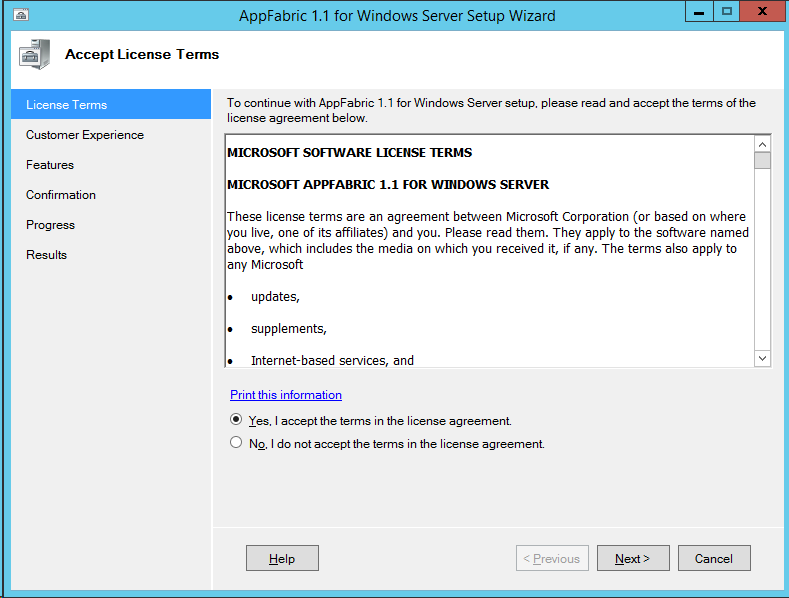
AppFabric service is usually installed on Application server cluster (often revert it as API wen servers, midleware).

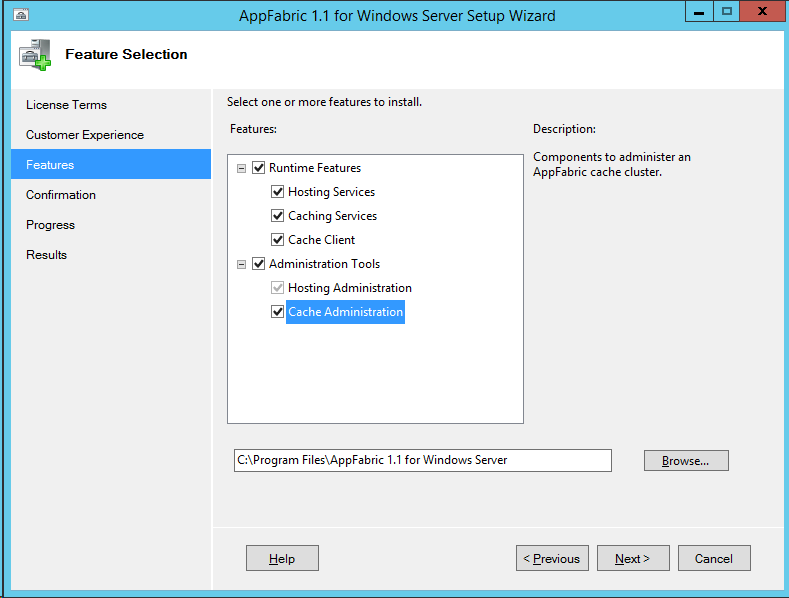
### AppFabric Installation

AppFabric Caching Service 1.1 installation is available through Web Platform Installer (WPI), however, this installation will try install SQL Server Express as well which is not required. It is better to download package from Internet. Search for AppFabric 1.1 for windows server, or try link below:

<https://www.microsoft.com/en-us/download/details.aspx?id=27115>

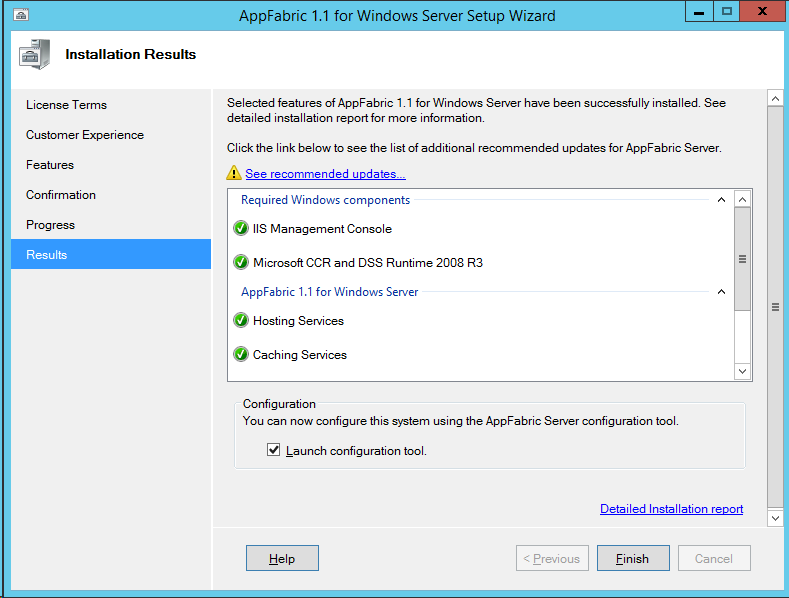
Make sure that x64 version is downloaded (WindowsServerAppFabricSetup\_x64.exe). AppFabric also uses windows update to retrieve latest cumulative updates (CUs), so please enable windows update on server.



Accept the licensing agreement to proceed, and on Feature selection screen, choose following (all) options:

1. Runtime Features
   1. Hosting Services
   2. Caching Services
   3. Cache Client
2. Administration Tool
   1. Hosting Administration
   2. Cache Administration

Finish installation by clicking on Next and Then on Install

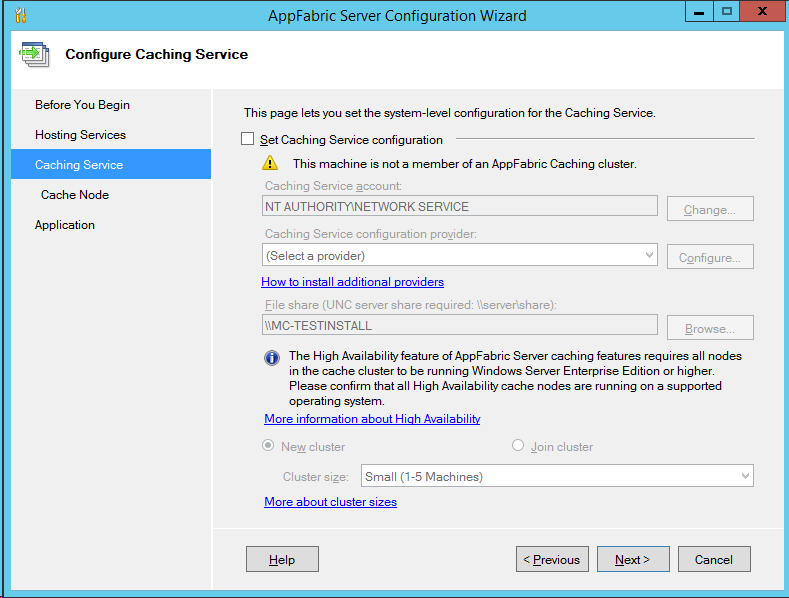


Leave “Launch configuration tool” option checked to proceed to configuration.

Configuration can be launched later by searching for “app” and launching Configure AppFabric.

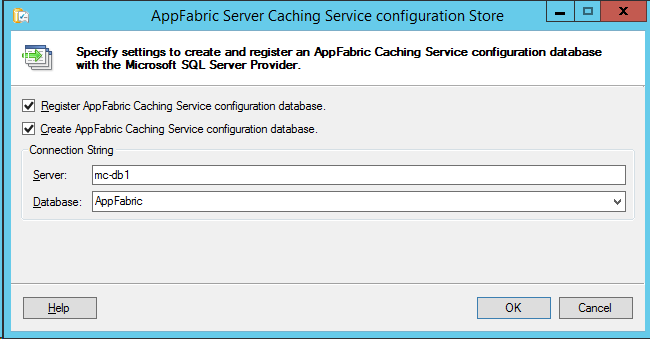
#### AppFabric Configuration

On AppFabric configuration wizard advance all through “Before You Begin” and “Configure Host Services” steps to “Configure Caching Service” step.



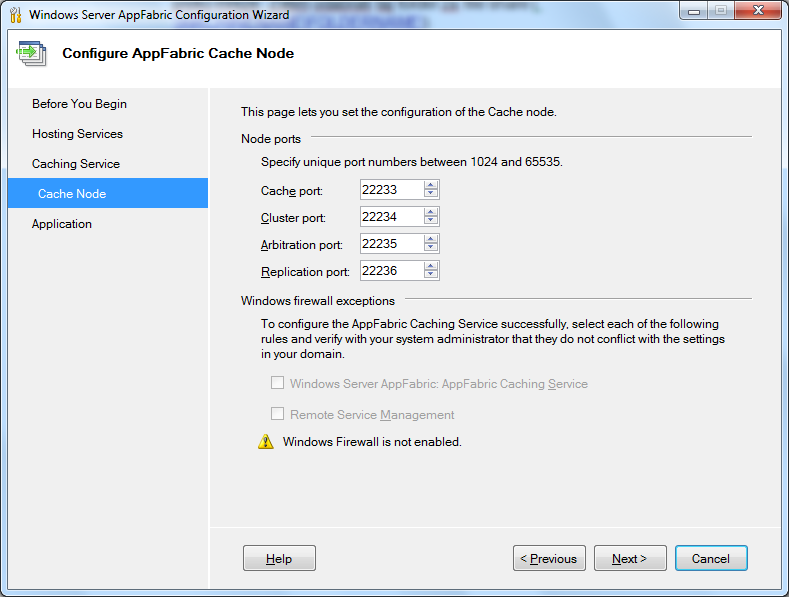
Check “Set Caching Service configurationConfigure” option, select account with sufficient user rights on target SQL Server as Caching Service Account, and set “Caching Service configuration provider” to “SQLServer AppFabric Caching Service Configuration Store Provider” to store AppFabric configuration in database. Click on “Configure…” to set provider details.

It is possible to store AppFabric model to existing or in new database. We recommend that AppFabric should have its own database called “AppFabric”. Check both options and enter Database server (and instance if exists) and database name (AppFabric) as shown on screen shot below.



Check to ***New cluster*** option and leave cluster size to “Small (1 - 5 Machines)”.

On Configure AppFabric Cache Node screen, leave settings as it is:

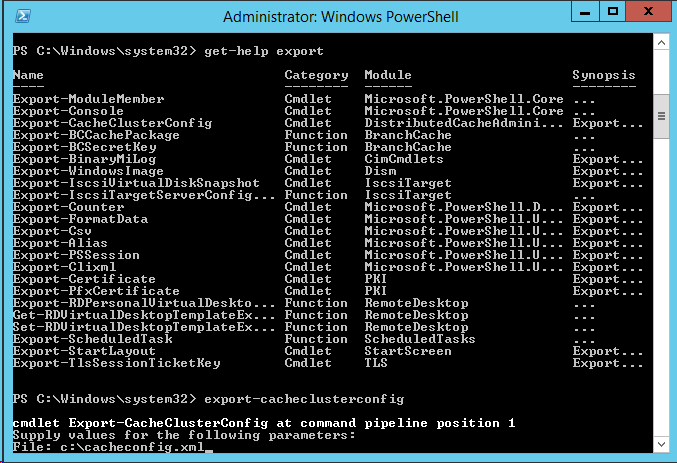


Make sure that these [settings are included](#_Network_load_balancing) in NLB cluster. Finish configuration.

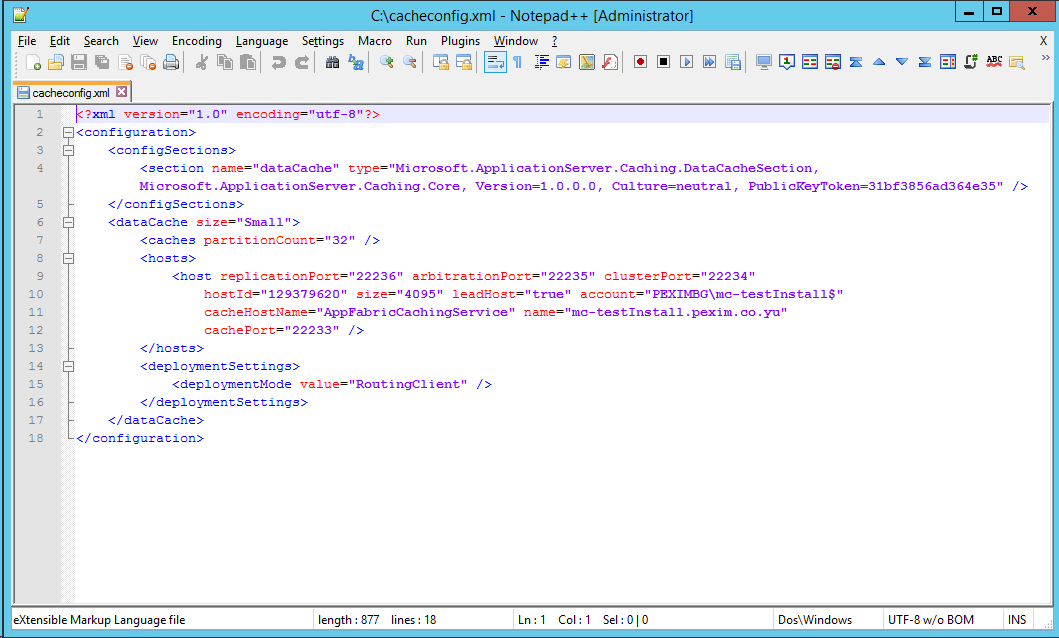
AppFabric is using “Caching Administration Windows PowerShell” console is used for managing configuration settings, however, for the first time it is simpler to export configuration, make changes and then import modified configuration back.

To do so, search for “cach” in Windows Start menu and launch “Caching Administration Windows PowerShell” with Administrator privileges. Type: Export-CacheClusterConfig (please note that copying may introduce formatting characters which can prevent command execution if pasted directly).

Enter name of file like shown on image below and press enter to perform export.



Depending on chosen path, open exported file in some Xml editor which will simplify navigation and editing (e.g. Notepad++).



Replace **caches** node with following caches deffinition:

<caches>

<cache consistency="StrongConsistency" name="session">

<policy>

<eviction type="None" />

<expiration defaultTTL="7200" isExpirable="false" />

<serverNotification isEnabled="true" />

</policy>

</cache>

<cache consistency="StrongConsistency" name="default">

<policy>

<eviction type="Lru" />

<expiration defaultTTL="7200" isExpirable="false" />

<serverNotification isEnabled="true" />

</policy>

</cache>

</caches>

After node “deploymentSettings”, at the same level add node advancedProperties:

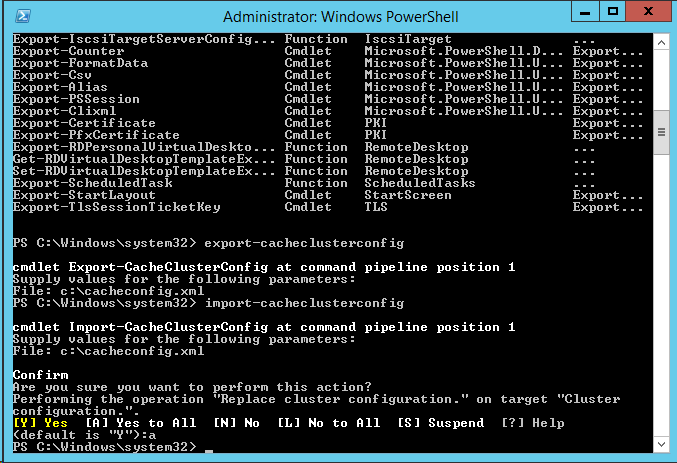
<advancedProperties>

<memoryPressureMonitor syncGCInterval="5" />

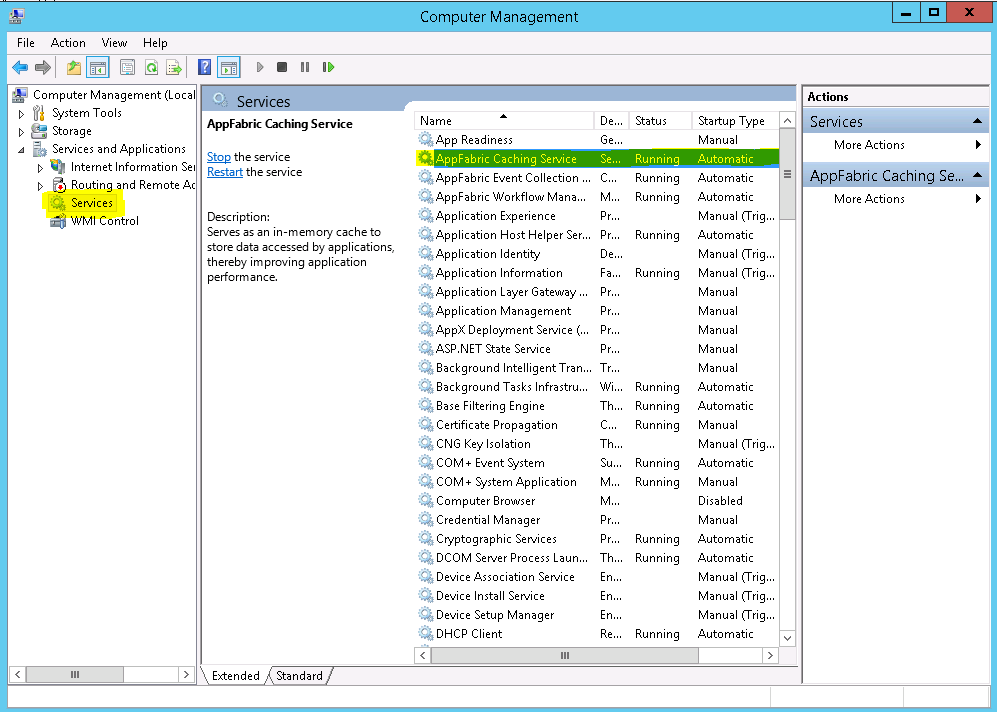
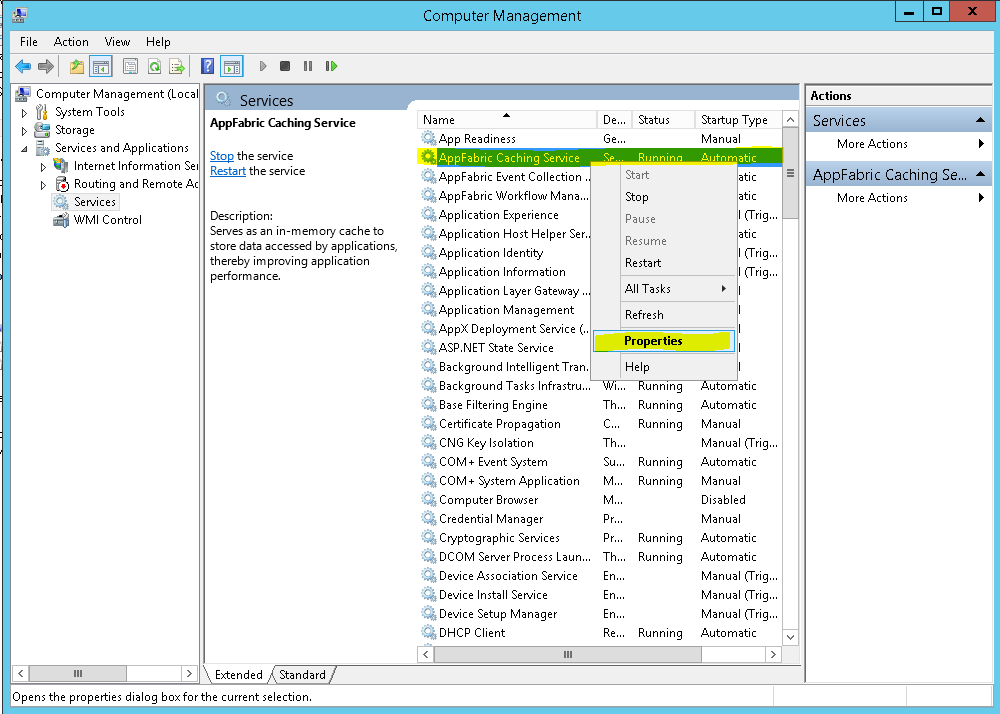
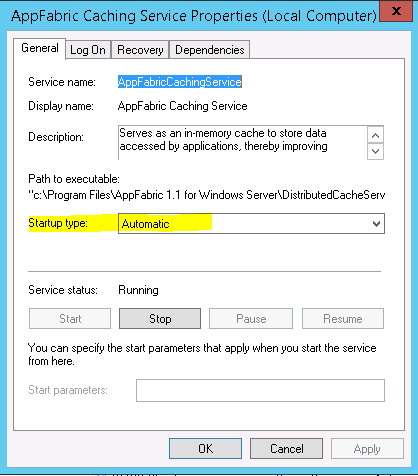
<securityProperties mode="None" protectionLevel="None" />

<transportProperties maxBufferSize="1000000000" receiveTimeout="86400000" />

</advancedProperties>

Save configuration file (as a new or overwrite existing) and return to AppFabric Caching PowerShell and type: Import-CacheClusterConfig followed by path and filename to previously modified configuration file. Finally press A or Y to perform actual import.

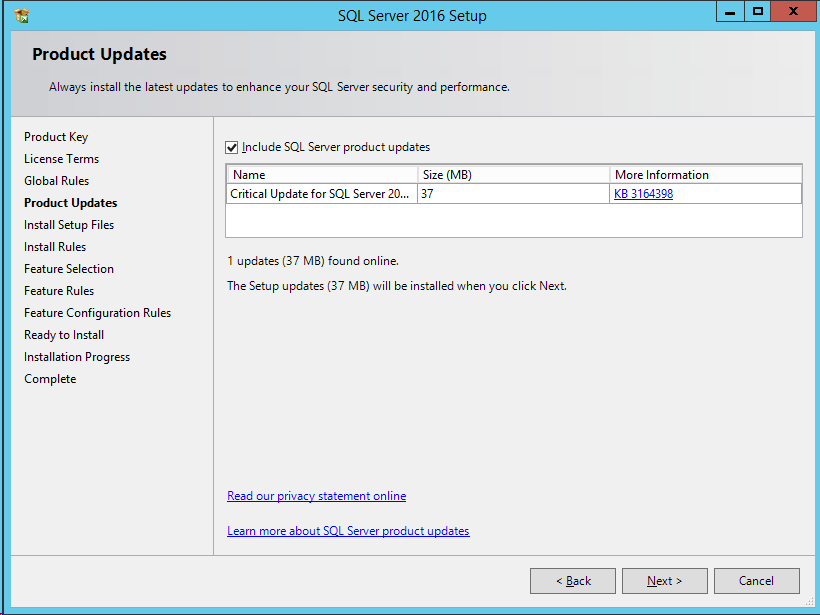
After installationIn order to ***cache*** work properly, few more parameters has to be set up in the environment. First step is to be done on the computer that has ***Cache*** installed, and second has to be done on the client computers.

1. On the computer that has ***Cache*** installed, set ***Cache*** service to start-up when computer is started-up:  
     
   Right-click on the service, then ***Properties***:  
     
   Choose Startup type = ’Automatic’:  
   

## SQL Server installation

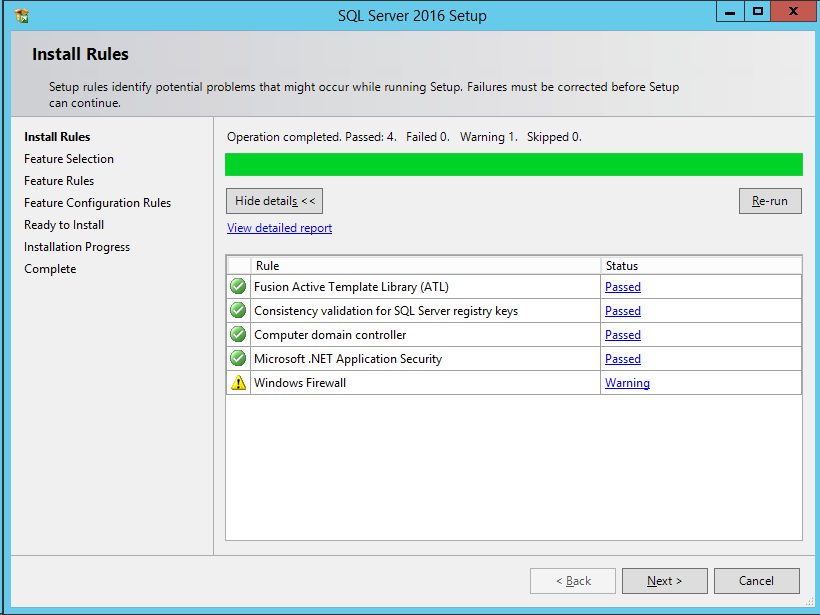
SQL Server 2016 Standard is recommended to be used in DE environment.

Start Installation process and on SQL Server Installation Center, choose installation on left pane, and then click New SQL Server stand-alone installation on the right pane. On new SQL Server 2016 Setup screen choose proper edition and enter product key for it and accept license agreement on next screen to proceed to actual installation.



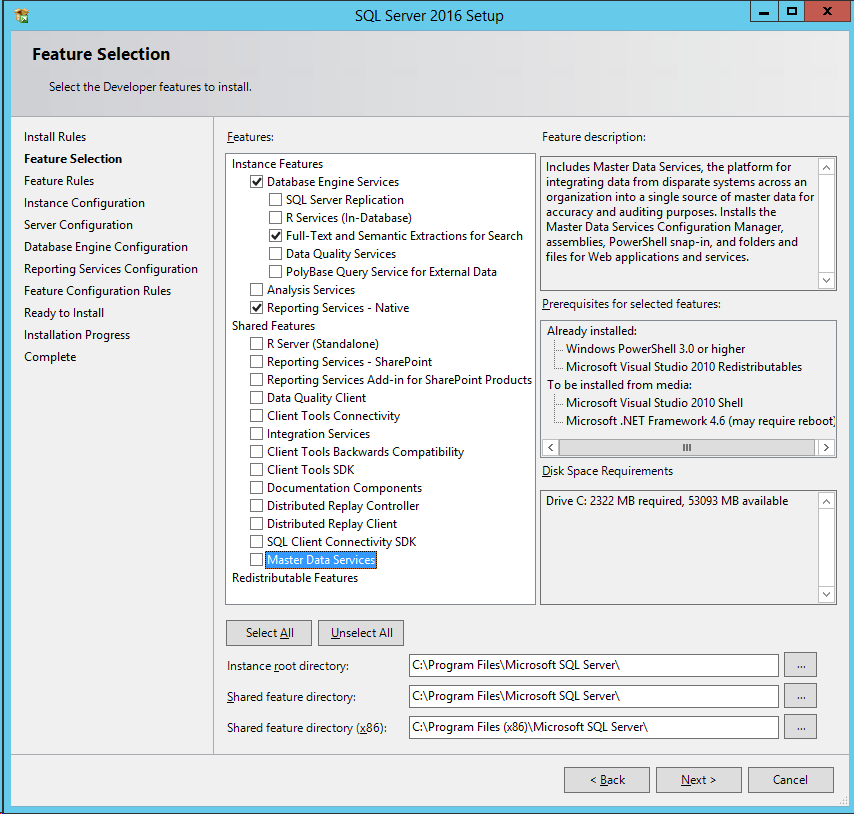
It is recommended to install all product updates including the Windows updates (KB2919355).

Install rules is next check point. Sometimes, after a lot of product updates are installed, some of them or previous installation may require restart. Warnings can be skipped.



On feature selection screen, choose following Instance Features:

* Database Engine Services
  + Full-Text and Semantic Extractions for Search
* Reporting Services – Native



Click on Next to continue.

## MongoDB installation