Orleans Configuration Guide

For the Orleans v0.8.0 Alpha release, April-2012

This Configuration Guide explains the key configuration parameters and how they should be used for several most typical usage scenarios.

[1 Introduction 1](#_Toc327169955)

[2 Server and Client 1](#_Toc327169956)

[3 Server (Silo) Configuration 2](#_Toc327169957)

[3.1 Connectivity 2](#_Toc327169958)

[3.2 Cluster Membership and Reliability 3](#_Toc327169959)

[3.3 Grain Index 3](#_Toc327169960)

[3.4 Primary Silo 4](#_Toc327169961)

[4 Client Configuration 4](#_Toc327169962)

[4.1 Fixed Gateway Configuration 4](#_Toc327169963)

[4.2 Gateway Configuration Based on Cluster Membership 5](#_Toc327169964)

[4.3 Local Silo 5](#_Toc327169965)

[5 Typical Configurations 5](#_Toc327169966)

[5.1 Local Development 5](#_Toc327169967)

[5.2 Reliable Production Deployment 6](#_Toc327169968)

[5.3 Reliable Production Deployment with Only Self-Managed Grains 6](#_Toc327169969)

[5.4 Unreliable Deployment on a Cluster of Dedicated Servers 6](#_Toc327169970)

# Introduction

Orleans can be used in a variety of configurations that fit different usage scenarios, such as local single node deployment for development and testing, cluster of servers, multi-instance Azure worker role, etc. All the different target scenarios are achieved by specifying particular values in the Orleans configuration XML files. This guide provides instructions for the key configurations parameters that are necessary to make Orleans run in one of the target scenarios. There are also other configuration parameters that primarily help fine tune Orleans for better performance. They are documented in the XSD schema and in general are not required even for running the system in production.

# Server and Client

Orleans is a framework for building and running high scale services. A typical deployment of an Orleans application spans a cluster of servers. The instances of the Orleans runtime, called silos, running on each of the servers need to be configured to connect to each other. In addition to that, there is always a client component that connects to the Orleans deployment, most typically a web frontend, that needs to be configured to connect to the silos. The Server Configuration and Client Configuration sections of the guide cover those aspects respectively.

# Server (Silo) Configuration

There are four key aspects of silo configuration:

* Connectivity: silo’s endpoints for other silos and clients
* Cluster membership and reliability: how silos discover each other in a deployment and detect node failures
* Grain index: where the system maintains the index of created Orleans-managed grains
* Persistence: how persistent state of grains is stored

Depending on the environment you want to run Orleans in some of these parameters may or may not be important. For example, for a single silo development environment one usually doesn’t need reliability, and all the endpoints can be localhost. Or if you only use self-managed grains in your application, you don’t need to configure grain index.

The following sections detail the configuration setting for the four mentioned key aspects. Then in the scenarios section you can find the recommended combinations of the settings for the most typical deployment scenarios.

## Connectivity

The connectivity settings define two TCP/IP endpoints: one for inter-silo communication and one for client connections, also referred to as client gateway or simply gateway.

### Inter-silo Endpoint

<Networking Address=" " Port="11111" />

Address: IP address or host name to use. If left empty, silo will pick the first available IPv4 address.

Port: TCP port to use. If left empty, silo will pick a random available port. If there is only one silo running on the machine, it is advisable to specify a port for consistency and for easy configuration of the firewall. For running multiple silos on the same machine, you can either provide each of the silos with different configuration files or leave the Port attribute empty for a random port assignment.

For machines that have more than one IP address assigned to them, if you need to choose an address from a specific subnet or an IPv6 address, you can do that by adding a Subnet and PreferredFamily attributes respectively (refer to the XSD schema for exact syntax of those attributes).

For local development environment, you can simply use localhost as the host name:

<Networking Address="localhost" Port="11111" />

### Client Gateway Endpoint

The setting for client gateway endpoint is identical to the inter-silo endpoint except for the XML element name:

<ProxyingGateway Address="localhost" Port="30000" />

You have to specify a port number different from the one used for the inter-silo endpoint.

It is possible to configure clients to connect to the inter-silo endpoint instead of the gateway, but that requires opening a listening socket on the client (thus requires enabling incoming connections on the client machine firewall), and in general is not advisable other than for a very limited set of scenarios.

## Cluster Membership and Reliability

Usually, a service built on Orleans is deployed on a cluster of nodes, either on dedicated hardware or in Azure. For development and basic testing, Orleans can be deployed in a single node configuration. When deployed to a cluster of nodes, Orleans internally implements a set of protocols to discover and maintain membership of Orleans silos in the cluster, including detection of node failures and automatic reconfiguration.

For reliable management of cluster membership, Orleans uses the Azure Table storage for synchronization of nodes. The reliable membership setup requires two settings in the silo configuration file: Liveness and Azure nodes.

<Liveness LivenessType ="AzureTable" />

<Azure DeploymentId="<your deployment ID>" DataConnectionString="DefaultEndpointsProtocol=https;AccountName=<Azure storage account>;AccountKey=<Azure table storage account key>"/>

AccountName and AccountKey parts of the DataConnectioString setting are the storage account and key for your Azure Table store. DeploymentId is a unique string that defines a particular deployment. When deploying an Orleans based service to Azure it makes most sense to use the Azure deployment ID of the worker role.

For development or if it’s not possible to use Azure Table, silos can be configured to use the membership grain instead. Such a configuration is unreliable as it will not survive a failure of the primary silo that hosts the membership grain. “MembershipTableGrain” is the default value of LivenessType.

<Liveness LivenessType ="MembershipTableGrain" />

## Grain Index

Orleans runtime maintains an index of all Orleans-managed grains (those that are not self-managed). The index contains a list of all Orleans-managed grains created in the system and their queryable properties. For a reliable deployment, grain index needs to be configured to use a SQL database, SQL Server or SQL Azure.

<Index ConnectionString=" Data Source=<db>.database.windows.net;Initial Catalog=Orleans;User=your\_user@<db>.database.windows.net;Password=<password> "/>

For development or if it’s not possible to use SQL, silos can be configured to use the domain index grain instead. Such a configuration is unreliable as it will not survive a failure of the primary silo that hosts the index grain.

<Index ConnectionString="DomainIndexGrain"/>

If your service only uses self-managed grains, then you don’t need a grain index and can set Index to “None”. “None” is the default value for Index.

<Index ConnectionString="None"/>

## Primary Silo

In a reliable deployment, one that is configured with membership using Azure Table and grain index using SQL server, all silos are created equal, with no notion of primary or secondary silos. That is the configuration that is recommended for production that will survive a failure of any individual node or a combination of nodes. For example, Azure periodically rolls out OS patches and that causes all of the role instances to reboot eventually.

For development or a non-reliable deployment when either MembershipTableGrain or DomainIndexGrain is used, one of the silos has to be designated as Primary and has to start and initialize before other, Secondary, silos that wait for Primary to initialize before joining the cluster. In case of a failure of the Primary node, the whole deployment stops working properly and has to be restarted.

In a non-Azure environment, Primary is designated in the configuration file with the following setting within the Globals section.

<SeedNode Address="<host name or IP address of the primary node>" Port="11111" />

# Client Configuration

The key parameter that has to be configured for a client is the silo’s client gateway endpoint(s) to connect to. There are two ways to do that: manually configure one or more gateway endpoints or point the client to the Azure Table used by silos’ cluster membership. In the latter case the client automatically discovers what silos with client gateways enabled are available within the deployment, and adjusts its connections to the gateways as they join or leave the cluster. This option is reliable and recommended for production deployment.

## Fixed Gateway Configuration

A fixed set of gateways is specified in the ClientConfiguration.xml with one or more Gateway nodes:

<ClientConfiguration xmlns="urn:orleans">

<Gateway Address="gateway #1" Port="30000"/>

<Gateway Address=" gateway #2" Port="30000"/>

<Gateway Address=" gateway #3" Port="30000"/>

</ClientConfiguration>

One gateway is generally enough. Multiple gateway connections help increase throughput and reliability of the system.

## Gateway Configuration Based on Cluster Membership

To configure the client to automatically find gateways from the silo cluster membership table, you need to specify the Azure Table connection string and the target deployment ID.

<ClientConfiguration xmlns="urn:orleans">

<Azure DeploymentId="target deployment ID" DataConnectionString="Azure Table connection string"/>

</ClientConfiguration>

## Local Silo

For the local development/test configuration that uses a local silo, client gateway should be configured to localhost.

<ClientConfiguration xmlns="urn:orleans">

<Gateway Address="localhost" Port="30000"/>

</ClientConfiguration>

# Typical Configurations

Below are examples of typical configurations that can be used for development and production deployments.

## Local Development

For local development, where there is only one silo running locally on the programmer’s machine, the configuration is already included in the SDK. The local silo that can be started with the [SDK-Root]\ StartLocalSilo.cmd script is configured as follows.

<OrleansConfiguration xmlns="urn:orleans">

<Globals>

<SeedNode Address="localhost" Port="11111" />

<Index ConnectionString="DomainIndexGrain"/>

</Globals>

<Defaults>

<Networking Address="localhost" Port="11111" />

<ProxyingGateway Address="localhost" Port="30000" />

</Defaults>

</OrleansConfiguration>

To connect to the local silo, the client needs to be configured to localhost and can only connect from the same machine.

<ClientConfiguration xmlns="urn:orleans">

<Gateway Address="localhost" Port="30000"/>

</ClientConfiguration>

## Reliable Production Deployment

For a reliable production deployment, you need to use the Azure Table option for cluster membership, and, if you are using any Orleans-managed grains (not self-managed), use SQL for grain index.

<OrleansConfiguration xmlns="urn:orleans">

<Globals>

<Liveness LivenessType ="AzureTable" />

<Azure DeploymentId="<your deployment ID>" DataConnectionString="DefaultEndpointsProtocol=https;AccountName=<Azure storage account>;AccountKey=<Azure table storage account key>"/>

<Index ConnectionString=" Data Source=<db>.database.windows.net;Initial Catalog=Orleans;User=your\_user@<db>.database.windows.net;Password=<password> "/>

</Globals>

<Defaults>

<Networking Address="" Port="11111" />

<ProxyingGateway Address="" Port="30000" />

</Defaults>

</OrleansConfiguration>

Clients need to be configured to use Azure Table for discovering the gateways.

<ClientConfiguration xmlns="urn:orleans">

<Azure DeploymentId="target deployment ID" DataConnectionString="Azure Table connection string"/>

</ClientConfiguration>

## Reliable Production Deployment with Only Self-Managed Grains

If you don’t use Orleans-managed grains, you don’t need a SQL database for grain index.

<OrleansConfiguration xmlns="urn:orleans">

<Globals>

<Liveness LivenessType ="AzureTable" />

<Azure DeploymentId="<your deployment ID>" DataConnectionString="DefaultEndpointsProtocol=https;AccountName=<Azure storage account>;AccountKey=<Azure table storage account key>"/>

</Globals>

<Defaults>

<Networking Address="" Port="11111" />

<ProxyingGateway Address="" Port="30000" />

</Defaults>

</OrleansConfiguration>

## Unreliable Deployment on a Cluster of Dedicated Servers

For testing on a cluster of dedicated servers when reliability isn’t a concern you can leverage MembershipTableGrain and DomainIndexGrain and avoid dependency on Azure Table or SQL. You just need to designate one of the nodes as a Primary.

<OrleansConfiguration xmlns="urn:orleans">

<Globals>

<SeedNode Address="<primary node>" Port="11111" />

<Liveness LivenessType ="MembershipTableGrain" />

<Index ConnectionString="DomainIndexGrain"/>

</Globals>

<Defaults>

<Networking Address=" " Port="11111" />

<ProxyingGateway Address=" " Port="30000" />

</Defaults>

</OrleansConfiguration>

<ClientConfiguration xmlns="urn:orleans">

<Gateway Address="node 1" Port="30000"/>

<Gateway Address=" node 2" Port="30000"/>

<Gateway Address=" node 3" Port="30000"/>

</ClientConfiguration>