



Lifting Off into Space-based Architecture with Magic xpi 4.x

Self-Paced Tutorial

Book ID: UTLBAMI4

Edition: 1.3, March 2015

Course ID: UCLOSAMI4

Magic University Official Courseware

The information in this manual/document is subject to change without prior notice and does not represent a commitment on the part of Magic Software Enterprises Ltd.

Magic Software Enterprises Ltd. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose.

The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms and conditions of the license agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement.

No part of this manual and/or databases may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or information recording and retrieval systems, for any purpose other than the purchaser's personal use, without the prior express written permission of Magic Software Enterprises Ltd.

All references made to third-party trademarks are for informational purposes only regarding compatibility with the products of Magic Software Enterprises Ltd.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of Magic xpi.

Magic™ is a trademark of Magic Software Enterprises Ltd.

Btrieve® and Pervasive.SQL® are registered trademarks of Pervasive Software Inc.

IBM®, Topview™, System i5/System i™, pSeries®, xSeries®, RISC System/6000®, DB2®, WebSphere®, Domino®, and Lotus Notes® are trademarks or registered trademarks of IBM Corporation.

Microsoft®, FrontPage®, Windows™, WindowsNT™, ActiveX™, Exchange 2007™, Dynamics® CRM, SharePoint®, Excel®, and Word® are trademarks or registered trademarks of Microsoft Corporation.

Oracle®, JD Edwards EnterpriseOne®, JD Edwards World®, and OC4J® are registered trademarks of the Oracle Corporation and/or its affiliates.

Google Calendar™ and Google Docs™ are trademarks of Google Inc.

Salesforce® is a registered trademark of salesforce.com Inc.

SAP® Business One and SAP® R/3® are registered trademarks of SAP AG in Germany and in several other countries.

Linux® is a registered trademark of Linus Torvalds.

UNIX® is a registered trademark of UNIX System Laboratories.

GLOBETrotter® and FLEXlm® are registered trademarks of Macrovision Corporation.

Solaris™ and Sun ONE™ are trademarks of Sun Microsystems Inc.

HP-UX® is a registered trademark of the Hewlett-Packard Company.

Red Hat® is a registered trademark of Red Hat Inc.

WebLogic® is a registered trademark of BEA Systems.

Interstage® is a registered trademark of the Fujitsu Software Corporation.

JBoss™ is a trademark of JBoss Inc.

GigaSpaces, GigaSpaces eXtreme Application Platform (XAP), GigaSpaces eXtreme Application Platform Enterprise Data Grid (XAP EDG), GigaSpaces Enterprise Application Grid, GigaSpaces Platform, and GigaSpaces, are trademarks or registered trademarks of GigaSpaces Technologies.

Clip art images copyright by Presentation Task Force®, a registered trademark of New Vision Technologies Inc.

This product uses the FreeImage open source image library. See <http://freeimage.sourceforge.net> for details

This product uses icons created by Axialis IconWorkShop™ (<http://www.axialis.com/free/icons>)

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes software developed by Computing Services at Carnegie Mellon University (<http://www.cmu.edu/computing/>).

Copyright © 1989, 1991, 1992, 2001 Carnegie Mellon University. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes software that is Copyright © 1998, 1999, 2000 of the Thai Open Source Software Center Ltd. and Clark Cooper.

This product includes software that is Copyright © 2001-2002 of Networks Associates Technology Inc All rights reserved.

This product includes software that is Copyright © 2001-2002 of Cambridge Broadband Ltd. All rights reserved.

This product includes software that is Copyright © 1999-2001 of The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved.

All other product names are trademarks or registered trademarks of their respective holders.

Lifting Off into Space-based Architecture with Magic xpi 4.x

March 2015

Copyright © 2013-2015 by Magic Software Enterprises Ltd. All rights reserved.

Table of Contents

Introduction	5
About Magic xpi 4.x's Space-based Architecture	5
About the Course	6
How to Use This Guide	7
Exercises	7
Course Materials	7
Installing the Course Materials	7
Benefits of Space-based Architecture	9
Tier-based Architecture	10
Magic xpi 4.x Architecture at a Glance	10
Summary	12
Magic xpi Architecture with GigaSpaces	13
Magic xpi's Architecture	14
In-Memory Data Grid Terminology	15
Summary	16
Project Execution	17
Startup Mechanism	18
Workers and Triggers	23
Variables	25
Licensing	25
Summary	27
Recovery and Monitoring	29
Recovery	30
Monitoring	34
GigaSpaces User Interface Monitor	36
Exercise	39
Summary	39
Deployment	41
Deploying the Magic Space	42
Partitions and Containers	44
Memory Allocation	49
Clustering	50
Starting Projects from the Command Line	54
Starting and Stopping Projects from the Monitor	54
Exercise	55
Summary	55
Troubleshooting	57
GigaSpaces UI Troubleshooting	58
Log Files	60
Project Startup Troubleshooting	60
Exercise	63
Summary	64
Glossary	65
Magic xpi Space-related Terminology	65
In-Memory Data Grid Terminology	66



Magic®
University

Introduction

Welcome to Magic Software University's **Lifting Off into Space-based Architecture with Magic xpi 4.x** self-paced tutorial. We, at Magic Software University, hope that you will find this tutorial informative and that it will assist you in getting started with this exciting product.

About Magic xpi 4.x's Space-based Architecture

While Magic xpi 3.x was based on the Magic Request Broker middleware, Magic xpi 4.x uses an In-Memory Data Grid (IMDG) as its underlying messaging and context persistency layer.

You can easily prepare to handle new demands created by enterprise mobility and cloud services and take advantage of the immediate benefits and capabilities of Magic xpi 4.x including:

- Built-in clustering and fail-over capabilities
- Unlimited linear and elastic scalability
- High availability
- Automated recovery
- Improved management and monitoring capabilities

About the Course

Magic University's **Lifting Off into Space-based Architecture with Magic xpi 4.x** course is intended for people with experience in working with Magic xpi 3.x or iBOLT V3.x, and who want to take full advantage of the Space-based architecture on which Magic xpi 4.x is built.

In this course you will learn about:

- The benefits of Space-based architecture
- The architecture of Magic xpi and its new Space-based infrastructure
- Magic xpi 4.x's startup mechanism and licensing
- The Magic xpi recovery processes
- The improved monitoring capabilities
- What to consider when deploying Magic xpi 4.x
- Some useful guidelines for troubleshooting Magic xpi 4.x

Course Prerequisites

Before you start with the course there is basic knowledge that you need to have:

Development knowledge	Familiar with Magic xpi 3.4 or iBOLT V3.x
-----------------------	---

Your computer must also meet some basic requirements:

Hardware	<ul style="list-style-type: none"> • Windows XP Pro and later. The course was tested on Windows 7 • Pentium processor 1.8GHz and upwards • 4Gb RAM or greater • At least 1Gb of free space • Screen resolution of at least 1024x768 pixels
MS SQL Server	MS SQL version 2000 and above. You will need the supervisor password for this installation.
.NET Framework	<ul style="list-style-type: none"> • Verify that either .NET Framework version 2.0, 3.0, or 3.5 is installed on your machine. Ensure that one of these is the .NET CLR 2.0 version. • Also verify that .NET Framework 4.0 is installed on your machine.
License	The course uses the standard license. Please obtain a Magic xpi 4.x IBNPSRV evaluation license from your local Magic Software Enterprises representative.

How to Use This Guide

To get the most out of this guide, follow the classroom lesson and ask any questions that you have. You can then review the self-paced guide for that lesson, and if you have further questions you can ask the instructor before the next lesson.

The self-paced guide provides detailed step-by-step instructions. If you are learning using this self-paced tutorial, feel free to contact your Magic Software Enterprises representative or the Support department for further assistance.

Exercises

At the end of most lessons, there is an exercise. These are based on what you have learned during the lesson. In some of these exercises, you are asked to work with a second computer to try and get the best productivity out of the exercise.

Course Materials

The course includes the following:

- Course project – This is the course project, **SpaceCourse**. During this course you will use this ready-built project. The project is a simple project with a single flow containing a Scheduler service that is invoked every two seconds. There are five Delay services that are invoked every second to copy XML files, and another Delay service that deletes all of the created files.

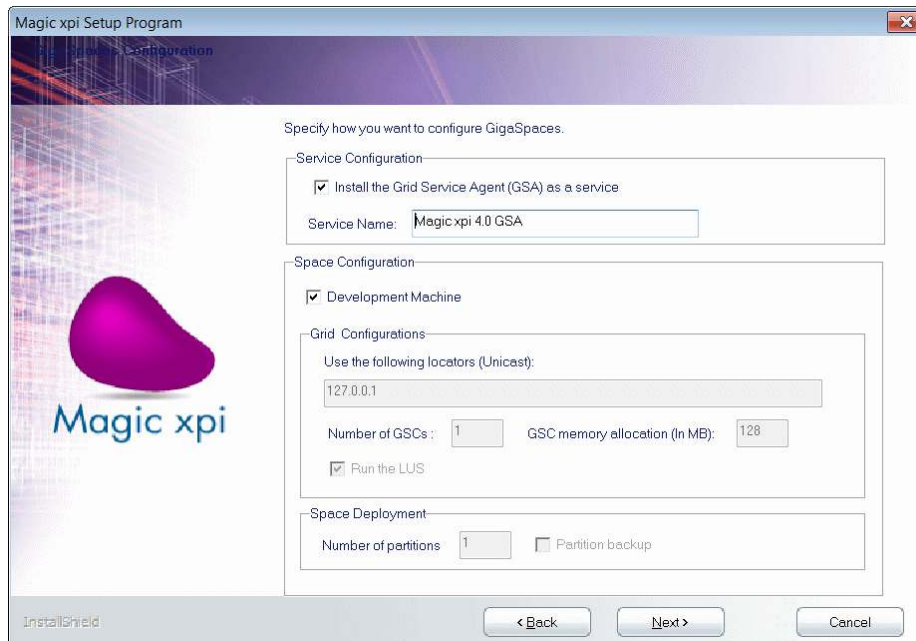
Installing the Course Materials



Install the Magic xpi 4.x Suite by running the **Setup.exe** file. Use the default settings.

- Copy the course project **SpaceCourse** to your **projects** directory.
- Please ensure that your computer meets the hardware requirements listed on the previous page.

A number of installation screens including the screen where you can set the Space-based configuration.



For this tutorial, there is no need to change any of the defaults. You will learn about the different options later on in this tutorial when you learn about working with a clustered environment.

Magic xpi 4.x Licensing

Magic xpi 4.x does not work with licenses from previous versions. You need to request a new license from your Magic Software Enterprises representative.

The new license contains MAGIC 2.000 version and the flag VERSION=4.0, as part of the Vendor String.

```
FEATURE IBPRSRVI MAGIC 2.000 31-may-2014 35
VENDOR_STRING=PT=MGENT2,C=00FF,P=N,IBOLT=Y,VERSION=4.0
HOSTID=ANY DUP_GROUP=NONE ISSUER="Magic xpi Non-production
server" SN=
```

Upgrading from Previous Versions

Upgrading to Magic xpi 4.x is simple. The method is the same as earlier versions. Copy your previous version's project into your new project's folder. Then launch Magic xpi 4.x and open the project. Magic xpi 4.x recognizes that this project was developed in a previous version and opens a dialog box where you can select to upgrade to the current version.

Lesson 1

Benefits of Space-based Architecture

Magic xpi 4.x now features a Space-based architecture, thereby enhancing the capability of the product. This lesson introduces you to the benefits of Magic xpi's new Space-based architecture and how it improves your final product.

This lesson covers various topics including:

- Tier-based architecture
- Magic xpi 4.x's architecture at a glance
- In-Memory Data Grid
- Replication
- Clustering

Tier-based Architecture

Magic xpi 3.x is based on the Magic Request Broker middleware, which uses a tier-based architecture. Tier-based architecture is comprised of separate tiers, including a business logic tier, a data tier and a web client, and sometimes also a web tier and a messaging tier. One main issue with the separate tiers is that a bottleneck can occur in any of the tiers, which makes maintaining and troubleshooting tier-based projects time consuming and expensive. Each tier requires a different skill set, thereby increasing the manpower needed for its upkeep.

Magic xpi 4.x Architecture at a Glance

With Magic xpi 4.x, you can now set up any of the following configurations:



- Single host + Single engine
- Single host + Multiple engines
- Multiple hosts + Multiple engines

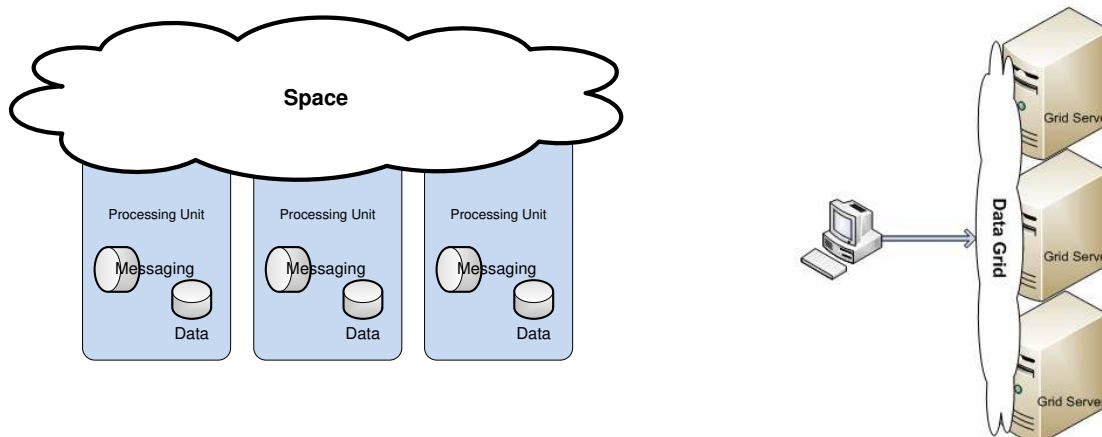
All of these configurations are available thanks to the In-Memory Data Grid.

In-Memory Data Grid

In Magic xpi 4.x, the broker was replaced by the **In-Memory Data Grid (IMDG)**. One of the biggest benefits of the new Space-based architecture is that the IMDG does away with a single point-of-failure by spreading out the logic across multiple servers running on multiple machine instances (physical or virtual). In the In-Memory Data Grid, the business logic tier, data tier and the messaging tier reside together on each of the **space partitions** and are stored in memory. Since the data is managed in memory, disk access is reduced.

Each partition handles all of the logic in a **processing unit**. Each processing unit holds a subset of the data. Together, the partitions make up the **Magic Space**, which for all intents and purposes, works like a database.

The processing units are independent of one another, so that the application can scale upwards by adding more processing units. The IMDG can contain multiple spaces.



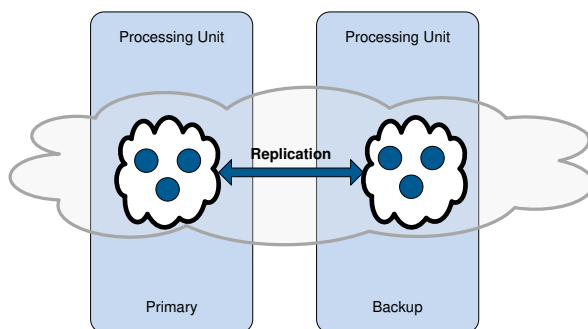
The capabilities of the new architecture provide built-in active/active clustering and fail-over capabilities for Magic xpi projects. It also enables unlimited linear scalability and improves performance.

As you can see in the image on the right, you can add servers to the grid to form a logical grid. You can add servers without disrupting the other servers, thereby extending the data grid. You will learn more about how this works and the various components later in this tutorial.

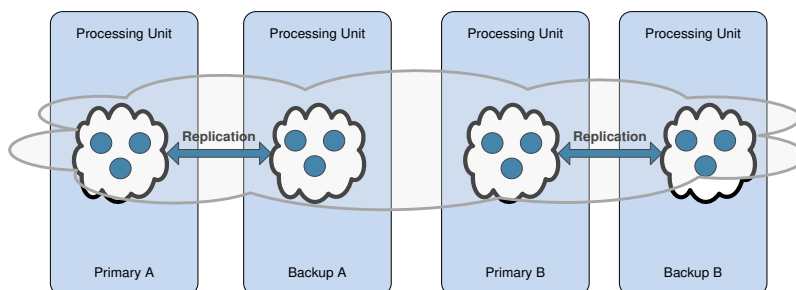
Replication

In the Magic xpi architecture, each partition can have a backup, which ensures high availability of the information. This improves data consistency and provides high availability of the data.

In replication, you have a partition which is active, known as the primary partition, and another partition which is the backup, often residing on a different server or on a different process on the same server. The so-called primary partition replicates the data with the backup partition.



If the primary partition fails, the backup partition immediately becomes active as the primary partition and already contains the entire information set. The system will then try to automatically load another processing unit that will behave as the new backup unit. This mechanism ensures that the system is self-healing and has zero downtime.



Clustering

In Space-based architecture, the Magic Space can reside on several computers (physical or virtual) that are viewed as a single logical unit, also known as a cluster. A cluster can be seen as a single "large" Space and enhances your applications' scalability, high availability and load balancing. You can have a combination of both clustering and partitioning to ensure that the Magic Space and the data are always available.

Summary

In this lesson, you learned about the benefits of Space-based architecture. In this architecture, you have a processing unit that is made up of three layers: business logic, data, and messaging. One or more processing units form what is known as the Magic Space. An incoming request will only see the Magic Space, and not each individual processing unit. As a result, it is easy to add processing units to keep up with demand.

The Magic Space itself is an entity on the In-Memory Data Grid. Data is kept in memory, thereby providing faster access to the data. This makes each process faster.

To develop a fail-safe system, you can use a primary-backup implementation and clustering, which ensure zero downtime of your system.

In the next lesson, you will learn how Magic xpi works with the Space-based architecture.

Lesson 2

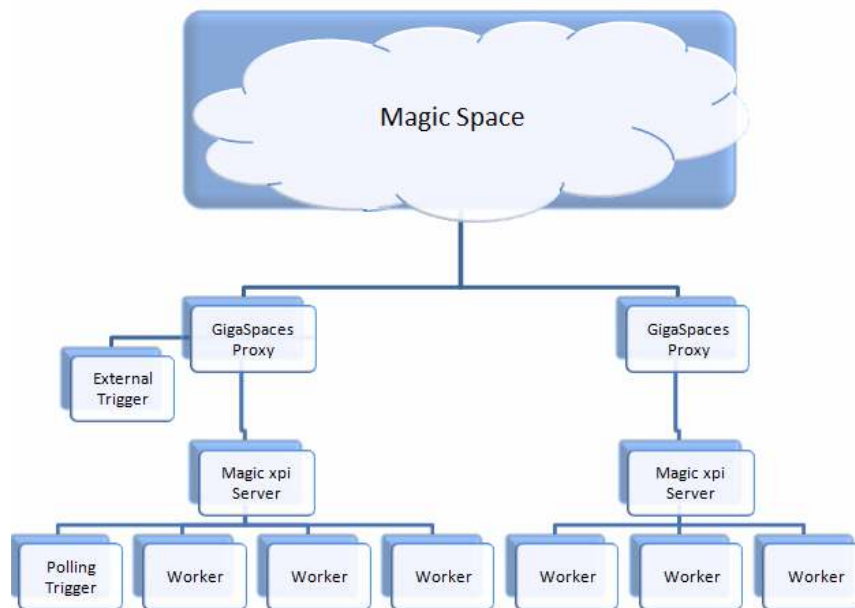
Magic xpi Architecture with GigaSpaces

GigaSpaces' XAP in-memory computing technology is the middleware that implements Magic xpi's functionality on the In-Memory Data Grid. The GigaSpaces middleware is the underlying messaging and context persistency layer of Magic xpi.

This lesson covers various topics including:

- A detailed explanation of Magic xpi's Space-based architecture
- In-memory grid terminology

Magic xpi's Architecture



This section describes the various parts of Magic xpi's new architecture. A more detailed explanation of each of the parts will be described in the next lesson.

To enable complete scalability and independence, Magic xpi 4.x loads the project's metadata and runtime context into the Magic Space and not into the process memory of the local machine as in previous versions. By doing this, all shared data is available to all the Magic xpi servers and workers running a project, and on all machines participating in the grid. This enables any project to easily scale to any number of machines.

The Magic xpi servers (engines) communicate with the Magic Space through a **GigaSpaces proxy**. The GigaSpaces proxy is the GigaSpaces module (the middleware layer) that connects client applications to the Space. The Magic Space can reside on a number of processes and a number of machines as a single unit. It is the responsibility of the GigaSpaces proxy to make the right connections between the servers and the Space partitions.

Three elements work together when the server is up and running: workers, polling triggers and external triggers.

Workers

In general, a worker is a thread that takes a message from the space and executes the message (the flow logic).

In previous versions, requests were sent to the broker and it was the broker's duty to find a server (engine) that could handle the request. This is known as a *push mechanism*. If there were no available threads to handle the request, the broker would put them in a queue until there was an available thread or a timeout was reached. If there was a problem with the broker, no requests were handled.

In Magic xpi 4.x, the broker has been replaced with a *pull mechanism*. In Magic xpi 4.x, the server has zero or more **workers** (a pool of workers) that are simultaneously actively searching the Magic Space for requests to handle.

Triggers

Polling Triggers

A polling trigger (such as the Directory Scanner), also known as an asynchronous trigger, is a Magic xpi server thread that constantly checks external systems (such as an email account) to see whether it needs to invoke a flow without waiting for a response. When it needs to invoke a flow, it places a message in the Magic Space.

External Triggers

An external trigger, such as the HTTP Requester or a Web server, is an external application. Once the trigger receives a request, it places a message in the Magic Space that invokes the flow. The synchronous external triggers will also wait for a response message.

In-Memory Data Grid Terminology

You will come across the following terms when you work with GigaSpaces. Throughout the tutorial you'll be introduced to these concepts and see how they tie into the architecture. You can find a more detailed explanation of these terms in the Glossary at the end of this tutorial.

- Grid Service Agent (GSA) – This is installed by the Magic xpi installation as an operating system process. It runs on a specific machine and is responsible for adding that machine to the grid, monitoring grid processes on its own machine, and restarting them in case of process failure.
- Grid Service Container (GSC) – A container that hosts processing units or Space partitions.
- Grid Service Manager (GSM) – The service that manages the Grid Service Containers. The GSM deploys and undeploys processing units to the grid. The GSM monitors the processing units running within each GSC.
- Lookup Service (LUS) – A registry that contains information as to where each loaded component is actually situated.

Summary

In this lesson, you were introduced to the terminology used in Space-based architecture. You learned about the various grid components and the part that they take in the flow cycle.

You learned about the new Magic xpi 4.x architecture, and how the message mechanism has changed from a *push* mechanism to a *pull* mechanism.

Lesson 3

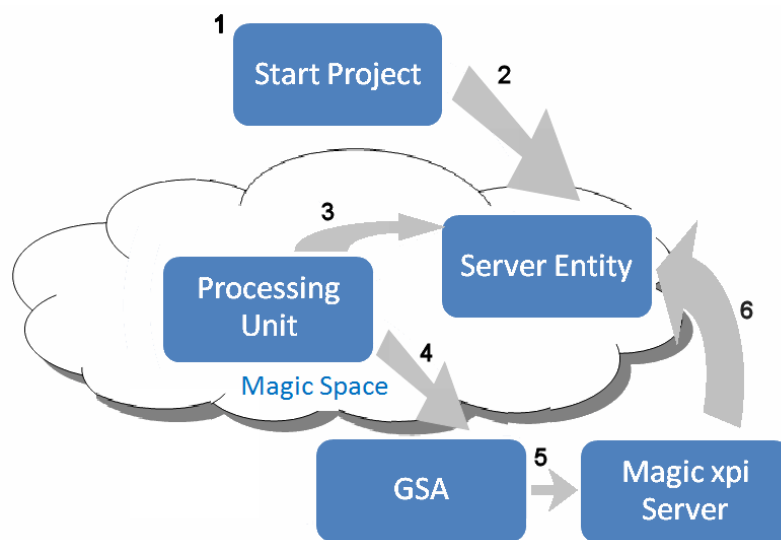
Project Execution

How the project loads and runs has been enhanced to ensure that the project works smoothly and effortlessly.

This lesson covers various topics including:

- Startup mechanism
- Workers
- Triggers
- Licenses

Startup Mechanism



In general, when you start a project, this is the cycle that is initiated.

1. The project is started.
2. The start option creates a server entity in the Magic Space and updates the status in the Advanced Monitoring Console with **START_REQUESTED**.

Status
START_REQUESTED

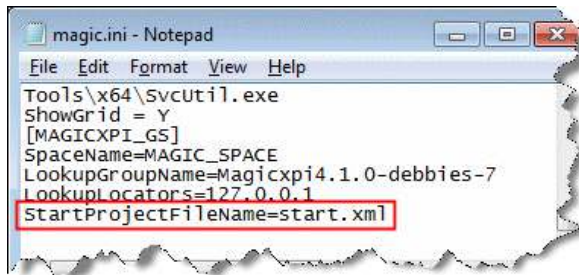
 (You'll learn more about the statuses in a later lesson.)
3. The Magic processing unit scans the Magic Space looking for server entities whose status is **START_REQUESTED**.
4. Once the Magic processing unit finds a server entity with a status of **START_REQUESTED**, it scans for the Grid Service Agent (GSA) according to the host name or IP address defined in the server entity. The Magic processing unit passes all of the parameters defined in this server entity to the GSA.
5. The GSA starts the Magic xpi server according to parameters passed to it from the Magic processing unit.
6. When the Magic xpi server connects to the Magic Space, it searches for the server entity according to its ID. The Magic xpi server then updates the server entity's status to **RUNNING**, and the cycle is complete. The project starts executing.

In the following sections, you will read more about the steps described above. First though, it's important that you become familiar with the **start.xml** configuration file.

Start.xml File

The **start.xml** configuration file is created by the Magic xpi Studio during the build process (from the **File** menu), and only if it does not already exist.

This is the configuration file that will be used when the project is executed. The file is located in the **projects** folder. The name of the **start.xml** file is set in the **StartProjectFileName** property in the **Magic.ini** file's new **[MAGICXPI_GS]** section.



Any modifications to the configuration file need to be done manually using an external editor, such as Notepad or any XML editor.



If you deploy your project on a different machine than the development machine, you need to modify the **start.xml** file accordingly.

Here is an example **start.xml** file:

```

<Magicxpi_Startup>
  <Projects>
    <Project Name="SpaceCourse" ProjectsDirPath="c:\Magic xpi 4.1\projects">
      <Servers>
        <Server host="debbies-7" alternateHosts="">
          <ProjectsDirPath></ProjectsDirPath>
          <NumberOfWorkers>10</NumberOfWorkers>
          <NumberOfInstances>1</NumberOfInstances>
          <Triggers load="true">
          <Scheduler load="true"/>
          <AutoStart load="true"/>
        </Server>
      </Servers>
    </Project>
  </Projects>
</Magicxpi_Startup>
  
```

As you can see in the example above, from within the configuration file you can define:

- An alternate host (**alternateHosts**). This flag gives you the option to define an alternate host for the server to work with if the main host is unavailable or if the startup procedure on the main host fails.
- The number of workers that the server will load (**NumberOfWorkers**). You can load a server without workers, **NumberOfWorkers=0**, if you want, for example, that the server will only listen for triggers.
Note: The number of workers that will execute a flow at any given time is bound by the number of threads in the license.
- The number of instances of the Magic xpi server (**NumberOfInstances**). If you set the number of workers as **10** and the number of instances as **2**, you will have two identical servers with 10 workers each.
- Whether this server will load the triggers. When you load more than one server to handle a specific project, there is no need for each server to handle the triggers. You should define that a specific server will listen for trigger events and the other servers will not.
- Whether the server will load the scheduler. Note that only a single server should load the scheduler; otherwise, you will have duplicate scheduled invocations.
- Whether the server will execute the AutoStart flows. If you have a flow that initializes database tables, you would not want all the servers that load to initialize the tables when they load.


Remember that with Magic xpi 4.x, you can set up any of the following configurations:



- Single server + Single engine + Multiple workers
- Single server + Multiple engines + Multiple workers
- Multiple servers + Multiple engines + Multiple workers

Step 1: Starting a Project

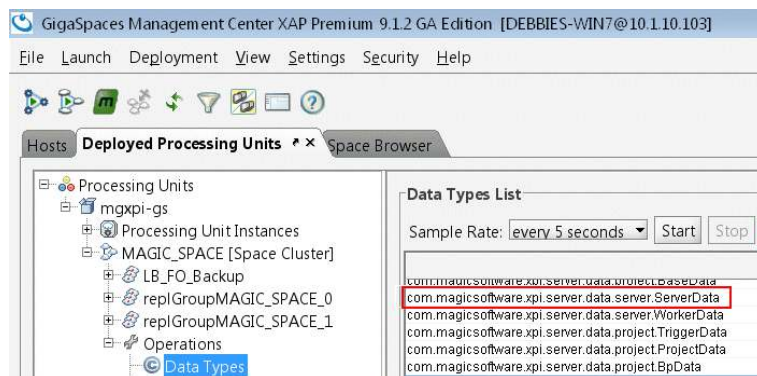
Starting a project is done in one of three ways:

1. Clicking the  **Start** link. This **Start** link is created in the project's directory when the project is built. The link points to the **start.xml** configuration file.
2. Clicking the **Start** option from the Monitor or the Debugger. This option also uses the **start.xml** file that is in the project folder.
3. Automatically, by creating a file called **projectsStartup.xml** and saving it in the **<Magic xpi installation>/config** folder. When the Magic xpi service starts and manages to deploy the Magic Space, it will automatically start the projects and servers that are listed in the **projectsStartup.xml** file. The structure of the **projectsStartup.xml** is identical to the structure of the **start.xml** created under each project. Details about how to work with this file are discussed later on in the Deployment lesson.

Step 2: Magic Server Entity Created

When you start the project as described in step 1, the engine creates an entity (or entities) in the Magic Space with the metadata from the project's **start.xml** file, a unique ID and a status of **START_REQUESTED**. Every time you start the project, another instance of the entity is created in the Magic Space.

In the **GigaSpaces Management Center**, you can see that a data type called **ServerData** is created. This is the server entity that is created when you start a project. This screen will be discussed later on in this tutorial.



Steps 3 and 4: Magic Processing Unit

The Magic processing unit scans the Magic Space looking for server entities whose status is **START_REQUESTED**. Once it finds a server entity with a status of **START_REQUESTED**, the Magic processing unit scans for the Grid Service Agent (GSA) according to the host name (as shown in the image below) or the IP address defined in the server entity, which is based on the **start.xml** file. The Magic processing unit passes all of the startup information (parameters) defined in the server entity to the GSA.



If the GSA is not available, the Magic processing unit will keep checking at a later stage to see if it is available.



In the **start.xml** configuration file, you can define machines that are not running or you can define that the GSA is not running. You can start the GSA on those machines when the workload is high. Once the GSA loads, the Magic processing unit will find it and the cycle will begin.

Step 5: Starting the Magic xpi Server

The GSA running on the host machine specified in the server entity starts the Magic xpi Server (the **MgxpServer.exe** process, also known as the engine) according to the parameters passed to it from the Magic processing unit.

- If a project is not running yet, the first **MgxpServer.exe** process that starts will create the project by loading the project entity to the Magic Space.
- If a project is already running, the loaded **MgxpServer.exe** process will join the existing project and add its own workers and triggers to the work force.

Step 6: Server Up and Running

One of the parameters passed to the Magic xpi project is also the ID of the server entity created in the Space. When the Magic xpi server registers, it connects to the Magic Space and searches for the server entity identified by the ID and updates the status to **RUNNING**. The server can now start handling requests.

Startup Failed

If a server entity's status remains as **START_REQUESTED** and does not change to **RUNNING**, the Magic processing unit checks with the GSA to see whether the actual process is running.

- If the process is not running, the server entity's status will change to **START_FAILED**.
- If the process is running, the Magic processing unit will first instruct the GSA to terminate the process and then it will update the server entity with **START_FAILED**.

Why the startup fails and how to deal with this are discussed later on in the lesson about troubleshooting.

Workers and Triggers

Now that the project is running, three elements work simultaneously: workers, polling triggers and external triggers. As was said earlier, each Magic xpi server can run one or more workers/triggers for a variety of tasks. Here is an explanation of the actual process that the workers and triggers go through.

Workers

Each worker scans the Magic Space for jobs to execute (messages whose status is **READY_FOR_USE**). The message in the Magic Space contains all of the data (payload) that the worker needs to execute the flow. This is known as the *message payload*.

When the worker finds a message with a status of **READY_FOR_USE**, it changes the status to **IN_PROCESS**. It does this within a transaction to prevent two workers from executing the same message. The worker then executes the flow.

Created Time	Message ID	Message Status	Invocation Type
Mon Dec 30 10:20:58	2606	IN_PROCESS	Scheduler
Mon Dec 30 10:20:59	2609	IN_PROCESS	parallel
Mon Dec 30 10:21:03	2632	IN_PROCESS	parallel
Mon Dec 30 10:21:03	2633	IN_PROCESS	parallel
Mon Dec 30 10:21:03	2634	READY_FOR_USE	Scheduler



When a worker takes the flow message from the Magic Space together with its payload, it does not remove the flow message from the Space. It remains there until the flow has completed. One reason for this is to be able to restart the flow during recovery.

Triggers

Polling Triggers

The polling trigger constantly checks external systems to see whether it needs to invoke a flow without waiting for a response. When it needs to invoke a flow, it places a message with a status of **READY_FOR_USE** in the Magic Space.

External Triggers

Once an external trigger receives a request, it places a message in the Magic Space that invokes the flow and waits for a response message.

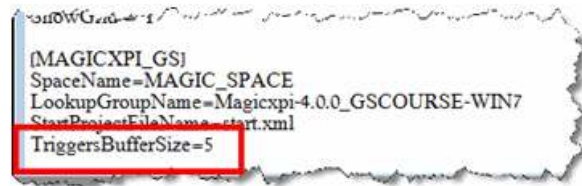
Trigger Buffer

Each trigger has a predefined queue or buffer. Once the queue is full, the trigger will wait before polling the system and creating a new flow request message in the Magic Space.

The default size of the trigger buffer is set to **10**. This is set internally by the Magic xpi server.

However, you can override this value by changing the **TriggersBufferSize** in the **Magic.ini** file. By default, this flag does not appear in the **Magic.ini** file. To override the value, you should:

1. Open the **Magic.ini** file in Notepad.
2. Go to the **[MAGICXPI_GS]** section of the **Magic.ini** file.
3. Add a line such as **TriggersBufferSize=5**.



Remember that this flag defines the buffer size for all triggers.

Multiple Triggers

You can use multiple instances of some Magic xpi triggers in your projects and the distribution of triggers over the project's engines is configurable at deployment time. This not only allows you to control where each trigger will run, but also to set the same trigger to run on more than one engine. This lets you scale the trigger processing capability.

You use the **start.xml** file to define which triggers will run on each server. Whenever you build or rebuild a project that contains triggers, an example **start.xml.triggers** file is created under the **projects/<project name>** folder.

In the **start.xml.triggers** file, you will see these trigger details expressed in the following way:

```
<Trigger BP_ID="1" Flow_ID="3" Trigger_ID="2" Name="SFDC Customers sync"/>
<Trigger BP_ID="1" Flow_ID="7" Trigger_ID="3" Name="Check Emails"/>
<Trigger BP_ID="2" Flow_ID="5" Trigger_ID="5" Name="SugarCRM Contacts"/>
```

To determine which triggers will run on which server, you need to copy and paste the trigger details into your project's **start.xml** file, as shown in red in the image below.

```
<Server host="debbies-7" alternateHosts="">
  <ProjectsDirPath></ProjectsDirPath>
  <NumberOfWorkers>10</NumberOfWorkers>
  <NumberOfInstances>1</NumberOfInstances>
  <Triggers load="true">
    <Trigger BP_ID="1" Flow_ID="3" Trigger_ID="2" Name="SFDC Customers sync"/>
    <Trigger BP_ID="1" Flow_ID="7" Trigger_ID="3" Name="Check Emails"/>
    <Trigger BP_ID="2" Flow_ID="5" Trigger_ID="5" Name="SugarCRM Contacts"/>
  </Triggers>
  <Scheduler load="true"/>
  <AutoStart load="true"/>
</Server>
```


After the Main Flow Is Completed

When the main flow is completed, the worker updates the request message's status in the Magic Space to **DONE**, and is then free to scan for additional messages.

If the message is from a synchronic trigger (such as an HTTP trigger), when the flow ends, the worker writes a response message to the Magic Space, which the trigger will send to the client. Every parallel and stand-alone branch is also handled as a separate message written by the flow to the Magic Space. This new thread (the parallel or stand-alone branch) can be handled by any worker of the project, even a worker running on a different machine.

Variables

Since business process and global variables are shared by all servers belonging to the same running project, these values are stored in the Space and not in each processes' memory, which is where they were stored in Magic xpi 3.x.

Licensing

In Magic xpi 4.x, all licenses are floating licenses with an option to reserve a fixed (minimum) number of license threads for each project. Each license entry has an entry for the number of workers (threads) that the server can execute concurrently. The licenses are available for use by all of the projects. The Magic Space acts as the license server. Before executing a flow, the worker will attempt to check-out the license from the pool. When it has finished executing the flow, the worker will check the license back into the pool.

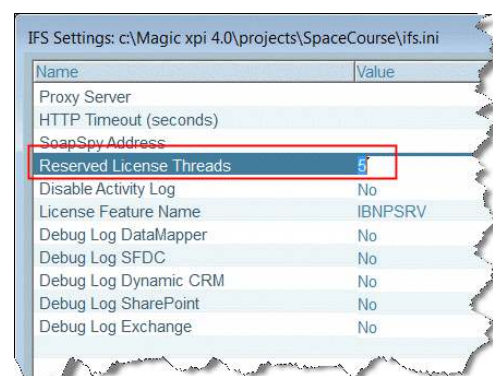


In contrast with previous versions where you defined the maximum number of licences that the project will consume, in Magic xpi 4.x you define the **minimum** number of licenses that you need.

Reserving License Threads for a Project

If you are running multiple projects, and some projects may consume all threads under stress conditions (for example, there are many Web services and/or HTTP requests), you should consider reserving a fixed (minimum) amount of license threads for critical projects that need to run continually.

As with previous versions, you can define reserved licenses for a certain project as well. You do this by opening the **IFS Settings** dialog box, and setting the **Reserved License Threads** property. When the project is executed, the workers will check out the licenses defined in this property, but will not check them back in. The license is not



released even when the worker has completed the flow. The project can also use other licenses from the pool.

If a server is manually shutdown or stopped in the Advanced Monitoring Console, the reserved licenses will be returned to the pool. Servers handling that project will continue consuming floating licenses. However, when the server is restarted or another server is loaded for that project, the reserved licenses will once again be checked out.

Host Locked License

All licenses produced by Magic Software Enterprises are now host locked. This means that once activated, you must use them on the machine used for the activation. When a Magic xpi project loads from this server, it adds the license into the Magic Space according to the **License Feature Name** defined in the `ifs.ini` file, such as `IBPRSRVI`, and the serial number defined in the license file.

All Magic xpi servers will start normally, even if they are running on a different host, but they will not run any flows until the server that was host locked runs a Magic xpi server. The Magic xpi server that runs on the host locked server is the only one that can update the thread count in the Magic Space. The Magic xpi server on the host locked server does not have to keep running. Once it sets the thread count in the Magic Space, it is not needed to maintain the license. It can therefore be stopped and started like any other Magic xpi server on the grid.

License for Clustered Environments

In Magic xpi 4.x clustered environments, you only need a single host locked license for the entire cluster. All Magic xpi installations should be configured to point to the same license file (you can put it in a shared location). This will be discussed in more detail in the Clustering section of this tutorial.

Different Serial Numbers

Each feature is based on a serial number. If you have two licenses with the same license feature name but different serial numbers, the total number of workers available will be the combination of both licenses. However, the expiration date of the license will be the earliest date. This can happen if you have two separate servers that have licenses with activated hosts.

Refreshing a License

When you receive a new license, which may either contain an extension of the expiration date or more license threads, you need to refresh the license record in the Magic Space. The refresh process is simple; you simply need to load a Magic xpi server on the server activated for the license. This will immediately update the information on the Magic Space. If you added license threads, the threads will then be available for all of the running workers.

The license will not be updated if there are fewer licenses than before, or if the expiration date is earlier than the previous license.

Summary

In this lesson, you learned how the Magic xpi project loads and all of the steps involved in the startup mechanism.

You also learned about the **start.xml** configuration file and each of its flags.

You then learned that the Magic xpi 4.x license is now host locked and that you select the minimum number of required licenses.



Magic®
University

Lesson 4

Recovery and Monitoring

Magic xpi 4.x includes robust recovery processes enabling you to develop projects that recover automatically from many disaster and failure scenarios. Magic xpi 4.x also provides various monitoring capabilities including the Monitor that you are familiar with from Magic xpi 3.x, the Advanced Monitoring Console, the new Magic Monitor and the monitoring abilities available using the GigaSpaces user interface.

This lesson covers various topics including:

- Magic xpi server recovery
- Worker recovery
- Work process recovery
- Magic xpi Monitor
- Advanced Monitoring Console
- Magic Monitor
- GigaSpaces user interface

Recovery

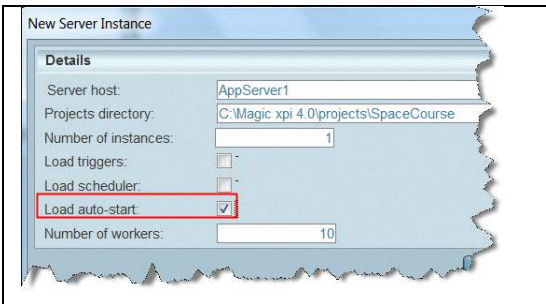
The new architecture provides Magic xpi 4.x with an enhanced recovery mechanism. The Grid Service Agent is able to automatically reload components that have crashed. When a primary partition crashes in the Magic Space, the backup immediately takes over. Magic xpi also has its own recovery mechanisms.

Magic xpi Server Recovery

All of the Magic xpi entities, meaning the Magic xpi server and the workers, have a “keep alive” mechanism. At certain predefined intervals, they update the Magic Space with a timestamp to show that they are still active.

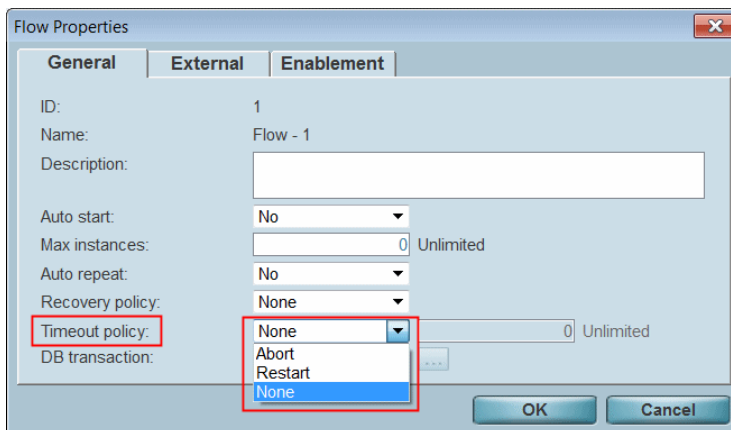
With the Magic xpi server (engine), if the Magic Space is not updated with a new timestamp after a fixed predefined internal timeout interval of two minutes, the Magic processing unit uses the same mechanism that you learned about earlier in this lesson in the **Startup Failed** section. However, instead of changing the status to **START_FAILED** as it does in the startup scenario, here it loads another Magic xpi server. The steps here are:

1. The Magic processing unit checks with the Grid Service Agent to see whether the process is running.
2. If the process is running, the Magic processing unit instructs the GSA to terminate the process.
3. It instructs the GSA to load another Magic xpi server, thereby recovering the server and updates the server entity with **START_REQUESTED**.
4. If the **ServerData** entity contains **AutoStart load="true"** for a server that has crashed, Magic xpi will not perform another autostart when the engine recovers. You can see this setting in the Advanced Monitoring Console or in the **start.xml** file.

	<pre data-bbox="772 1384 1414 1686"><?xml version="1.0" encoding="UTF-8"?> - <Magicxpi_Startup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> - <Projects> - <Project ProjectsDirPath="c:\Magic xpi 4.0\projects\" Name="SpaceCourse"> - <Servers> - <Server host="debbies-7"> <ProjectsDirPath/> <NumberOfWorkers>10</NumberOfWorkers> <NumberOfInstances>1</NumberOfInstances> <Triggers load="true"></Triggers> <Scheduler load="true"/> <AutoStart load="true"/> </Server> </Servers> </Project> </Projects> </Magicxpi_Startup></pre>
Advanced Monitoring Console	start.xml file

Worker Recovery

The worker also updates the Magic Space with a time interval to show that it is still alive. If the Magic xpi server sees that a worker is not responding, it will check to see whether the worker thread is still alive. If it is no longer alive, the Magic xpi server will load another in its place. However, if the process is still running and is not responding, it may be processing an external task, such as an SQL statement. The Magic xpi server will wait until the timeout duration has exceeded, and then will follow the **Timeout policy** defined in the flow properties. If no policy has been defined, Magic xpi will wait indefinitely.



If you are using multiple Magic xpi Servers, make sure that all of the machines' clocks are synchronized.

Work Process Recovery

A work process refers to the execution tree of an entire root flow (the main flow) including its child flows. The work process as a whole is treated as a single business transaction. The root flow is a flow that was not invoked by a parent flow. The flow may have been started by a trigger, the Scheduler, Auto-start, or a Publish/Subscribe scenario.

Magic xpi uses the recovery policy defined in the root flow and ignores the recovery policies defined for the child flows.

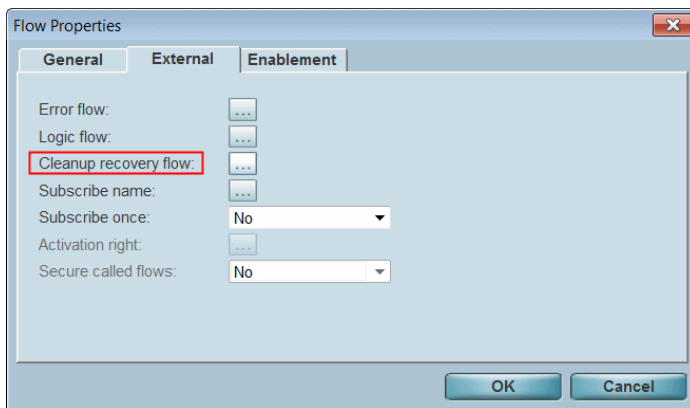


A stand-alone branch is not considered part of the work process since it is detached from the work process. Therefore, it has its own execution tree. If a stand-alone branch crashes, the recovery policy will be taken from its own flow properties and not from the main flow.

The Magic processing unit constantly scans the Magic Space to ensure that all processes are running. If the Magic processing unit identifies a flow message whose status is **IN_PROCESS**, but no active worker is handling it, the Magic processing unit identifies this as a problem. This means that somehow a worker crashed and did not finish its process, and that there is still a flow message that is only partially handled. As you learned earlier, another worker will be loaded to replace the worker, but the flow still needs to be cleaned up.

The flow may have various parallel processes all executing together, but all need to be stopped for the cleanup to begin. The Magic processing unit sends a message to all workers in the flow execution tree to abort their current tasks.

All of the workers attempt to abort their tasks. If a thread has still not aborted, the cleanup process will wait until the thread is clear. After all of the workers have aborted their task, the workers are free to take on new messages and a cleanup of the root flow will be carried out by the Magic processing unit. The cleanup mechanism invokes the **Cleanup recovery flow** defined in the flow properties of the root flow.

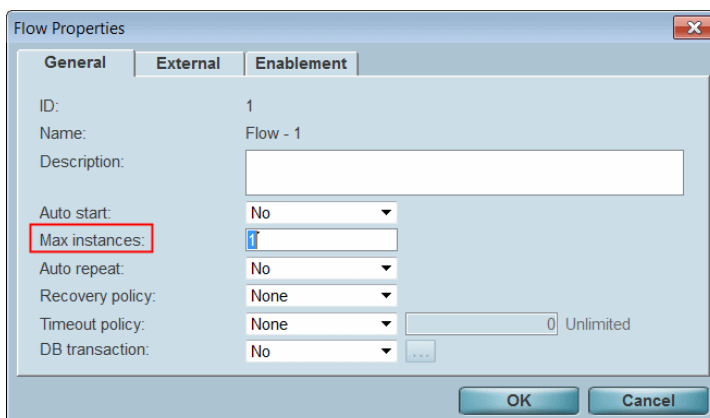


It will also release any resources locked by the **Lock Resource Service**.

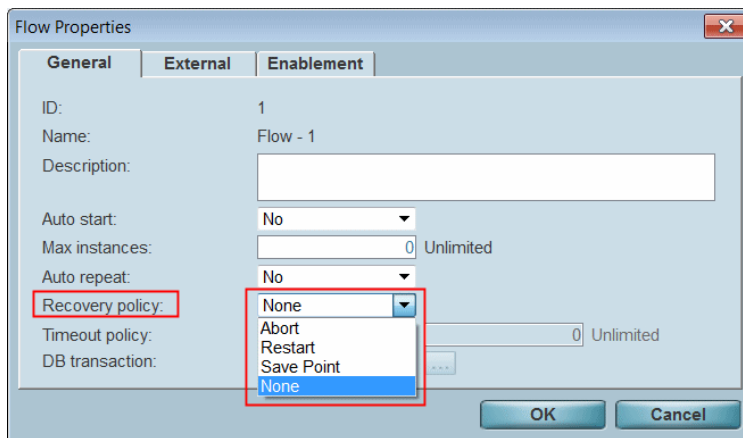


Lock Service

If the **Max instances** property was defined for this flow, the flow will be released so that another instance of the flow can be invoked.



When the cleanup has finished, the recovery process will start according to the recovery policy defined in the **Recover policy** of the flow.



If the policy is:

- **Abort**, no further action is required.
- **Restart**, the topmost (root) flow is restarted using the original data from the root flow. The Magic processing unit will change the message status to **READY_FOR_USE**, and this will be added to the pool of messages waiting to be handled.
- **Save Point**, the worker updates the Magic Space with the last Save Point or restarts the flow if no last Save Point exists or if that step was not reached. This is only relevant for the root flow's linear branch. During the recovery process, the Magic processing unit will update the message with **READY_FOR_USE** and add the specific step based on the Save Point details. If the Save Point was defined after parallel workers were invoked, the parallel processes will not be executed.
- **None**, there will be no attempt at recovery. Parallel tasks will not be stopped and will continue running as if there was no crash. Therefore, it is good practice to define a Recovery policy.

Monitoring


Magic xpi 4.x has enhanced monitoring functionality.



Magic xpi 4.1 introduces the Magic Monitor, which was introduced to make the deployment of your projects easier. You open this monitor via the Windows **Start** menu's **Magic Monitor** shortcut. The Magic Monitor lets you examine the project and see where you need to make any modifications to improve performance. For example, you can identify issues caused by heavy data loads or by a possible shortage of licenses. For the purpose of this course we will use the pre-V4.1 legacy monitor described below.

When you load the Monitor, the Monitor connects to the Magic Space and then displays the **Projects View**.

By parking on a project in the **Projects View**, such as the **SpaceCourse** project, you can click the **Start** button at the bottom of the screen to start the project. A **Start Project** option is also available from the context menu. The Monitor project startup process will be the same as if you clicked on the **Start** link of the project. The **Startup Mechanism** section above describes the steps. This is different from previous versions. In previous versions, if you launched the Magic xpi Server from the **Start** link, you were launching it from your own user credentials but if you started a server from the Monitor, you were launching the server using the user credentials of the broker process. In Magic xpi 4.x it is always the GSA that loads the Magic xpi server.

While the Magic xpi server is loading the project, you may see a warning icon  next to the project's status. This shows that there are currently no servers handling the project. The warning icon will appear whenever there are fewer servers running than there should be. For example, if you defined a different machine in the **start.xml** file's **<server>** section, and that machine is not available, the warning icon will appear.

From the **Projects View** you can see:

- The number of flow messages that have been handled by the project.
- The time elapsed since the project was executed.
- The number of servers currently running the project. If you park on the **Servers Running** column, you will get a tooltip showing the list of servers currently handling this project.
- The number of workers available for this project. Every time you load a server with workers, those workers are added to the pool.
- The reserved licenses for this project.
- The current number of licenses that are being used by this project.

Servers Running	Workers
1	10
LAWRENCEFWIN7 [pid: 8372]	

The Activity Log is opened by pressing the **Open** button or by right-clicking and selecting the **Open Project** option. This is the same view as in previous versions.

Advanced Monitoring Console

Magic xpi 4.x introduces the Advanced Monitoring Console, which provides you with additional runtime information about the Magic xpi project. You open it by selecting the **Advanced Monitoring Console** option from the Magic xpi Monitor's context menu or its **View** menu. You can have several Advanced Monitoring Console views open at the same time, each one monitoring a different project.

The Advanced Monitoring Console consists of three panes: **Navigation**, **Summary**, and **Details**. The **Navigation** pane displays the hierarchical structure of the triggers, servers, and flows in a project. The information displayed in the **Summary** and **Details** panes depends on which hierarchical level you stand on in the **Navigation** pane.

The **Summary** pane provides read-only information about the selected project, trigger, server, or flow, as well as licensing information.

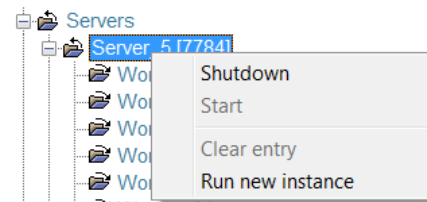
The **Details** pane provides in-depth runtime information about a specific project, trigger, server, or flow, and in some cases licensing information.

For detailed information about each of the panes, please see the **Magic xpi Help**.



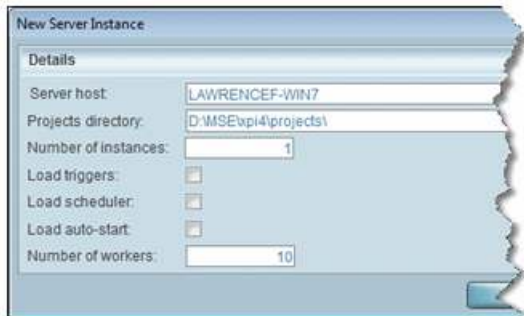
The data in the **Advanced Monitoring Console** is not refreshed automatically. For this purpose, a **Refresh** button is available on the lower left side.

Magic xpi enables you to execute operations on individual servers. By parking on a specific server node in the **Navigation** tree, you can access the context menu and perform one of the following actions:



- **Shutdown** – Shut down the current server. A dialog box allows you to set the timeout period for this to take effect. When you shut down a server, all workers are sent messages to continue until the timeout or if no messages are available. Each worker that finishes the flow will terminate and will not take new work. This option is available only when the selected server is currently running. When you shut down a server, the **Projects View** will display the warning icon ⚠ to indicate that not all servers are running.
- **Start** – Instruct the server to start, using the Magic server entity settings that were loaded earlier for this server. This option is available when the selected server has stopped or failed, or when the server's status is **START_FAILED**.
- **Clear entry** – Delete the server data from the Magic Space. This is also only available when the selected server has stopped or failed.

- **Run new instance** – Run another instance of this Magic xpi server according to the data defined in the server entity. (If you access this from the **Server** node, all of the fields are blank.) This is useful if you need to add more workers to the pool. By default, this dialog box displays data from the current server, the one you are parked on. In the example displayed below, you will see that the Load triggers, Load scheduler, and Load auto-start fields are not checked. This will add a single instance of the server and add 10 more workers to the pool, meaning more workers to handle the flow requests.

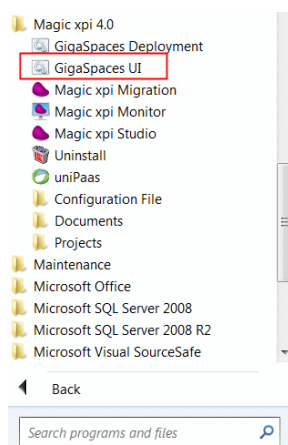


If you simply run a new server instance, all of the definitions defined in the original **start.xml** file are used, including whether to load the scheduler and triggers.

GigaSpaces User Interface Monitor

The GigaSpaces user interface (UI) includes a monitor, the **GigaSpaces Management Center**, which enables you to monitor the grid. In general there is little necessity to access this monitor, but you should be aware of what is available in the monitor.

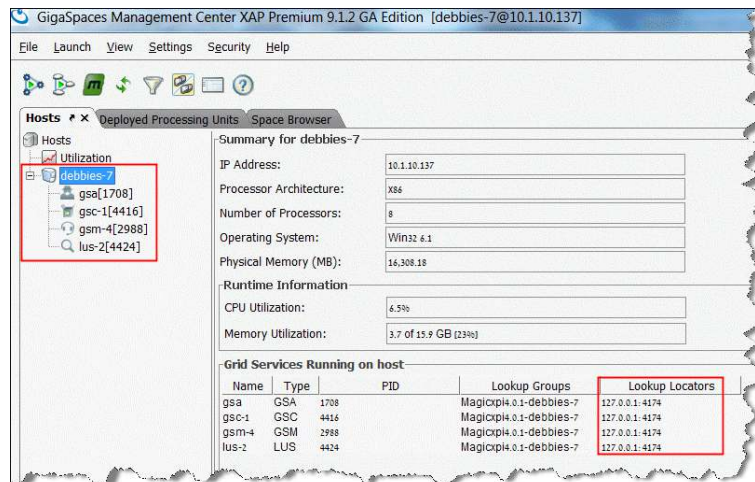
You load the monitor using the **GigaSpaces UI** shortcut link. **GigaSpaces Management Center**



When the monitor loads you will see three tabs: **Hosts**, **Deployed Processing Units**, and **Space Browser**. You can find additional information about this monitor in the GigaSpaces documentation.

Hosts Tab



If the grid has started properly, you should see your machine listed in the **Hosts** tab. This view shows all of the computers that are a part of this grid as well as the **Lookup Locators** defined during installation.

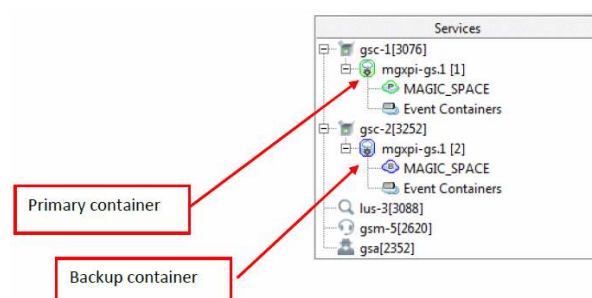


When you select the **Development Machine** check box during the installation (the default option), under the name, you should have one Grid Service Agent (GSA), one Grid Service Container (GSC), one Grid Service Manager (GSM), and one Lookup Service (LUS). In the Troubleshooting lesson, you'll find information about what to do if the grid has failed to start properly.

You can see each component that was loaded and its Process ID (PID) in the operating system, such as **gsa[1708]**, where **1708** is the PID. This number is useful because:

- It will appear as the **java.exe** entry's PID in the Task Manager.
- It is attached at the end of the log to differentiate one log from another. The log files will be discussed later on in this tutorial.

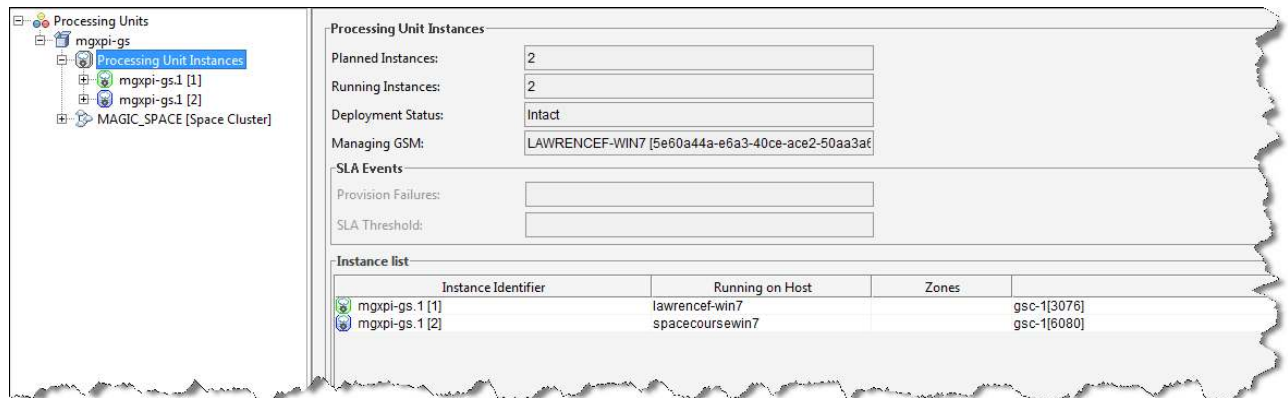
If you look in the **Services** pane, the primary GSC is identified by the green  icon (**P** for primary) and the backup GSC is identified by the blue icon  (**B** for backup). In the image below, they are both on the same computer, since this was the computer that loaded the Magic Space.



Deployed Processing Units Tab

By clicking the **Deployed Processing Units** tab, you can see where the partitions are deployed. This is similar to the information in the **Hosts** tab.

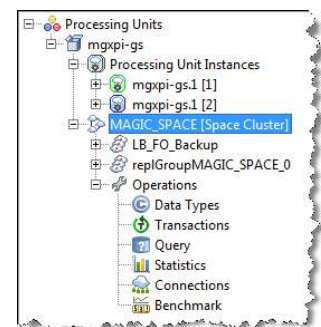
When you click on the **Processing Unit Instances** node, in the **Details** pane, you can see where each partition is deployed. You can identify which are the primary and backup partitions according to their colors.



Instance Identifier	Running on Host	Zones
mgxpi-gs.1 [1]	lawrencef-win7	gsc-1[3076]
mgxpi-gs.1 [2]	spacecoursewin7	gsc-1[6080]

By clicking on the **MAGIC_SPACE** node, you will receive more information. In the **Details** pane, you will also see the name of the Lookup Group that your cluster belongs to.

By opening that node, you will get even more information. When you open the **Operations** node, you will see other nodes. One of these is called **Data Types**.



In the **Data Types** list, you can see all of the system's data types, including those dealing with Magic xpi, such as the **ServerData** data type. The **ServerData** data type is the entity created when you start a project. You can park on a data type and select **Query** from the context menu to see more information about the data type.

Exercise

The goal of this exercise is to be able to monitor the process:

1. Define a **start.xml** file with 5 workers that do not load triggers, schedulers, or auto-run tasks. Open the Monitor and start the project. You will notice that all workers are idle.
2. Once the project is running, add a new server instance with 2 workers that loads the scheduler. You will notice that all 7 workers are now active.
3. Shut down all of the servers and clear the data from the Space.
4. Edit the **start.xml** file and add another server with 3 workers. This time, load the scheduler. Click the **Start** link. You can also run the Magic xpi server on another machine. You will notice that all 8 workers are active and running.

Summary

In this lesson, you learned how the system recovers from:

- The Magic xpi engine crash.
- A worker crash.
- A work process crash or failure.

You learned about the Monitor and the Advanced Monitoring Console, and how to load new instances of the server from the Monitor for load balancing purposes. You also learned about monitoring through the GigaSpaces user interface.



Magic®
University

Lesson 5

Deployment

Once you complete your project, you will want to give some thought to how the project is to be deployed. You can use the initial installation that Magic xpi provides, but there are some concepts that you should be aware of and settings that you may want to adjust.

This lesson covers various topics including:

- Space deployment
- Memory allocation
- Clustering
- Partitions and containers

Deploying the Magic Space



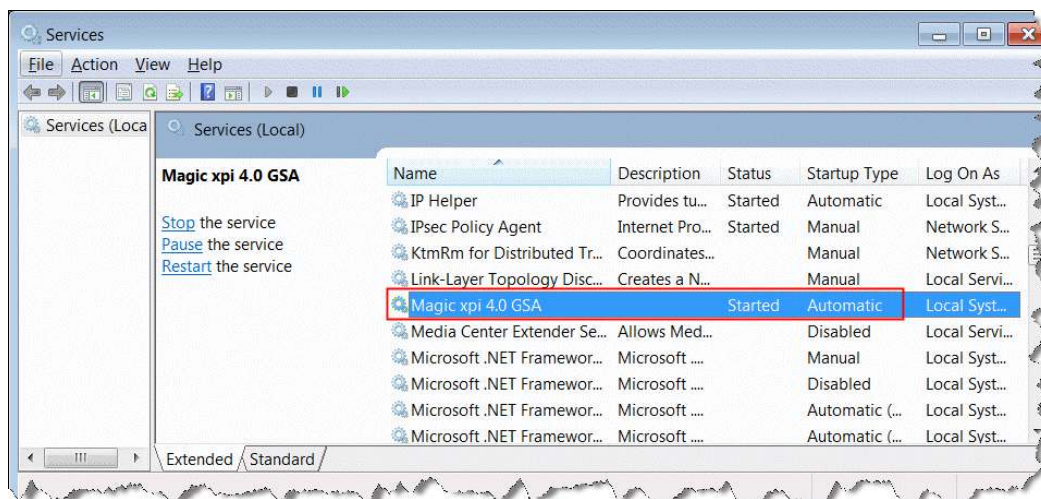
Remember that deployment is a one-time operation in a production environment.

Starting the Magic xpi GSA Service

If you select the **Install the Grid Service Agent (GSA) as a service** check box when installing Magic xpi, the **Magic xpi GSA** service should be running on your machine. This is the default when installing Magic xpi, so you should already have it on your machine.

If you want to check that it's up and running:

1. On your machine, from the **Start** menu, click **Run**.
2. In the **Run** dialog box, enter **services.msc**.
3. In the **Services** dialog box, look for **Magic xpi 4.0 GSA**. If it has a **Started** status and an **Automatic** startup type, then it's running.
4. If not, double-click on it and in the **Magic xpi 4.0 GSA Properties (Local Computer)** dialog box's **Startup type** parameter, select **Automatic**.
5. Click **OK** to finish.



When the **Magic xpi GSA** service starts on a specific machine, it loads the grid locally and searches the LAN for other grid components with the same Lookup Locator names. If any such components are found, the local grid is considered to be part of that Lookup Locator. In this way, a single unified grid is established in the network.

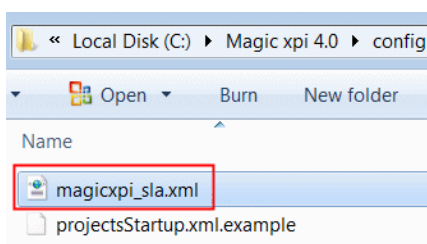


It's important to remember that the Magic xpi engines run under the user defined for the GSA service and not by the logged in user. By default, the user defined for the service is the **Local System account** as shown in the image below. For running Magic xpi on a single machine, this is usually fine. However, on a clustered environment, the service should run as a user who has privileges to access network resources.



Magicxpi_sla.xml

Once the grid is up and running, the **Magic xpi GSA** service automatically deploys the Magic Space on the grid. How the Magic Space is deployed on the grid, such as how many partitions and backups to use, is defined (by default) in the **magicxpi_sla.xml** configuration file, which is located at **<Magic xpi 4.x installation>\config**.



By default, this file defines two partitions with one backup each (four in total), and with a restriction that a primary partition and its backup partition cannot run under the same process.

The most common SLA settings in the **magicxpi_sla.xml** file are:

- **cluster-schema** – This should always be set to **partitioned-sync2backup**, which means that data can be in partitions and each partition can have a backup that is synchronized with it.
- **number-of-instances** – This refers to the number of partitions, meaning instances of the Magic processing unit, which will be loaded. The default is **1**. If you have a lot of data in memory, you may need to increase this number.

- **number-of-backups** – Here you define the number of backup partitions for each primary partition. During development you can decide that you do not need a backup and you can set this value to **0**. If the **number-of-instances="2"** and the **number-of-backups="1"**, there will be four instances of the Magic processing unit.
- **max-instances-per-vm** – This defines how many instances of the same partition will be deployed in the same JVM (GSC), that is, under the same process. If you left the default as is, **max-instances-per-vm="1"**, the primary and backup instances of the same partition will not be deployed on the same GSC.
- **max-instances-per-machine** – This defines how many instances of the same partition will be deployed on the same machine. When this is set to **1**, you ensure that a primary partition and its backup(s) cannot be provisioned to the same machine. Setting this to **1** should be restricted to a cluster containing a minimum of three machines. Then, if one of the machines fails, the lost partitions will move to the third machine. Or, it can also be used in a two machine cluster, but there is a risk having primary partitions with no backup until the second machine is back up and running.

Magicxpi-gs-agent.bat

The number of Grid Service Containers (GSCs) is defined in the **magicxpi-gs-agent.bat** file, which is located in the **<Magic xpi installation>\Gigaspaces\bin** directory.

In **magicxpi-gs-agent.bat** file, in the command starting with **call gs-agent.bat**, you should define the number of GSCs to match the number of required partitions by modifying the number next to the **gsa.gsc** parameter. To set the number of GSCs that will be loaded:

1. Open the **magicxpi-gs-agent.bat** file in a text editor, such as Notepad.
2. In the **call gs-agent.bat** line, set the **gsa.gsc** parameter to the number of required GSCs, such **gsa.gs 2**. In this example, the value **2** indicates the number of GSCs to load.

Partitions and Containers

Each Space partition runs within the Grid Service Container (GSC). If both the primary and the backup partitions exist within the same container, both will end if the GSC process exits abnormally. Therefore, if you decide that you want a backup you will need more than one GSC. In general, you need to load enough GSCs to host the number of partitions that you need, taking into account the backups.

You configure the GSC and the backups in the two files that were just discussed:

- **magicxpi-gs-agent.bat**
- **magicxpi_sla.xml**

Here are some examples of the **magicxpi-sla.xml** file:

1. For single partitions with two backups, and primary and backup partitions on separate GSCs, you'll set the following in the **magicxpi_sla.xml** file::

```
<os-sla:sla cluster-schema="partitioned-sync2backup" number-of-instances="1"
number-of-backups="2" max-instances-per-vm="1">
```

The above example requires at least **three** containers on a single machine. Each container will hold a single partition.



Using two backups is not recommended. This example is brought here to illustrate how the required number of GSCs is calculated.

2. For two partitions with one backup each, and primary and backup partitions on separate GSCs, you'll set the following in the **magicxpi_sla.xml** file:

```
<os-sla:sla cluster-schema="partitioned-sync2backup" number-of-instances="2"
number-of-backups="1" max-instances-per-vm="1">
```

The above example requires at least **two** containers on a single machine. Each container will hold two partitions.

3. For two partitions with one backup, and primary and backup partitions on separate machines, you'll set the following in the **magicxpi_sla.xml** file:

```
<os-sla:sla cluster-schema="partitioned-sync2backup" number-of-instances="2"
number-of-backups="1" max-instances-per-machine ="1">
```

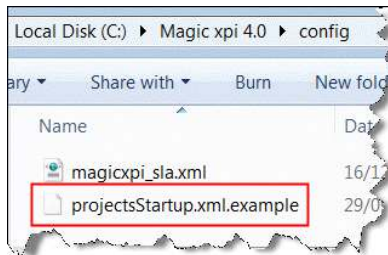
The above example requires at least **two machines** with at least **one** container on each machine. In each machine, the container will hold two partitions. If there is a cluster of two machines, and one of the machines fails, the Magic Space deployment will be incomplete (compromised) and no backup partition will replace the lost backup partitions until the failed machine starts up again.

Automatically Starting Projects

You can create a file called **projectsStartup.xml** file that tells Magic xpi to automatically start your projects and servers when the Magic xpi service deploys the Magic Space.

The structure of the **projectsStartup.xml** is identical to the structure of the **start.xml** created under each project.

1. Make a copy of the **start.xml** file or the **projectsStartup.xml.example** file, which is located in the **config** folder.



2. Make the changes that you want and save it to the **config** folder.

The startup sequence of the grid will then be as follows:

1. The Grid Service Agent (GSA) is started as a service on each application server that is part of the grid. The Grid Service Agent starts the core grid infrastructure to connect all application servers together.
2. The first GSA that starts successfully is responsible for deploying the Magic Space. This Magic Space contains all Magic-related objects that are shared across the grid.
3. The same GSA also deploys the **projectsStartup.xml** information into the Magic Space to enable projects to start automatically.
4. All GSAs in the grid query the **projectsStartup.xml** information in the Magic Space, and load the Magic xpi servers as defined.
5. For each project that is deployed, the first Magic xpi server that starts is responsible for initiating the project objects in the Magic Space to enable the projects to start.
6. For each project, a **start.xml** file in the same format is created automatically in the project's folder.

Spreading Out the Partitions

When the Magic Space is deployed, the Grid Service Manager (GSM) attempts to spread the partitions amongst the available containers (GSCs) in such a way that a single server failure will not affect the Magic Space operation and will not cause any data loss. This provisioning process is automatic, but once complete it will not rearrange itself.

If only one machine was running during the Magic Space deployment process, and there was no restriction in the SLA definition related to a single machine (**max-instances-per-machine**), this machine will hold all the partitions. Containers starting on other machines after the deployment was complete will not hold any Space partitions, and the single machine that is currently running the Magic Space is now considered a single point-of-failure.

When you have more than one machine that is part of the grid, you will want to have control over when the Magic Space is deployed. When the Grid Service Agent (GSA) loads, and the machine becomes a part of the grid, that machine will not host a part of the Magic Space if there is already a Magic Space deployed on the grid.

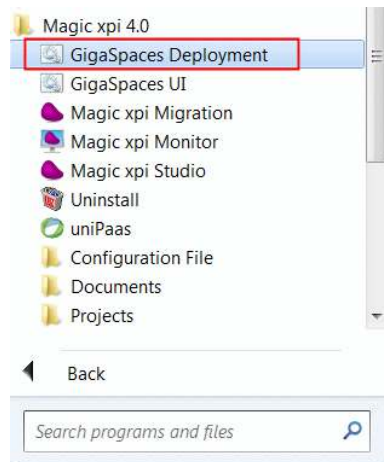
To spread the partitions over multiple machines when one machine holds all of the partitions, you have the following options:

1. Magic xpi can automatically monitor and rebalance the single points of failure of both primary and backup partitions running on the same host. Periodic checks are made to determine whether a rebalance of a partition's instance is required. This mechanism is controlled by the following two properties that are defined in the **mgdeploy.xml** file (located in the **<Magic xpi installation>\GigaSpaces\config\gsa** directory):
 - **rebalance-partitions** - When this property is set to **true** (default), or when it does not exist, the rebalancing mechanism is activated.
 - **rebalance-interval** - This property defines the intervals between rebalance checks. If the property does not exist, the default is **5** minutes.

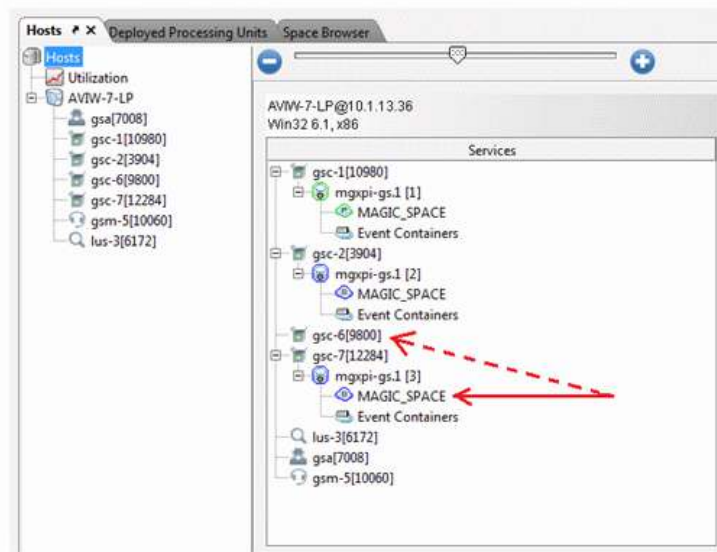
These properties appear in the **mgdeploy.xml** file as follows:

```
<argument>-rebalance-partitions</argument>  
<argument>true</argument>  
<argument>-rebalance-interval</argument>  
<argument>5</argument>
```

2. You can configure all of the services to avoid automatic Magic Space deployment and run the deployment batch once the whole grid is up, by first loading all of the GSAs and then deploying the Magic Space manually. You do this as follows.
 - a. In the **magicxpi-gs-agent.bat** file, set the **gsa.mgdeploy** entry to **0** (**gsa.mgdeploy 0**). Do this on all computers in the grid. Now, when you load the grid, all machines will be a part of the grid but the Magic Space will not be deployed.
 - b. Deploy the Magic Space by running the GigaSpaces Deployment shortcut or the **<Magic xpi installation>\Gigaspace\bin\Magicxpi_deploy.bat** file.



3. You can manually rearrange the partitions from the GigaSpaces UI. You do this as follows:
 - a. Open the Gigaspace UI **Hosts** tab.
 - b. Stand on the **Hosts** entry at the top of the hierarchy tree on the left.
 - c. In the Services pane, on the right side of the Gigaspace UI screen, you will see a tree of containers and partitions. You can now select a partition (either primary or backup) and drag it to a different container, as shown in the following image.

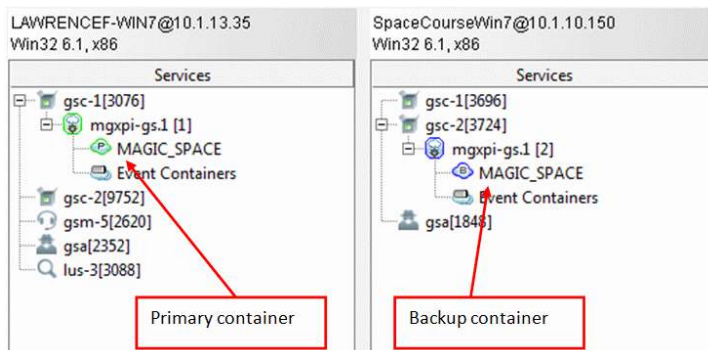


4. You can restart the backup GSC and GigaSpaces will provision the grid. You do this as follows:
 - a. Park on the GSC node of the backup partition.
 - b. From the context menu, select **Restart**.

GigaSpaces will attempt to place the backup container on the second computer, as you can see from the image below. This provides redundancy for your application.

If the secondary machine is not available, GigaSpaces will create the backup partition on the

current machine. When the secondary machine becomes available again, GigaSpaces may not automatically reposition the backup on the secondary computer. You may need to perform the operation manually.



5. You can use the **max-instances-per-machine** restriction in the SLA. This method should be restricted to a cluster of at least three machines, which ensures that at least two machines in the grid will run the Space partitions.
 - a. In the **magicxpi_sla.xml** file, set the **max-instances-per-machine** to 1 as explained earlier in this lesson.
 - b. When the automatic deployment process starts, it will not be completed until at least two machines are hosting the Space partitions.

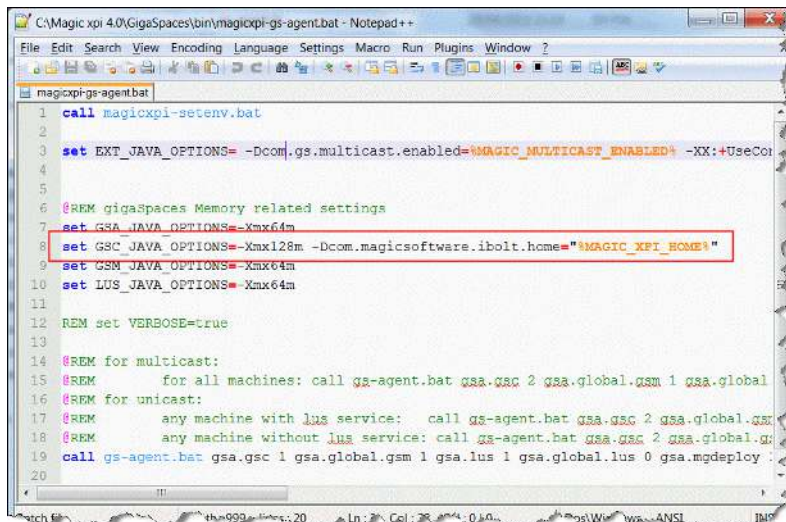
Memory Allocation

Each Magic xpi and grid component uses memory. How you define the memory to allocate to each component depends on what your project does. Memory allocation for the various GigaSpaces entities is determined in the **magicxpi-gs-agent.bat** file.

Grid Service Container (GSC)

When a trigger or a worker posts a flow invocation message into the Space, they are placing the message along with the message payload into one of the space partitions that is hosted on a GSC (which is a Java process). Therefore if you have many triggers, polling or external, that place heavy BLOBs into the Space, the GSC memory must be able to contain it.

If you encounter any memory-related issues with the GSC, consider changing the size to at least **512MB**. In the **magicxpi-gs-agent.bat** file, there is a **gigaSpaces Memory related settings** section. Within that section, find the **set GSC_JAVA_OPTIONS=** line (marked below in red). Change the number after the letters **Xmx**.



```

1 call magicxpi-setenv.bat
2
3 set EXT_JAVA_OPTIONS=-Dcom.gs.multicast.enabled=%MAGIC_MULTICAST_ENABLED% -XX:+UseCo
4
5
6 @REM gigaspaces Memory related settings
7 set GSA_JAVA_OPTIONS=-Xmx64m
8 set GSC_JAVA_OPTIONS=-Xmx128m -Dcom.magicsoftware.ibolt.home=\"%MAGIC_XPI_HOME%\"
9 set GSM_JAVA_OPTIONS=-Xmx64m
10 set LUS_JAVA_OPTIONS=-Xmx64m
11
12 REM set VERBOSE=true
13
14 @REM for multicast:
15 @REM     for all machines: call gs-agent.bat gsa.gsc 2 gsa.global.gsm 1 gsa.global
16 @REM for unicast:
17 @REM     any machine with lus service: call gs-agent.bat gsa.gsc 2 gsa.global.g
18 @REM     any machine without lus service: call gs-agent.bat gsa.gsc 2 gsa.global.g
19 call gs-agent.bat gsa.gsc 1 gsa.global.gsm 1 gsa.lus 1 gsa.global.lus 0 gsa.mqdeploy
20

```

So in this example, you would change **Xmx128m** to **Xmx512m**.

The GSA, GSM, and LUS entities have quite a small memory footprint, so you can leave these settings as is.

Clustering

In Space-based architecture, as you learned in the previous lesson, the Magic Space can reside on several computers that are viewed as a single logical unit. A cluster can be seen as a single "large" Space.


Space clustering defines the number of Space partitions, the number of partition backups, and the way they are spread across the available grid containers (GSCs).

Space clustering is governed by the SLA definitions set in the **magicxpi_sla.xml** file. This means that the grid will always try to maintain the defined clustering when deploying the Magic Space.

Adding Multiple Server Instances

To provide more resources for your project, you often want to run the same project on more than one server and in some cases you want to run the same project on different machines.

One of the methods to run more than one instance of the same server on a specific computer is simply to update the **NumberOfInstances** node in the **start.xml** configuration file. Remember that only updating the configuration file does not have any effect on what is already running.

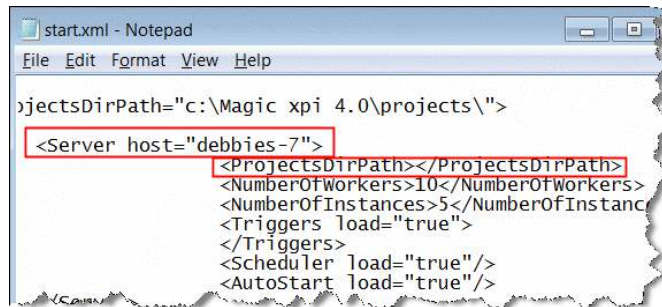
You need to click the  **Start** link again. This will load the additional entities that you defined (in the **NumberOfInstances** node) into the Magic Space.

For example, if you had the **NumberOfInstances** node set to **1** the first time you clicked the **Start** link and changed it to **3**, when you click the **Start** link again, you will see **4** instances of the server running in the Advanced Monitoring Console. Every time you click the **Start** link, the

entire configuration file will be read and loaded into the Space, not just a specific section.

To run the same project on different machines (multiple instances), you simply duplicate the `<server>` section in the `start.xml` file. In this case, you need to ensure that:

- The **Server host** that is defined in the server node is valid.
- The **ProjectsDirPath** value points to a path that is accessible on the second server. You can do this by either having the project on a network path that is accessible to all, or you can copy the project to the second server and update the **ProjectsDirPath** node.



```

start.xml - Notepad
File Edit Format View Help

ProjectsDirPath="c:\Magic xpi 4.0\projects\"
<Server host="debbies-7">
  <ProjectsDirPath></ProjectsDirPath>
  <NumberOfWorkers>10</NumberOfWorkers>
  <NumberOfInstances>5</NumberOfInstances>
  <Triggers load="true">
  </Triggers>
  <Scheduler load="true"/>
  <AutoStart load="true"/>

```



To define a specific server that will listen for triggers, you can duplicate the `<server>` section with `<Triggers load="true"/>` for one server and for the other servers you can have `<Triggers load="false"/>`.

Installing the Server License

In Magic xpi 4.x clustered environments, you only need a single host locked license for the entire cluster.

Each project needs to load with access to the `license.dat` file:

1. Copy the license file that you received after purchase, and copy it to all of the servers, or to the shared location. It is good practice to put the license file on a path where all servers will have access to it so that each server does not need its own license file
2. Modify the `Magic.ini` file with the license file location by updating the `[MAGIC_ENV] LicenseFile` entry to point to the shared license file.
For example: `LicenseFile = \\10.1.1.6\licenses\License.dat`
3. Make sure that your project's `ifs.ini` file is configured with the production license, such as `[MAGIC_ENV]LicenseName = IBPRSRVI` for a Windows machine.



Aside from the server hosting the license according to the license's `HOSTID` flag, any other server that loads will simply add a license entry into the Magic Space but will not add an actual license. The license count will therefore be zero.

Projects Folder Location

Deploying a project involves copying each project's folder from the development/staging environment to the designated shared folder location.



It is good practice to have projects that are shared by all servers on a shared location such as `\\SPACECOURSE-WIN7\projects\`.

Although a shared folder is potentially a single point-of-failure, with today's storage systems, this type of resource is usually highly redundant by itself.

An alternative, less recommended approach, is to deploy a copy of the Magic xpi projects to each of the application servers. This eliminates even the theoretical single point-of-failure, but at the expense of high maintenance costs and complexity.

Installation Settings

On each application server, when you run the setup program (**setup.exe**) from the installation media files, there are certain settings to be aware of.

1. When installing the first database server, select **Now**. Enter your database server and the Magic xpi administrator DB user.
2. For all other database servers, select **Later**.
3. On only one of the hosts in the cluster, select the **Install the Magic Monitor Services** check box. This will install the Magic Monitor services on the host machine.
4. On all application server setups, clear the **Install Web Service Framework** check box. You will install this separately on the front-end Web server.
5. In the new **Magic Service Configuration** screen, for all of the application servers:
 - a. Select the **Install the Grid Service Agent (GSA) as a service** check box.
 - b. Clear the **Development Machine** check box. This needs to be cleared to deploy the Magic Space in a clustered environment.
 - c. In the **Use the following locators (Unicast)** field, enter the addresses (comma separated) of the application servers that you designated as LUS servers.
 - d. In the **Number of GSCs** field, enter the number of GSCs that you want to deploy. This should be set to at least **2**.
 - e. In the **GSC memory allocation** field, enter at least **512**.
 - f. On the two application servers that you designed as LUS servers (the same two defined as the locators), select the **Run the LUS** check box.
Note: Make sure that you only select this check box for two of the application servers.
 - g. In the **Number of partitions** field, enter the number of partitions that you want your Magic Space to contain, such as **2**.
 - h. Select the **Partition backup** check box. It is always recommended to work with a backup.

6. For all of your machines, on the last screen, select the **Start the Magic xpi Services** check box. This will start the Magic Monitor Display Server, the Magic Monitor Web Server, and the Magic 4.x GSA.



For additional information about setting up a clustered information, such as permissions and network access, please see the **Magic xpi 4 x - Advanced Deployment Guide.pdf** file included with the Magic xpi installation. This information is also available on Magic Software Enterprises' DevNet at: http://devnet.magicsoftware.com/en/library?book=en/Magicxpi4/&page=Magic_xpi_4.x_Deployment.htm.

Configuring Multiple Network Cards (Optional)

If your application servers have multiple network cards, configure the use of a specific card for the Magic xpi 4.x server, as follows:

1. Modify the NIC_ADDR value found in the **<Magic xpi installation>\GigaSpaces\bin\magicxpi-setenv.bat** file to hold either the IP assigned to this network card or the name of the network card itself.
For example: NIC_ADDR=10.1.1.11 or NIC_ADDR="#eth0:ip#", where **eth0** is the name of the network card.

Tip: You can find the network cards' IPs and names by running the following script:
<Magic xpi installation folder>\GigaSpaces\bin\platform-info.bat -verbose
and look for the IPs and names in the **Network Interfaces Information** section.

2. Add the same NIC_ADDR value to the Magic.ini file's jvm_args section:
-Djava.rmi.server.hostname=<your network card IP address here>




The host name or IP address should **not** be surrounded by quotation marks.

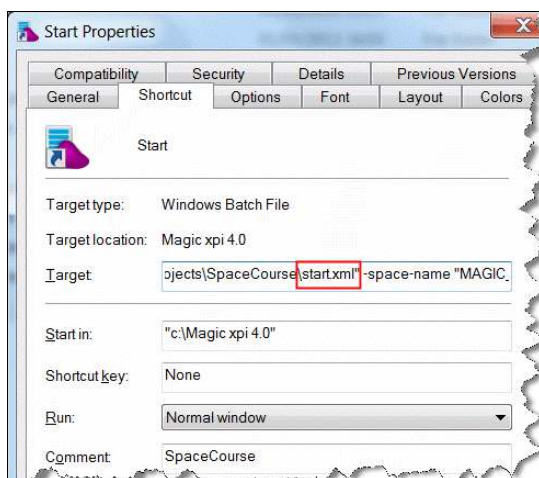
Starting Projects from the Command Line

This section is relevant if you want to change the default of the location from where you want to start the project.

Projects can be started from the command line. To start a project from the command line:

1. Go the  **Start** link in the project's folder.
2. Right-click and select the **Properties** option.
3. Find the **Target** field. The text in the field will look something like this:

```
"D:\Magic xpi\Magic xpi 4 GS 11_12\MgxpCmdl.bat" start-servers -startup-config-
file "D:\Magic xpi\Magic xpi 4 GS 11_12\projects\Project1\start.xml" -Space-name
"MAGIC_SPACE" -group "Magicxpi-4.0.0_AVIW-7-LP" -locators ""
```



4. This path points to a specific **start.xml** file that is residing under the project. Change the path to load a different **start.xml** file whose configurations can load several projects.
5. Click **OK**.

Starting and Stopping Projects from the Monitor

This section is relevant if you want to change the default of the location from where you want to start or stop the project.

The Monitor shows the projects based on the **[MAGIC_IBOLT]MPSPProjects** entry in the **ifm.ini** file, and runs the projects based on the **%projects%** environment variable. Follow these steps:

1. Modify the **[MAGIC_IBOLT]MPSPProjects** entry to point to the shared drive, such as **\\10.1.1.6\projects**.
2. In the **ifm.ini** file, create a **[MAGIC_LOGICAL_NAMES]projects** entry and point to the same shared drive.
3. Restart the Monitor to apply the settings.

For the new Magic Monitor, this is done using the **ApplicationsList.xml** file. For more information, see the Magic xpi Help file.

Exercise

1. Change the memory allocation of the GSC to 512 MB.
2. Try to manually rearrange the partitions from the GigaSpaces UI.

Summary

In this lesson, you learned how to control the GigaSpaces installation, which was installed during the Magic xpi installation process.

You learned about text files that are responsible for the way that GigaSpaces loads:

- The **magicxpi-setenv.bat** file found at <Magic xpi installation>\GigaSpaces\bin.
- The **magicxpi-gs-agent.bat** file found at <Magic xpi installation>\GigaSpaces\bin.
- The **magicxpi-sla.xml** file found at <Magic xpi installation>\config.

With these files you can control:

- The memory consumed by each component.
- How to deploy the Magic Space manually.
- How to define Space clustering (backups and partitions).



Magic®
University

Lesson 6

Troubleshooting

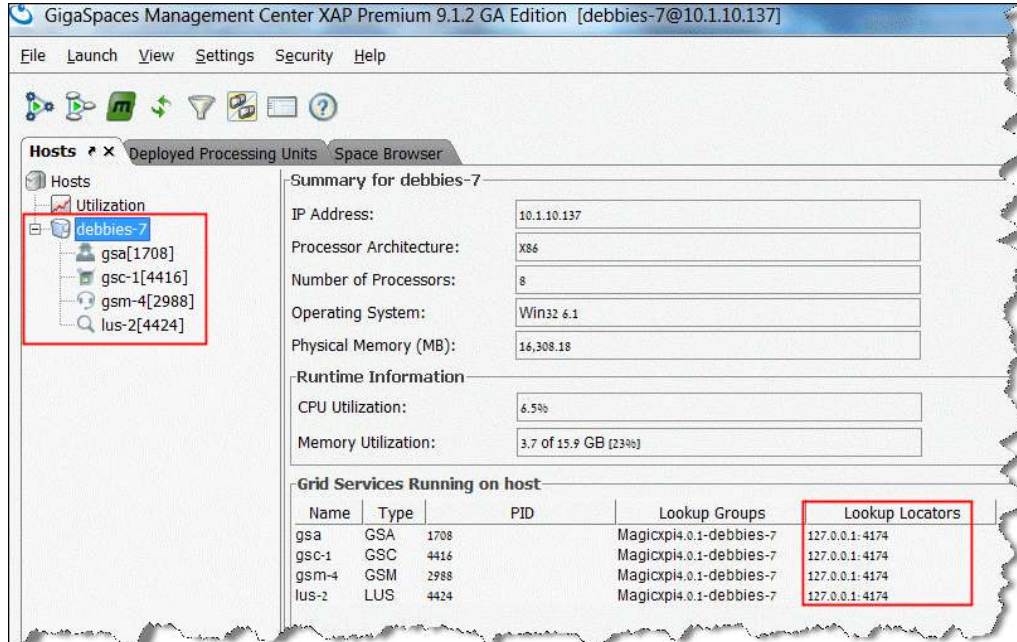
Troubleshooting is part of the life cycle of any project. It's important to have an understanding of the potential causes of problems and how to solve them.

This lesson covers various topics including:

- GigaSpaces UI
- Log files
- Project startup issues

GigaSpaces UI Troubleshooting

As was described earlier, if the grid started properly, you should see a grid listed in the **Hosts** tab and the grid components with the **Lookup Locator** defined during the installation.



The screenshot shows the GigaSpaces Management Center interface. The 'Hosts' tab is active, displaying a tree view of the grid 'debbies-7' with its components: gsa[1708], gsc-1[4416], gsm-4[2988], and lus-2[4424]. The right-hand pane shows a 'Summary for debbies-7' with the following details:

- IP Address: 10.1.10.137
- Processor Architecture: X86
- Number of Processors: 8
- Operating System: Win32 6.1
- Physical Memory (MB): 16,308.18
- Runtime Information:
 - CPU Utilization: 6.5%
 - Memory Utilization: 3.7 of 15.9 GB (23%)

Below the summary is a table titled 'Grid Services Running on host':

Name	Type	PID	Lookup Groups	Lookup Locators
gsa	GSA	1708	Magicxpi4.0.1-debbies-7	127.0.0.1:4174
gsc-1	GSC	4416	Magicxpi4.0.1-debbies-7	127.0.0.1:4174
gsm-4	GSM	2988	Magicxpi4.0.1-debbies-7	127.0.0.1:4174
lus-2	LUS	4424	Magicxpi4.0.1-debbies-7	127.0.0.1:4174

You may encounter one of the following issues:

- You will not see the grid listed in the **Hosts** tab. In the example above, you wouldn't see **debbies-7**. This means that the grid did not deploy.
- You will not see the **gsa**, **gsc**, **gsm** and **lus** entries under the grid name. This means that one or more of the Grid components did not load properly.

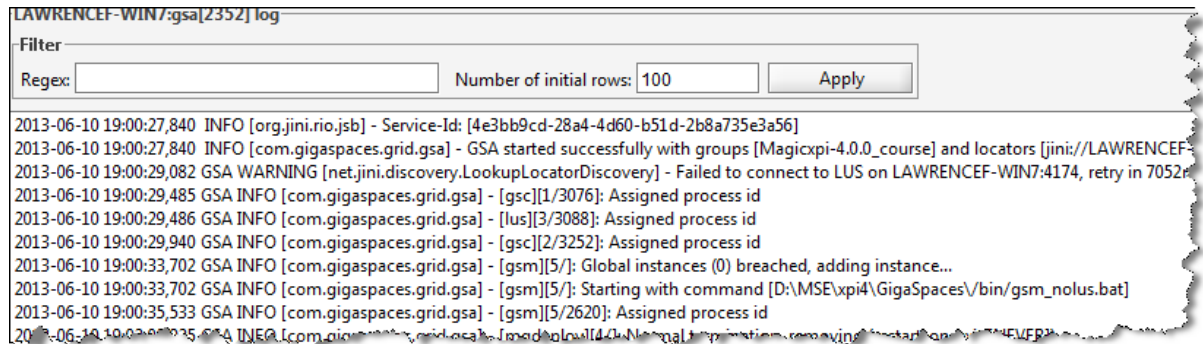


The first thing you should do when the grid is not working properly is to check that the **LOOKUPLOCATORS** value in the **magicxpi-setenv.bat** file is set correctly.

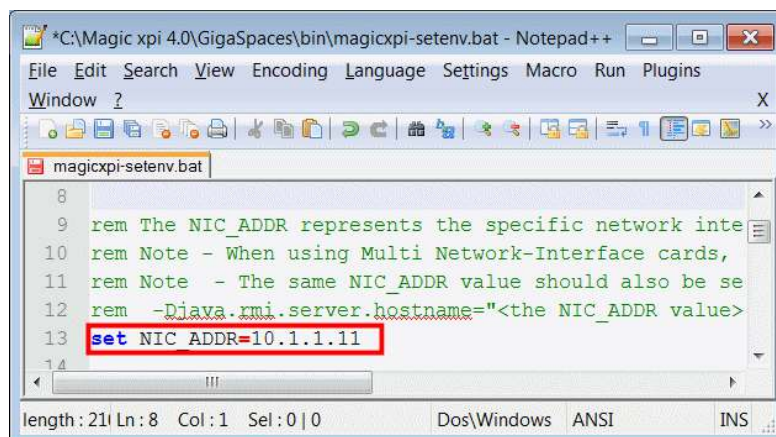
If you encounter one of these issues, you can also check the following:

- Try stopping the **Magic xpi GSA** service, waiting for all grid processes (if they exist) to terminate, and restarting the **Magic xpi GSA** service.
- If you did not select the **Install the Grid Service Agent (GSA) as a service** check box during installation, the grid will not deploy. Run the **installService.bat** file from **OS_Service\scripts** to install the GSA. With some operating systems, such as Windows 7 and above, you need to run this command using administrator credentials.
- If you did not select the **Run the LUS** check box during installation, the grid will not deploy. Define **dsa.lus 1** in the **magicxpi-gs-agent.bat** file.

- Click on one of the running components and you will see the log for that component. Any errors will be seen in the log, as shown in the image below:



- For all of the application servers, if your machine is running multiple network interfaces, make sure that the **NIC_ADDR** value (in the **<Magic xpi installation>\GigaSpaces\bin\magicxpi-setenv.bat** file) is set to hold either the IP assigned to the network card or the name of the network card itself. Then, add the **NIC_ADDR** value to the **magic.ini** file's **jvm_args** section:
-Djava.rmi.server.hostname=<your network card IP address here>
Note: The host name or IP address should not be surrounded by quotation marks.



- When working in a clustered environment, if the grid entities are not available in the GigaSpaces UI, check that the firewall is not blocking the ports used by GigaSpaces. There are two settings that control the ports:
 - The Discovery port, which should be opened in the firewall.
 - The range of LRMI ports, which should be set to a fixed range and also opened in the firewall.

Additional information about the ports can be found in the **Magic xpi 4 x - Advanced Deployment Guide.pdf** file in the Magic xpi help folder.

Log Files

Information about errors that occurred during runtime is logged to several log files and in several locations.

Magic xpi uses the **log4j** infrastructure for logging and the logging configuration is set in the **<Magic xpi installation>/java/classes/log4j.xml** file. The logs defined in the **log4j.xml** file are written by default to the **<Magic xpi installation>/logs/java** folder.

The logging of code related to GigaSpaces logs (**general_[PID].log** , **magicxpi_[PID].log** and **magicxpi-external_[PID].log**), the logging level can be changed in the **log4j.xml** while running and does not require an application restart.

You'll find a log file specific to the GigaSpaces infrastructure in the **<Magic xpi installation>/GigaSpaces/Logs** folder.

Project Startup Troubleshooting

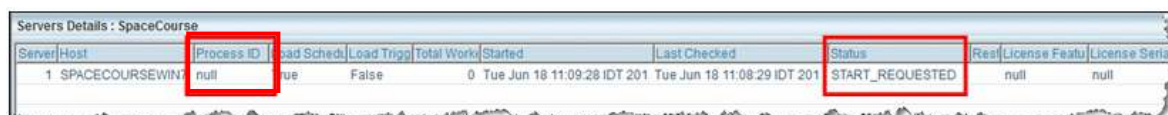
The following describe scenarios that might require troubleshooting during the startup process.

No Server in the Monitor

After starting a server from the Monitor or from the **Start** link, the server entry should appear in the Advanced Monitoring Console. If it does not appear, then no server entity (**ServerData** object) was created. To see why there was a problem creating an entity in the Magic Space or accessing the grid, check the **mgxpimdl.log** file on the machine where you requested the operation, and not on the remote machine.

Server Status Remains START_REQUESTED

After selecting **Start** from the link or from the Monitor, you initially receive a **START_REQUESTED** status.

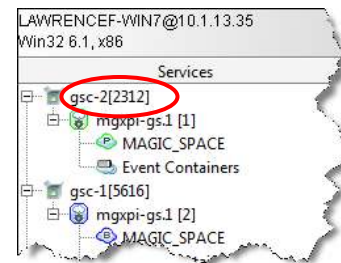


Server	Host	Process ID	Load Sched	Load Trigg	Total Work	Started	Last Checked	Status	Real License Featu	License Serial
1	SPACECOURSEWIN7	null	ue	False	0	Tue Jun 18 11:09:28 IDT 201	Tue Jun 18 11:08:29 IDT 201	START_REQUESTED	null	null

This is a temporary status and you expect that the status will change to **RUNNING**.

In the image above, the host machine (**SPACECOURSEWIN7**) has a status of **START_REQUESTED** and did not receive a Process ID, which you can see in the **Process ID** column where the value is set to **null**. To try and figure out why this happened, you need to find the relevant log. Remember that all logs have the name of the log and the process ID number:

1. In the <Magic xpi installation directory>\logs and <Magic xpi installation directory>\logs\java folders, try to delete all of the logs. The logs that you cannot delete are the ones currently being accessed by processes. These logs are the ones that you want to have a look at.
2. Load the GigaSpaces UI by clicking the link on the desktop or via the **Start** menu.
3. Look for the PID of the primary GSC, as in the image to the right. In this case, the Process ID is **2312**.
4. Open the log: `magicxpi_<PID>.log` located in the <Magic xpi installation directory>\logs\java folder (in this example, `magicxpi_2312.log`). You will notice that the Magic processing unit did not find the GSA on the host machine:



```
[[Project: SpaceCourse],NAME: SPACECOURSEWIN7,(ID: 1),(PROCESS:null),(GSA:null),(STATUS: START_REQUESTED)] (PID: 2312;)
013-06-18 11:08:30,294 [INFO]:(magicxpi-pu): admin.locateAgent - looking for host: SPACECOURSEWIN7 (PID: 2312;)
013-06-18 11:08:30,294 [WARN]:(magicxpi-pu): admin.locateAgent - no GSA found for host: SPACECOURSEWIN7 (PID: 2312;)
013-06-18 11:08:30,409 [INFO]:(magicxpi-spaceservices-global): LogModule magicxpi-spaceservices-global exist. Mexpi-logging.jar is up-to-date (PID: 2312;)
```

There are a number of reasons that this may happen, such as:

- This scenario may be valid, since you may have decided that the host machine will only be available at peak hours. In this case, you can simply wait until the machine becomes available.
- There is a problem with the host machine.
- There is a problem with network access to that machine. In other words, the machine is not available.
- There is a problem with the naming. The names of the host in the `start.xml` file may not match the name seen by the grid because of Domain Name System (DNS) resolution. Use the IP address of the machine instead of the name in the `start.xml` file to solve this issue.

Server Status Becomes START_IN_PROGRESS

The Magic processing unit has found the remote computer, and the GSA has managed to run the Magic xpi server with the parameters. The Magic processing unit updates the server status with **START_IN_PROGRESS**.

After an internal timeout that involves a number of checks to see if the `serverdata` entity's status is still **START_IN_PROGRESS**, the Magic xpi server could not connect to the Magic Space and the Magic processing unit then updates the server's status as **START_FAILED**.

Host	Proces	Load S	Load	Total	Started	Last Checked	Status	License F	License Serial	Consumed license
debbies-7	9424	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:55 IST	START_FAILED	null	null	0 (0, 0)
debbies-7	5940	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:55 IST	START_FAILED	null	null	0 (0, 0)
debbies-7	1984	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:55 IST	START_FAILED	null	null	0 (0, 0)
debbies-7	2124	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:55 IST	START_FAILED	null	null	0 (0, 0)

There are a number of reasons that this might happen, such as:

- A license problem.
- One of the parameters in the command line is incorrect, such as the project name or path.
- The user does not have access to the shared drive where the project exists. Remember that the Magic processing unit requests that the GSA on a remote machine load the Magic xpi server. Therefore, the Magic xpi server loads with the permissions of the GSA and not necessarily the permissions of the current user. The current user may have access to the project on the shared folder, as well as the license, but the user assigned to the GSA process may not have such access.

The errors above may appear in the **ifs.log** file. It is good practice to check in the **ifs.log** file for relevant errors.



The following are indications that there is a permission issue in accessing the project's folder:

- You do not have an **ifs.log** file.
- In the Windows Task Manager, the **MgxpiServer.exe** process consumes a small amount of memory (about 18MB).

Server Status Becomes SERVER_INITIALIZING

This is a temporary status in which the GSA has managed to run the Magic xpi server. The Magic xpi server updates the server status as **SERVER_INITIALIZING**.

Servers Details : SpaceCol											
Host	Process	Load S	Load	Total	Started	Last Checked	Status	License F	License Serial	Consumed license	
debbies-7	9424	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:46 IST	SERVER_INITIALIZING	null	null	0 (0, 0)	
debbies-7	5940	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:46 IST	SERVER_INITIALIZING	null	null	0 (0, 0)	
debbies-7	1984	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:46 IST	SERVER_INITIALIZING	null	null	0 (0, 0)	
debbies-7	2124	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:46 IST	SERVER_INITIALIZING	null	null	0 (0, 0)	
debbies-7	8700	True	True	0	Wed Jan 08 10:39:46 IST	Wed Jan 08 10:39:46 IST	SERVER_INITIALIZING	null	null	0 (0, 0)	

The Magic processing unit checks the server status to see if it has been updated to **RUNNING**. If it is not **RUNNING**, then after an internal timeout the Magic processing unit will update the server's status to **START_FAILED**. In some cases, this means that the actual Magic xpi server application and the project metadata were not loaded into the Magic xpi process. In other cases, such as database connectivity issues, this means that the project was created in the Space, but the engine will terminate later because of the database issue.

This occurs when the Magic xpi server could not access the database.

1. Open the **<Magic xpi installation directory>\logs** folder on the machine registered in the **start.xml** (in this case, **SPACECOURSEWIN7**), and look for the log for the project, such as **SpaceCourse_error.log**.
2. Open the log and see if there is an error there. If there was no access to the database, you might see an error such as: **Failed to open, data source: ifs_ods**.

Exercise

1. Find the backup partition and manually kill the process. Check that GigaSpaces manages to recreate a partition.
2. Load a server, from the link or by any other method.
3. Start a new server or instance with a non-existing machine.
4. Check the logs to find the reason that the server will not load.
5. Work with a colleague or a secondary computer:
 - a. Create an environment where there is a Lookup Service (LUS) running on each of the machines (this is the default).
 - b. Make sure that the Lookup Locator value contains the IPs of both machines running the LUS.
 - c. Configure only one of the machines to deploy the Space.
 - d. Start servers on both machines where only one of them loads the Scheduler.

Summary

In this lesson, you learned how to troubleshoot your project. You learned how to use the GigaSpaces UI and how to fetch the process ID (PID) of each component to assist you in finding the relevant log that you may need.

Remember that before troubleshooting it is a good idea to delete older logs so that you can quickly find the log that you are looking for.

You learned how to find the errors that prevented the server from being updated with the **RUNNING** status.

Glossary

Magic xpi Space-related Terminology

This section describes the new Space-related terminology, some of which was mentioned in the previous lesson.

Magic Processing Unit

The Magic processing unit runs in the Magic Space and has direct access to all the data in the Magic Space. The Magic xpi processing unit performs administrative tasks such as the startup of engines, monitoring, and recovery. All of the Magic xpi processing units together form the logical Space.

Magic Space

The Magic Space stores data objects in memory (similar to a database instance) and runs on the data grid. Magic xpi uses a single Space for running multiple projects. For redundancy and scalability, data and business logic in the Magic Space are replicated and partitioned across all machines participating in the data grid. This ensures continuous service even if there is a problem such as machine or software failure.

Worker

In Magic xpi, the workers are capable of executing any integration project logic. The worker is a generic thread that knows how to run any flow in the project. It performs the flow steps. These workers lie in wait for flow invocation requests, sometimes called messages. Once a flow invocation request is created, a flow worker executes the requested flow.

Work Process

A work process refers to the execution tree of a root flow, or a set of flows, comprising a single business transaction. The role of a work process is to process requests.

Magic xpi Project

The Magic xpi project is software composed of multiple server processes that execute Magic xpi 4.x integration project logic. Each Magic xpi server process, known as the *engine*, consists of multiple threads or workers. Each worker is capable of executing any integration project logic. It is possible to run a few servers for every project. Servers can be added and removed on demand while the Magic xpi project is running.

Space Partition

The Space partition contains the Magic xpi processing unit and an embedded space. Data loaded into the space is divided amongst the partitions. The partitions allow large amounts of data to be stored in a single Magic Space process memory with each Space partition containing a subset of the information. The combination of all the Space partitions within the same cluster creates the full set of information.

In-Memory Data Grid Terminology

Grid Service Agent (GSA)

The Grid Service Agent (GSA) runs on a specific machine and is responsible for adding that machine to the grid. Once the grid is up and running, the GSA will deploy the Magic processing unit and the Magic Space. It is also responsible for making sure that the processing units running in the grid are accessible, meaning that they have not crashed. If the processing unit is not running, the GSA restarts it automatically.

Grid Service Container (GSC)

The Grid Service Container (GSC) is a container that hosts processing units or Space partitions. It also hosts the data that this processing unit can access. It is common to start several GSCs on the same physical machine, depending on the machine CPU and memory resources. The GSC is normally loaded by the GSA, although it can be loaded on its own. Remember, however, that if it is loaded by the GSA, the GSA is responsible for ensuring that it is always available.

Grid Service Manager (GSM)

The Grid Service Manager (GSM) is the service that manages the Grid Service Containers. The GSM deploys processing units to the grid, and can also undeploy the processing units. The GSM monitors the processing units running within each GSC. For each processing unit, a fault detection mechanism is maintained by periodically pinging the processing unit. After multiple failures, the GSM tries to remove the instance and re-instantiate it on another GSC. The GSM is normally loaded by the Grid Service Agent, although it can be loaded on its own.

Lookup Service (LUS)

The Lookup Service (LUS) is a registry that contains information as to where each loaded component is actually situated. As an example, the LUS knows which computer a specific processing unit is running on. When a GSA, GSM, GSC, or processing unit loads, it registers itself with the LUS. Clients accessing the grid can find the information they are looking for by first accessing the LUS.