**Enterprise Application Integration**

09

**Application Integration with Message Queues**

**Assignment #4**

**Carlos Simões  
Miguel Graça Oliveira  
Pedro Saraiva**

# Introduction

In the scope of the project “Application Integration with Message Queues”, from the assignment #4 of the Enterprise Application Integration course, a set of design and implementation decisions had to be made. The goal of this report is to describe these decisions.

In addition, the report provides the instructions for deploying and executing the application.

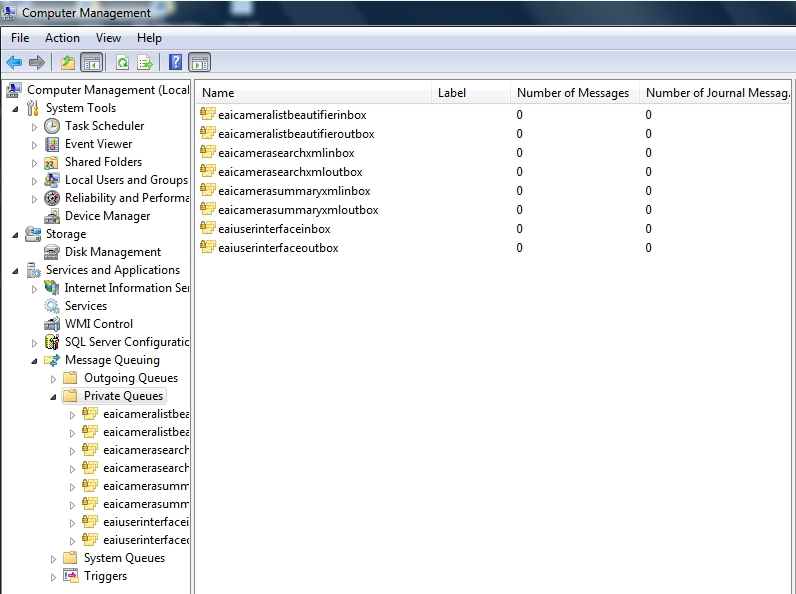
## Time Spent

|  |  |  |
| --- | --- | --- |
| Student | Mostly involved in | Time spent |
| Carlos Simões | * User Interface application * Wrappers * My Orchestrator | Reading 0:15  Coding 21:03  Report 0:15 |
| Miguel Graça Oliveira | * Wrapper * Report | Reading 0:45  Coding 0:60  Report 0:15 |
| Pedro Saraiva | * Testing * Report | Reading 0:40  Coding 0:45  Report 0:60 |

# Installation Instructions

1. To install all the applications, it’s only necessary to run the setup program and to follow the usual procedures.
2. The application when ran for the first time will be responsible to create the message queues. The names of the queues are configured in the applications configuration files.
3. Before running for the first time we need to configure the following files:
   1. IntegrationWrapper\_CameraListBeautifier.exe.config
      1. Parameter JavaCameraListBeautifierPath must be configured with the location of the java program.
      2. Parameter MsxslPath must be configured to point to the msxsl.exe.
   2. IntegrationWrapper\_CameraSearchXML.exe.config
      1. Parameter JavaCameraSearchXMLArguments to point to the CameraSearchXML.jar application.
      2. Parameter JavaPath to point to the Java JDK location.
      3. Parameter JavaCameraSearchXMLPath with the location to where the files will be saved.
   3. IntegrationWrapper\_CameraSummaryXML.exe.config
      1. Parameter JavaCameraSummaryXMLArguments to point to the CameraSummaryXML.jar application.
      2. Parameter JavaPath to point to the Java JDK location.
      3. Parameter JavaCameraSummaryXMLPath with the location to where the files will be saved.
   4. MyOrchestrator.exe.config
   5. UserInterface.exe.config

### Application Message Queues



# Implementation and Design decisions

## Architecture

This project is an evolution of the first project. The project goal was to integrate the three applications done in the first project using message queues.

This is the dynamic view of the imposed architecture:



### DeciSions

The first decision was to use the Microsoft Message Queue Server (MSMQ) as the Message Queuing middleware and to use .NET as development framework. Message Queuing enables the existing five applications that are running at different times to communicate across heterogeneous networks and systems that might be temporarily offline (not the case of this project). The existing applications send messages to pre-defined queues and read messages from pre-defined queues.

Message Queuing provided the following benefits related to the first project:

* Guaranteed delivery of messages
* Efficient routing
* Enhanced security
* Priority-based messaging

Several decisions were made:

1. It is necessary to give Generic Read and Generic Write permissions to the “Everyone” user;
2. We decided to create the My Orchestrator application as a normal Windows Forms Application, instead of a Windows Service, which would be correct. This decision was taken for the sake of simplicity. In an industrial environment, this would not be the most appropriate decision, since it is important that the orchestrator is always running. Nevertheless, all the remaining applications were developed to have that in mind, i.e., they are always active and expecting the event of an incoming message through a pre-defined queue;
3. Decided to use an Utilities (Common) library for common functions;
4. One main decision was to use a loose coupling approach. For that reason we decided to have three different Integration Wrappers, instead of a generic Integration Wrapper application;
5. To use the MSXLS program instead of modifying our Camera List Beautifier application because it would require to adapt an existing “legacy system”, and with the MSXLS program it become simpler and a faster option.
6. Integration Wrappers search the directories for newly created files to know when the resulting file was written by the “legacy systems”. This is the only way to do it without changing them since the applications constructed in the first project don’t communicate the name of the file they wrote.
7. All the applications use more than one thread to receive and send messages. This was a constraint imposed by the assignment. As a result, none of the applications behaves synchronously.
8. When a new query is initiated in the UserInterface application, the messages passed to the MyOrchestrator, to the IntegrationWrappers and back to the UserInterface, have a correlation id that allows to associate them.
9. Transactions were not used due to constraints in the schedule of the development.

### Configurations

All queues names can be configured through their configuration files. There are five configuration files where we can define the name of the queues and the location of the applications to be integrated (from the first project).

**MyOrchestrator**

* EAIUserInterfaceInbox – name of the inbox message queue of the UserInterface application
* EAIUserInterfaceOutbox – name of the outbox message queue of the UserInterface application
* iwCameraSearchXmlInboxQueue - name of the inbox message queue of the CameraSearchXML application
* iwCameraSearchXmlOutboxQueue - name of the outbox message queue of the CameraSearchXML application
* iwCameraSummaryXmlInboxQueue - name of the inbox message queue of the CameraSummaryXML application
* iwCameraSummaryXmlOutboxQueue - name of the outbox message queue of the CameraSummaryXML application
* iwCameraListBeautifierInboxQueue - name of the inbox message queue of the CameraListBeautifier application
* iwCameraListBeautifierOutboxQueue – name of the outbox message queue of the CameraListBeautifier application

**IntegratioWrapperCameraListBeautifier**

* iwCameraListBeautifierInboxQueue - name of the inbox message queue of the CameraListBeautifier application
* iwCameraListBeautifierOutboxQueue - name of the outbox message queue of the CameraListBeautifier application
* JavaCameraListBeautifierPath – path to the CameraListBeautifier application
* MsxmlPath – path to the msxml.exe application

**IntegratioWrapperCameraSummaryXML**

* iwCameraSummaryXmlInboxQueue
* iwCameraSummaryXmlOutboxQueue
* JavaCameraSummaryXMLArguments
* JavaPath – path of the java.exe application
* JavaCameraSummaryXMLPath

**IntegratioWrapperCameraSearchXML**

* iwCameraSearchXmlInboxQueue
* iwCameraSearchXmlOutboxQueue
* JavaCameraSearchXMLArguments
* JavaPath – path of the java.exe application
* JavaCameraSearchXMLPath

**UserInterface**

* userInterfaceInboxQueue - name of the inbox message queue of the UserInterface application
* userInterfaceOutboxQueue - name of the outbox message queue of the UserInterface application

### Application Integration with Message Queues Application

## C:\Users\Pedro.Saraiva\Documents\MSE\Curricular Units\4th Semester\Enterprise Application Integration\Source Code\Project 4\A4Programs.jpg