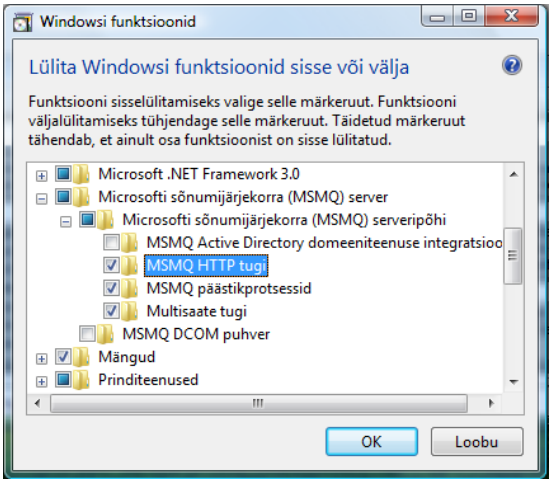
Message Queuing using .NET Framework

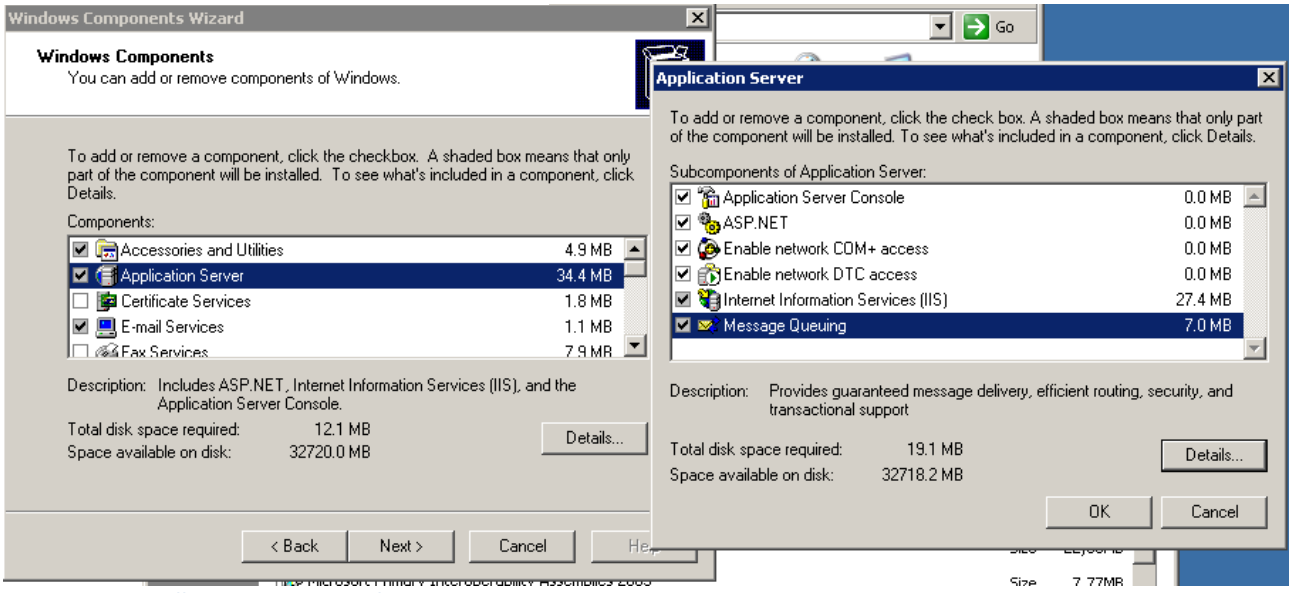
In this practice, two applications will be built. One application translates xCBL PO messages to RosettaNET PO messages. The application will listen to a message queue (MSMQ), process the message and store the result in a file. The transformation to be done is given to you as a XSLT file. Another application will pick files (xCBL PO documents) from a folder and send these to the first application for processing.
The tutorial covers two options for creating the applications. One will be using WCF and requires .NET Framework 3.5 (for which Visual Studio 2008 is the preferred IDE), the second one will use .NET Framework 2.0.
This tutorial is inspired by the scenario in an article written by David Chappel: http://xml.sys-con.com/node/40143.

Additional Requirements

To create message queues on your computer, you need to have Message Queuing feature installed.

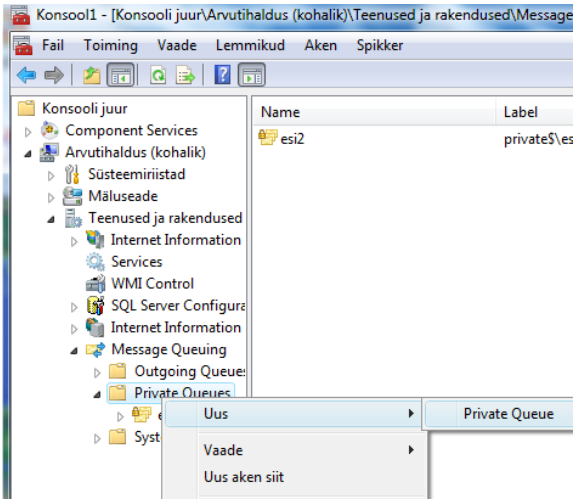


Screencut 1. Installing MSMQ on Windows Vista



Screencut 2. Installing MSMQ on Windows Server 2003

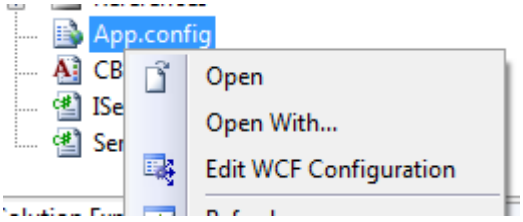
Creating a message queue



You can create a message queue by running Computer Management applet (in Control Panel Administrative Tools) and adding a private message queue in Services, Message Queuing. Please mark the queue as transactional. As message queues are system global, **please append your username to the message queue name in sandstorm!** This avoids conflicts with other students messages and services.

Creating the transforming application (WCF version)

- Create a new WCF Service Library project (if you want all your projects to be in one solutions subfolders, create a "blank solution" project first and then add projects to it).
- Add CBL2Rosetta_PO.xsl to the project.
  - Set the "Copy to Output Directory" property to "If newer"
- Change the service interface (IService1.cs)
  - Remove the DataMember.
  - Add new one way service operation (OneWay has to be specified in OperationContract attribute) called `TranslatePO` that gets `string` and returns nothing (it is one way).
- Edit the service (Service1.cs)
  - Remove the old service implementation.
  - Add the new imterface implementation.
  - Add OperationBehavior attribute to TranslatePO method. Set `TransactionAutoComplete` and `TransactionScopeRequired` `true`.
  - In the implementation you need to
    1. Create a new `XslCompiledTransform` object.
    2. Load the XSL into it.
    3. Create a new `XmlDocument` and load the contents from the incoming message.
    4. Create a new `FileStream` to save the output into a file. It is a good idea to add some hash or date information to the file name to differentiate between different messages.
    5. Transform the message (use empty argument list if necessary).
    6. Close the output stream.
- Open the WCF Configuration editor
  - Create a new Binding
    1. Set the type as netMsmqBinding.
    2. From security tab set Mode and MsMqAuthenticationMode to None.
  - Create new endpoint for service
    1. Set the mode to MSMQ.
    2. Set the client type to WCF application.
    3. Set the BindingConfiguration to the new binding you created.
  - Save the changes.



- Test the application by running it in debug mode. Copy/note the service URL from the WCF Test Client.

Creating a new endpoint demonstrates one of the WCF design goals. WCF makes it easy to expose services on different endpoints. The completed service should be available through both MSMQ and XML Web Service endpoints.

Creating a MSMQ client (WCF version)

- Create or add a new Console Application project.
- Add a new service reference to the URL you copied from the WCF Test Client.
- Add a new Application configuration file.
- Add the example CBL PO file to the project run directory.
- Open WCF Configuration editor.
  - Create a new client from the service configuration file (App.config in the service project).
  - Note the MSMQ endpoint name.
  - Save the changes.
- Change the application Main method:
  - Create a new Service client. If you have more than one possible endpoint, you need to specify the preferred endpoint configuration name to the constructor. If the preferred endpoint is not available, the client will try to communicate using other endpoints.
  - Load the example CBL PO xml into string.
  - Call the TranslatePO operation on the service.
- Test the client. If the client and service are in the same solution, the service will be started automatically when the client is debugged. Verify that you get a Rosetta PO in the service run directory.

Tip: You can load the full file with single instruction in the code by using static method ReadAllText of File object.

Create a MSMQ service using .NET Framework 2.0

- Create a new Console Application project. You can try creating a Windows service on your own computer instead to have more realistic approach (as you would probably implement service servers as service instead of an application in production environment), however, due to the lack of administrative privileges on sandstorm, the instructions are for application only.
- Add a reference to System.Messaging library.
- Add CBL2Rosetta_PO.xsl to the project.
  - Set the "Copy to Output Directory" property to "If newer"
- Edit the Main method of the application
  - Check for the existance of the MessageQueue (the name of the message queue will be sandstorm\\PRIVATE$*your_mq_name*)
  - If the message queue is present, create a MessageQueue object.
  - Add an infinite loop that
    1. Receives message from the queue
    2. Sets the formatter for the message to format strings (create new `XmlMessageFormatter` with an array of strings containing string "System.String")
    3. Transforms and saves the message (see the implementation for transformation given in the WCF version).
- Test the application.

Create MSMQ client using .NET Framework 2.0

- Create a new Console Application project.
- Add a reference to System.Messaging library.
- Add the example CBL PO file to the project run directory.
- Edit the main method
    - Check for the existence of the Message Queue.
    - If the message queue is present
        1. Create a new transaction (`MessageQueueTransaction`) and begin it.
        2. Send the example CBL PO to the queue (make sure to bind it with transaction).
        3. Commit the transaction.
- Test the client and the service. You should see a RosettaNet PO document appear in the service execution folder.

Monitoring

You can see what messages are in the queue by having a look at the queue from the Computer Management console.