How do I create a workflow instance reliably based on an external event?

Asked 16 years, 4 months ago Modified 16 years, 1 month ago

Viewed 1k times

3

a little new to the windows workflow stuff so go easy :)

I wish to design a workflow host environment that has high availability - a minimum of 2 WF runtime hosts on separate hardware both pointing to the same persistance or tracking SQL database.

I am looking for a pattern whereby I can asynchronously create new workflow instances based on some external event (i.e. some piece of data is updated in DB by a different application). For each event I need to create exactly one workflow instance and doesn't matter which host that instance is created on. There is also some flexibility regarding the duration of time between the event and when the workflow instance is actually created.

One solution I am considering is having a WCF interface on the WF hosts and placing them behind some sort of load balancer. It would then be up to whatever part of the system that is firing the "event" to make the WCF call.

I'm not really happy with this because if both\all WF hosts are down, or otherwise unavailable, the event could be "lost". Also, I won't be able manage load the way I would

like to. I envisage a situation where there may be lots of events in a small period of time, but it's perfectly acceptable to handle those events some time later.

So I reckon I need to persist the events somehow and decouple the event creation from the event handling.

Is putting these events into MSMQ, or a simple event table in SQL Server, and having the WF host just poll the queue periodically a viable solution? Polling seems to be a such a dirty word though...

Would NServiceBus and durable messaging be useful here?

Any insights would be much appreciated.

Addendum

The database will be clustered with shared fiber channel storage. The network will also be redundant. In order for WF runtime instances to have fail-over they must point at a common persistence service, which in this case is a SQL backend. It's high availability, not Total Availabilty :)

MSDN article on WF Reliability and High Availabilty

Also, each instance of the WF runtime must be running exactly the same bits, so upgrading will require taking them all down at the same time. I like the idea of being able to do that, if required, without taking the whole system down.

.net workflow workflow-foundation

Share

Improve this question

Follow

edited Nov 2, 2008 at 16:49

asked Aug 14, 2008 at 14:56

user1010

3 Answers

Sorted by: Highest score (default)

1

If you use a WCF service with a netMsmqBinding, you can receive queued messages without having to poll. Messages will wait if there is no service running to pick them up. You would want to make sure to use a clustered queue for reliability in case the main queuing machine goes down.

Also be aware when upgrading that you can't resuscitate instances from an old version of the service. So to upgrade long running workflows, you need to stop them from receiving new requests and wait until all instances are finished before changing the bits, or the old instances will be stuck in your persistence store forever.

Share Improve this answer

Follow

answered Sep 17, 2008 at 21:07

Tegan Mulholland

801 ● 8 ● 11

0

I would go with MSMQ/event table. Polling is only dirty if you do it wrong.

One thing to keep in mind: you say you want multiple WF servers for high availability, *but both of them use the same SQL backend*? High availability only works if you remove *all* single points of failure, not just some of them.

Share Improve this answer

Follow

answered Aug 14, 2008 at 15:56

Stu

15.8k ● 4 ● 45 ● 74

This is how I have solved it.

I'm using NServiceBus and with each WF runtime host pointing to the same messagebus (using MSMQ as a transport). NServiceBus supports transactional reads off the message bus and rollback. If a message is taken off the bus but the process terminates before the message is fully handled it remains on the queue and a different runtime host will pick it up.

In order to have WF runtime hosts running on separate machines, the messagebus\queue will have to reside on Windows 2008 server (MSMQ 4.0) or later, as earlier versions of MSMQ don't support remote transactional reads. Note also, in order to perform a remote transactional read, the machine performing the read will also need to have MSMQ 4.0 installed (i.e. Windows Server 2008)

answered Nov 2, 2008 at 16:55

user1010