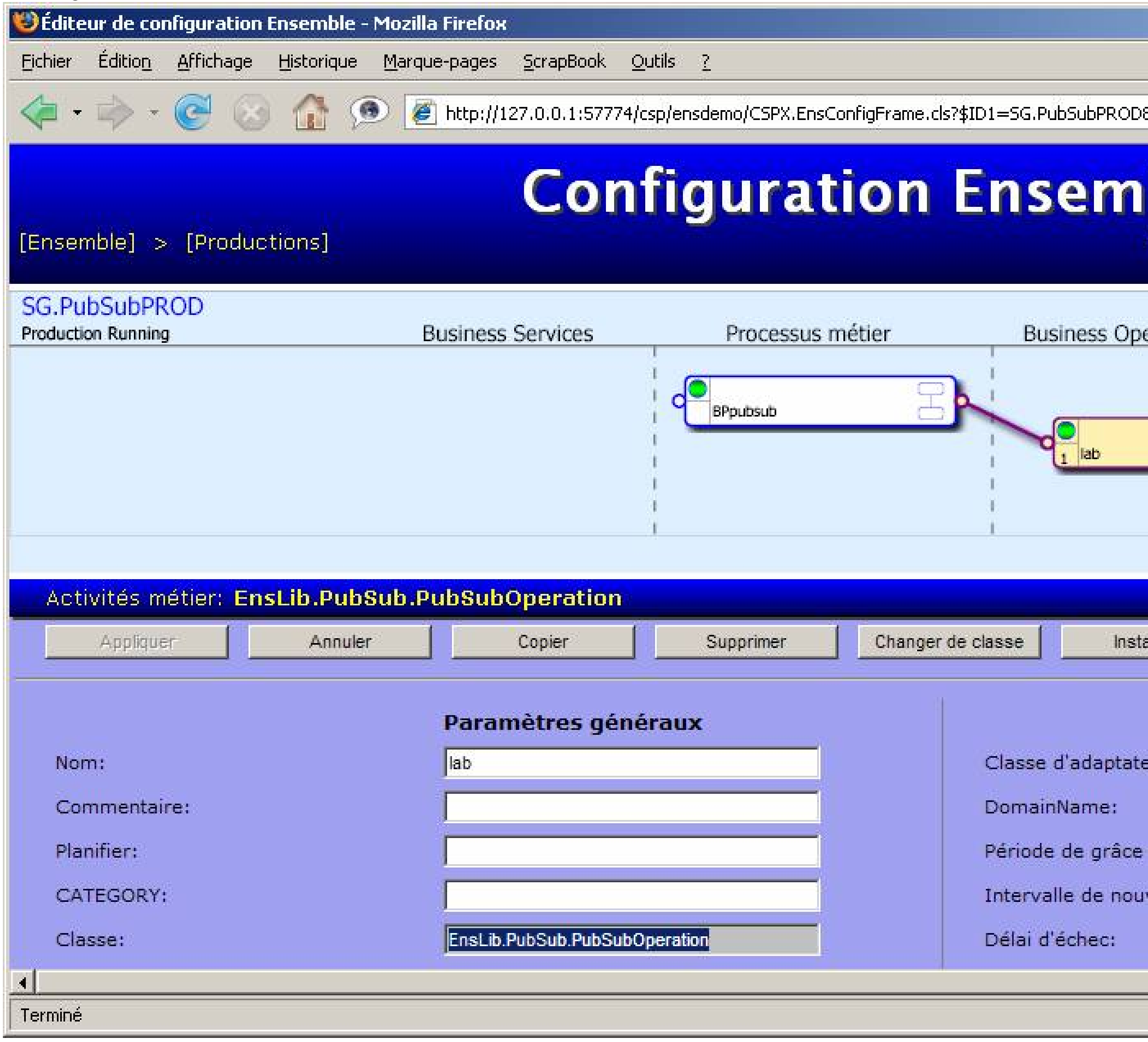
Hi Luca,

Attached is a Publish/Subscribe sample I've written with Ensemble 2007.1.2 (it may have changed with Ensemble 2008.1 ; I didn't check).
At the end of the email you'll find the only doc available at this time.
HIH

Configuration :

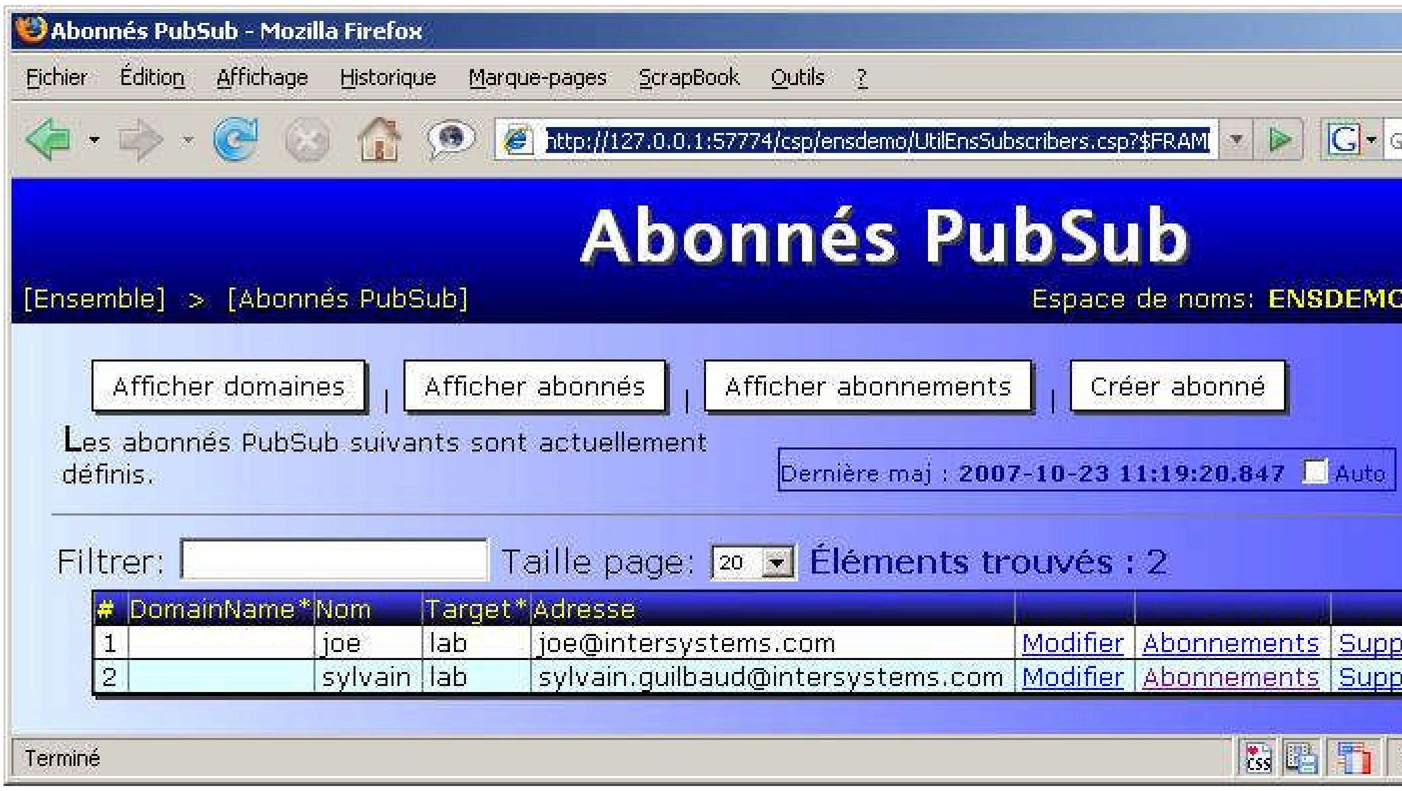

Éditeur de configuration Ensemble - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages ScrapBook Outils ?
http://127.0.0.1:57774/csp/ensdemo/CSPX.EnsConfigFrame.cls?$ID1=SG.PubSubPROD
Configuration Ensem
[Ensemble] > [Productions]
SG.PubSubPROD
Production Running
Business Services
Processus métier
Business Ope
BPpubsub
lab
Activités métier: EnsLib.PubSub.PubSubOperation
Appliquer
Annuler
Copier
Supprimer
Changer de classe
Insta
Terminé

Paramètres généraux

Nom:	lab
Commentaire:	
Planifier:	
CATEGORY:	
Classe:	EnsLib.PubSub.PubSubOperation

Classe d'adaptate
DomainName:
Période de grâce
Intervalle de nou
Délai d'échec:

Page de création des abonnés :
http://127.0.0.1:57774/csp/ensdemo/UtilEnsSubscribers.csp?$FRAME=&$ID1=2&$NAMESPACE=&$SUBMITBTN=$AUTOFORM_SAVE



Abonnés PubSub - Mozilla Firefox

http://127.0.0.1:57774/csp/ensdemo/UtilEnsSubscribers.csp?$FRAM

Abonnés PubSub

[Ensemble] > [Abonnés PubSub]

Espace de noms: ENSDEMO

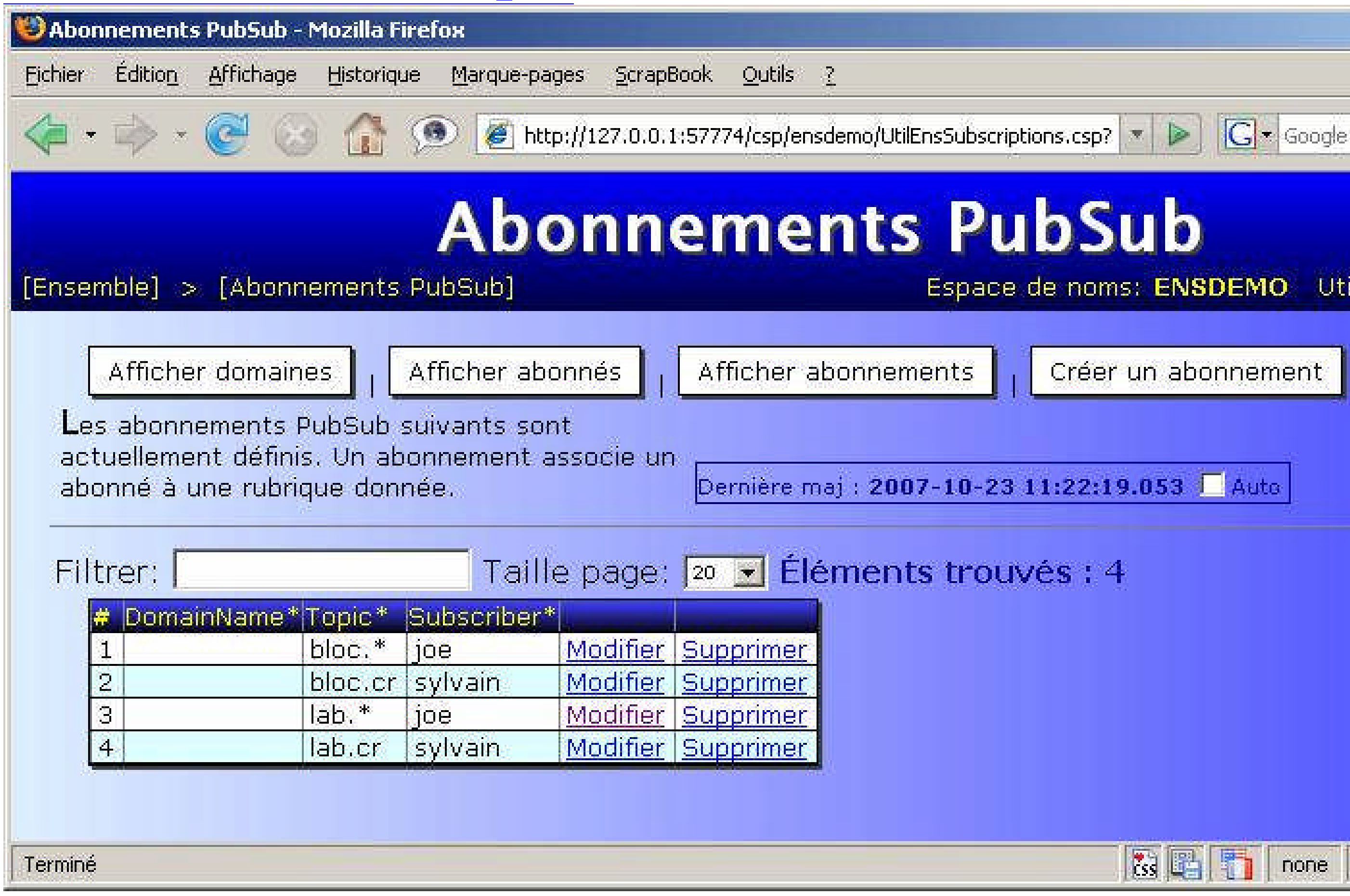
Les abonnés PubSub suivants sont actuellement définis.

Dernière maj : 2007-10-23 11:19:20.847 Auto

Filtrer: Taille page: 20 Éléments trouvés : 2

#	DomainName*	Nom	Target*	Adresse			
1		joe	lab	joe@intersystems.com	Modifier	Abonnements	Supp
2		sylvain	lab	sylvain.guilbaud@intersystems.com	Modifier	Abonnements	Supp

Page de création des abonnements :
http://127.0.0.1:57774/csp/ensdemo/UtilEnsSubscriptions.csp?$FRAME=&$ID1=4&$NAMESPACE=&$SUBMITBTN=$AUTOFORM_SAVE



Abonnements PubSub - Mozilla Firefox

http://127.0.0.1:57774/csp/ensdemo/UtilEnsSubscriptions.csp?

Abonnements PubSub

[Ensemble] > [Abonnements PubSub]

Espace de noms: ENSDEMO

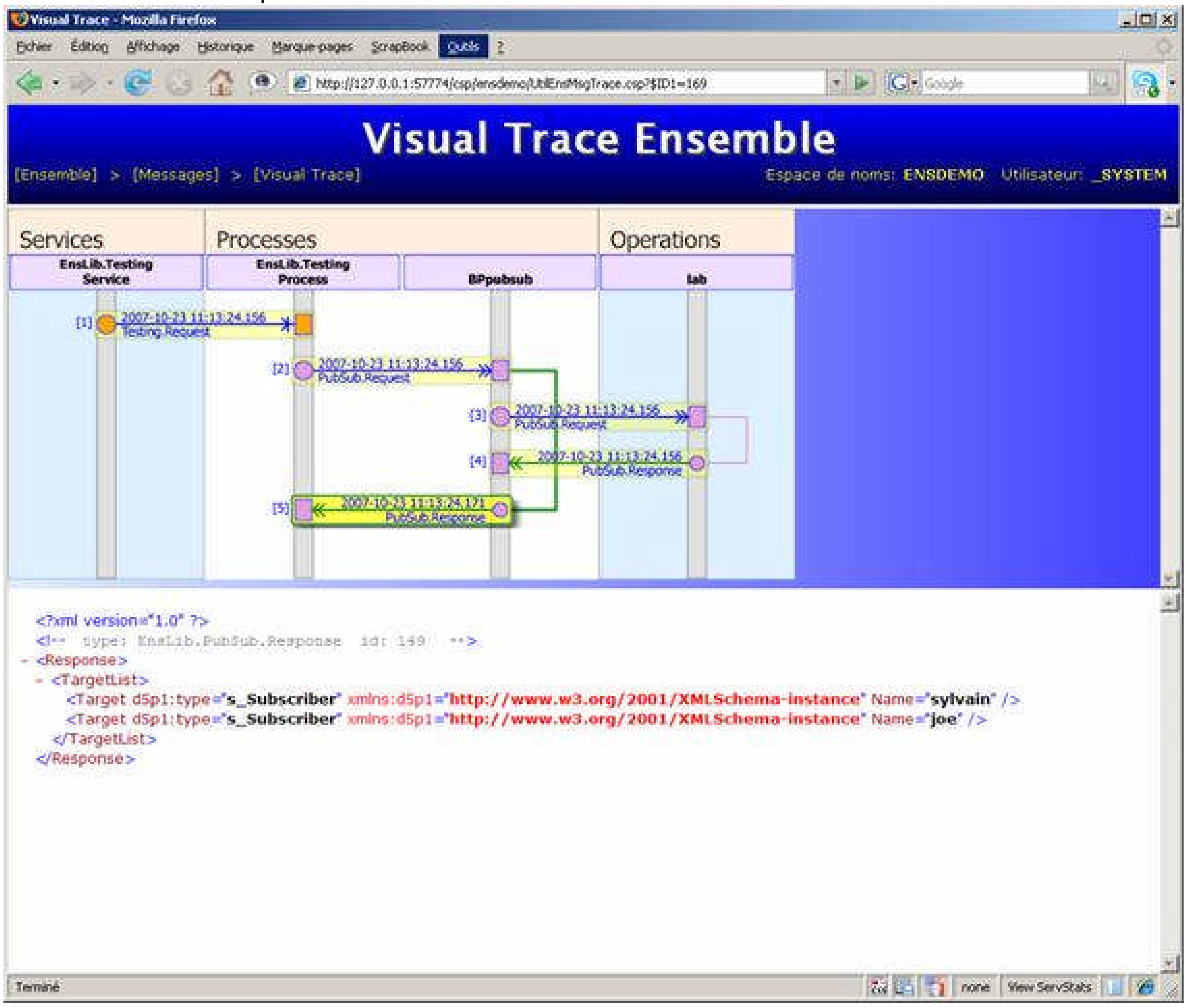
Les abonnements PubSub suivants sont actuellement définis. Un abonnement associe un abonné à une rubrique donnée.

Dernière maj : 2007-10-23 11:22:19.053 Auto

Filtrer: Taille page: 20 Éléments trouvés : 4

#	DomainName*	Topic*	Subscriber*		
1		bloc.*	joe	Modifier	Supprimer
2		bloc.cr	sylvain	Modifier	Supprimer
3		lab.*	joe	Modifier	Supprimer
4		lab.cr	sylvain	Modifier	Supprimer

```
Résultat avec request= « bloc.cr »:
```


Visual Trace - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages ScrapBook Outils ?
http://127.0.0.1:57774/csp/ensdemo/UtilEnsMsgTrace.csp?$ID1=169
Visual Trace Ensemble
[Ensemble] > [Messages] > [Visual Trace]
Espace de noms: ENSDEMO Utilisateur: _SYSTEM
Services
Processes
Operations
EnsLib.Testing Service
EnsLib.Testing Process
BPpubsub
lab
[1] 2007-10-23 11:13:24.156 Testing.Request
[2] 2007-10-23 11:13:24.156 PubSub.Request
[3] 2007-10-23 11:13:24.156 PubSub.Request
[4] 2007-10-23 11:13:24.156 PubSub.Response
[5] 2007-10-23 11:13:24.171 PubSub.Response
Terminé
none
View ServStats

```
<?xml version="1.0" ?>
<!--  type: EnsLib.PubSub.Response  id: 149  -->
- <Response>
  - <TargetList>
      <Target d5p1:type="s_Subscriber" xmlns:d5p1="http://www.w3.org/2001/XMLSchema-instance" Name="sylvain" />
      <Target d5p1:type="s_Subscriber" xmlns:d5p1="http://www.w3.org/2001/XMLSchema-instance" Name="joe" />
    </TargetList>
  </Response>
```

```
Regards,
Sylvain

--
```

De : Alex Schaefer [mailto:Alex.Schaefer@intersystems.com]
Envoyé : mardi 23 octobre 2007 10:18
À : Sylvain Guilbaud
Cc : EAI Engineering List
Objet : Re: PubSub

This is what Susan sent around last week, not sure if it helps but I believe that's all we have right now.

Publish and Subscribe

Ensemble provides support for Publish and Subscribe Message routing. Publish and Subscribe refers to a technique where a "message" is sent to one or more "subscribers" based on their interest in a specific "topic"; typically a topic is inferred from the contents of the message.

Within Ensemble PubSub Message routing:

A *message* is an Ensemble message. Typically a request is received by an external system, converted to an Ensemble message, and then handed to the PubSub Message router for processing.

A *topic* is a string that is used to describe the contents of a message. A topic string is of the form "A.B.C.D", where A, B, C, D, are subtopics and are delimited by the "." character. Each subtopic can be up to 50 characters in length; a given topic can contain any number of subtopics. Ensemble does not define any topics: the meaning of topics and subtopics are defined by users and their applications. Similarly, the determination of what topic best characterizes a given message is the responsibility of the application. For example, a HealthCare production may define a set of topics like:

    "Patient.LAB.X3562564"
    "Patient.RAD.X3489989"
    "Doctor.ICU.88494"

Where the top-level subtopic is either "Patient" or "Doctor" and determines who a given message pertains to; the second-level subtopic refers to a department, and the third-level subtopic provides some sort of topic-specific ID number (such as a patient ID).

A *Subscriber* is an entity (user or external system) that may be interested in a specific topic or set of topics. Each subscriber defines how it wishes to be contacted (i.e., how Ensemble should send a message to it).

A *Subscription* defines an association between a Subscriber and a specific topic (or set of topics). Each subcription consists of a Subscriber and a matching topic string (which can contain "*" as a wildcard match for any subtopic; note that you cannot use wildcards as part of a subtopic: "A*" will not match anything).

For example, using the topic definition outlined above, we could define three Subscribers:

    Abel
    Baker
    Charlie

Now we can define a simple set of subscriptions:

Abel	Doctor.ICU.88494
Abel	Doctor.ICU.88495
Baker	Doctor.ICU.88495

```
    Baker   Patient.LAB.*
    Charlie *.*.X3562564
```

In this case, Abel is notified whenever the exact topics, "Doctor.ICU.88494" or "Doctor.ICU.88495" are processed. Baker is also notified whenever the exact topic "Doctor.ICU.88495" is processed. In addition, Baker is notified whenever any message related to Patients in the Lab are processed. Charlie is notified whenever anything related to someone with ID of "X3562564" is processed.

You can define the set of Subscribers and Subscriptions using either a set of dedicated web pages within the Ensemble Management portal, or you can manipulate these structures directly via the API provided.

The Ensemble PubSub routing engine does not actually send messages to subscribers; instead, it provides a mechanism to quickly find the set of interested subscribers for a given topic. It the responsibility of an application-specific Business Process to both invoke the PubSub routing engine and to dispatch messages to subscribers as part of a Business Process (this is in order to make the actual processing of messages as flexible as possible).

Implementation:

Within Ensemble, Publish and Subscribe messaging is implemented as follows:

* Ensemble provides a Business Operation class, EnsLib.PubSub.RoutingOperation, that provides a generic framework for processing messages and deciding how to route them. The idea being that additional types of message routing (other than Publish and Subscribe) can be defined either as part of the Ensemble library or by Ensemble users.

* There is a Business Operation class, EnsLib.PubSub.PubSubOperation, that provides the implementation of Publish and Subscribe message routing. This is a subclass of the afore-mentioned EnsLib.PubSub.RoutingOperation class.

* There is a request class, EnsLib.PubSub.Request, that is used to package up requests to the PubSubOperation class. Basically, this wraps an incoming message with additional information about what Topic and DomainName (q.v.) should be used for determining how the message should be routed.

* There is a corresponding response class, EnsLib.PubSub.Response, that is used for responses from the PubSubOperation class. Specifically, this returns a list of 0 or more *targets*--instances of EnsLib.PubSub.Target object. Each EnsLib.PubSub.Target object

provides details on how a message should be routed (who to send it to, how to transform it, etc.).

* There is a persistent class, EnsLib.PubSub.Subscriber, that represents individual subscribers-- entities that are interested in being notified when certain messages arrive. The EnsLib.PubSub.Subscriber class includes any information needed to notify a subcriber.

* There is a persistent class, EnsLib.PubSub.Subscription, that stores the actual subscription information. This associates a given Subscriber with a Topic string. This topic string can contain "*" as a wildcard character.

* There is a persistent class, EnsLib.PubSub.DomainName, that is used to hold the set of PubSub "domain" names. A domain name is simply a way to organize multiple subscription lists by associating them with arbitrary domains. An Ensemble production may use PubSub routing for several different reasons; PubSub domain names provide a to keep the different subscription lists separate. Alternatively, you can simply choose to ignore PubSub Domain Names.

* There is a utility class, EnsLib.PubSub.Utils that provides a programmatic API for creating and deleting domains, subscribers, and subscriptions.

Using PubSub:

1) First you have to set up a list of Subscribers. A subscriber is some agent that is to be notified when a specific topic is encountered. The UtilEnsSubscribers.csp page lets you see the current set of subscribers and add to or edit them.

A subscriber has a Name and Target-- Target is the configuration name of a Business Operation to which messages for this subscriber should be sent.

Note that Subscribers can be placed within a Domain if desired. This is simply a mechanism to allow multiple sets of Subscribers without collision. Domain is optional.

2) Once you have Subscribers, you can create Subscriptions. A Subscription associated a topic or set of topics with a Subscriber. A topic is simply a string containing one or more subtopics with "." as a delimiter. For example "books", "books.fiction", and "books.fiction.latin" are all topics.
Note that there is no predetermined definition of topics-- it is up to the application to decide what set of topics it will use.
A subscriber can subscribe to a specific topic (such as "books.fiction.latin") OR subscribe to a range by using the wildcard

character "*". For example: "books.*", "books.*.latin". The "*" can appear in the place of any subtopic. A trailing "*" should match on any number of additional subtopics. You cannot use * within a subtopic: "b*" is NOT a valid subscription.

3) To use PubSub you have to set up a production that includes the PubSub operation. There should be a business process that calls the PubSub operation with a EnsLib.PubSub.Request message. This request include a Topic (as described above, but, of course, with NO wildcards in it); a Domain name (if used); and, optionally, the message being routed (the PubSub engine does not need to see the message). Note that it is the application's responsibility to determine what the Topic string should be for a specific message.

The PubSub operation finds and returns a list of all Subscribers that are interested in the given topic (using a very fast search algorithm). The response from the PubSub operation contains a collection, TargetList, of EnsLib.PubSub.Target objects; these describe the set of targets (subscribers) that the message should be sent to. The calling Business Process should iterate over this collection and dispatch the message as specified. Note that this Business Process is not built-in as every application may have its own concept of how it wants to process this information.

###

On Oct 23, 2007, at 4:13 AM, Sylvain Guilbaud wrote:

Hi all,

Any information/samples regarding EnsLib.PubSub would be greatly appreciated.

A customer would like to use it.

Thanks in advance.

Regards,

Sylvain