weswit
Software Made in Italy

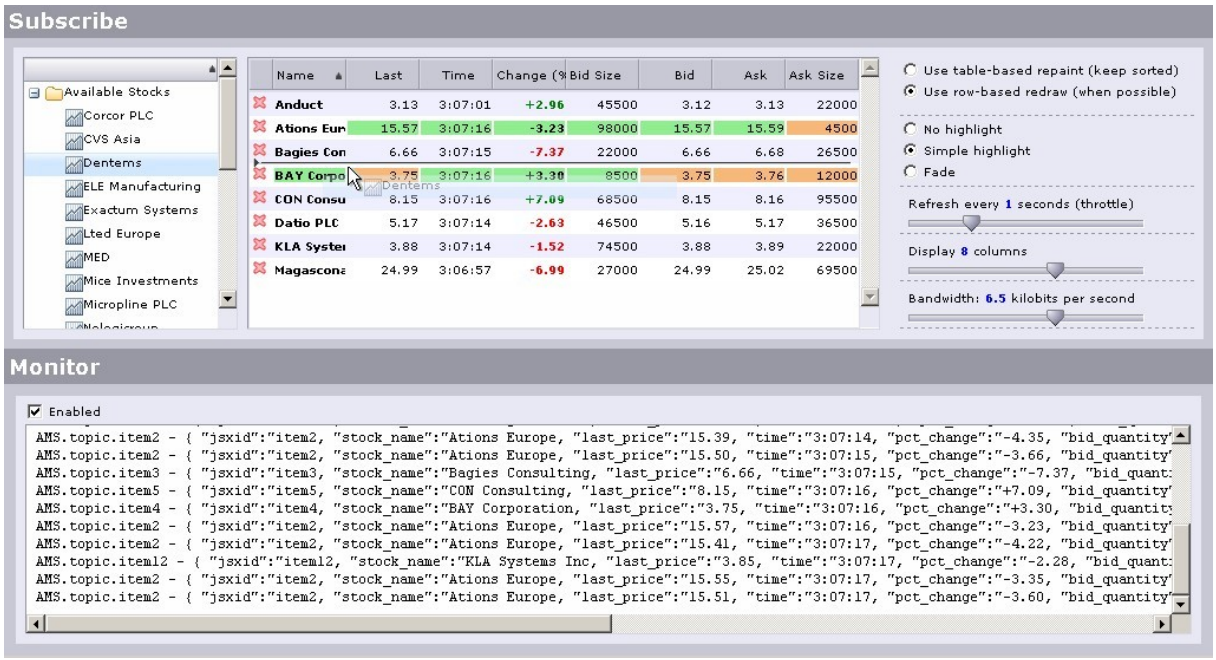
GI – PageBus – AMS Integration Demo

Last updated: 24/05/2012

Table of contents

1 Introduction

This demo shows a complete integration between Ajax Message Service (AMS) and General Interface (GI). The AMS client, receiving events from Lightstreamer Server, publishes stock updates on the PageBus. The application, created with the GI builder, subscribes to the PageBus in order to receive and present updates to the user.

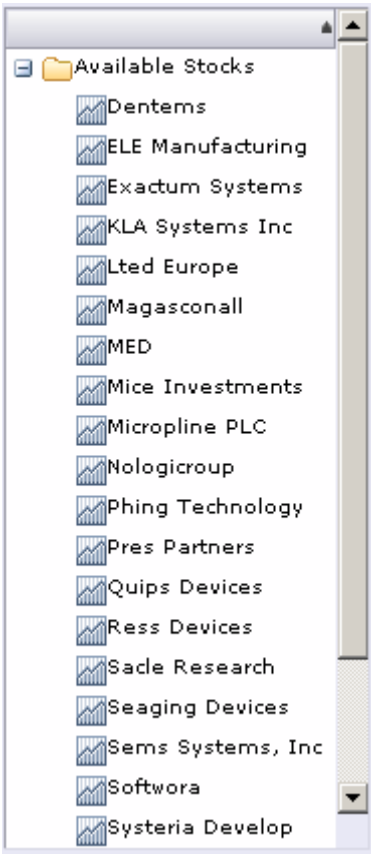


2 Using the demo

The application is comprised of four panels:

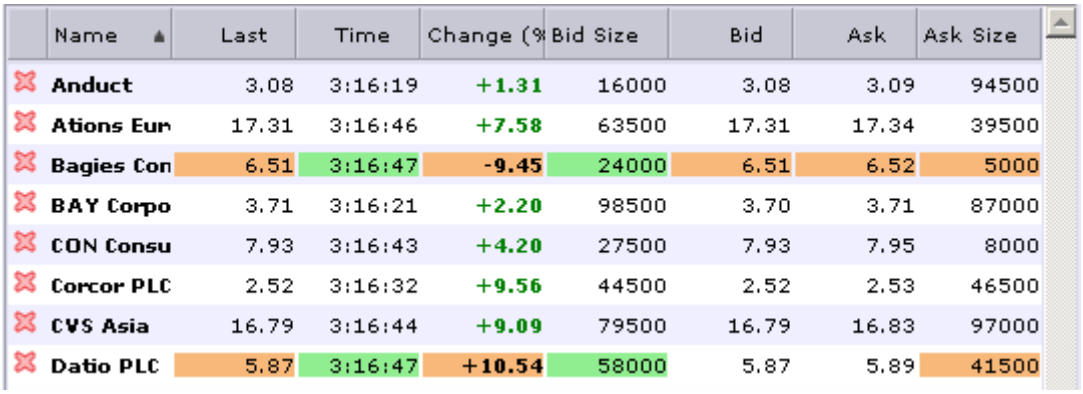
- Available stock list
- Update panel
- Configuration panel
- Monitor panel

2.1 Available Stock List



This part of the application shows the available stocks to be subscribed. Here you can see a list of stock names. Drag a stock to the Update Panel to subscribe the stock to the server. Once added the stock will disappear from the available stocks list and updates will start to be published on the update panel. You can also subscribe a stock by double-clicking it. The behaviour will be the same as you have dragged the stock.

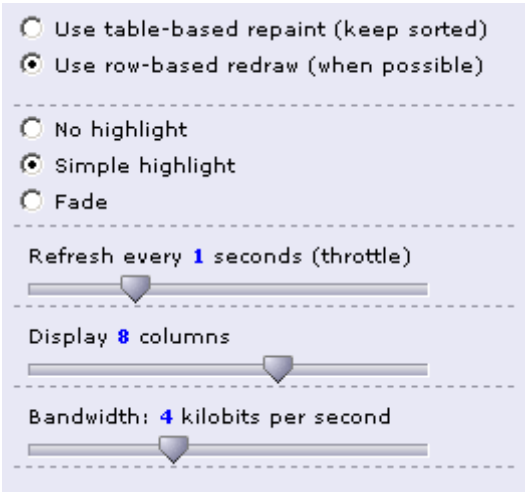
2.2 Update Panel



Name	Last	Time	Change (%	Bid Size	Bid	Ask	Ask Size
Anduct	3.08	3:16:19	+1.31	16000	3.08	3.09	94500
Ations Eur	17.31	3:16:46	+7.58	63500	17.31	17.34	39500
Bagies Con	6.51	3:16:47	-9.45	24000	6.51	6.52	5000
BAY Corpo	3.71	3:16:21	+2.20	98500	3.70	3.71	87000
CON Consu	7.93	3:16:43	+4.20	27500	7.93	7.95	8000
Corcor PLC	2.52	3:16:32	+9.56	44500	2.52	2.53	46500
CVS Asia	16.79	3:16:44	+9.09	79500	16.79	16.83	97000
Datio PLC	5.87	3:16:47	+10.54	58000	5.87	5.89	41500

This panel shows the updates for the subscribed stocks. New updates are highlighted in order to catch the user's attention. The highlighting is green whenever the new value for a certain field is greater than the older value (for example the Time column is always green highlighted) and red in the other case.
You can click on column headers in order to sort the table on a certain field and can drag columns to switch their positions (however notice that the first two columns are fixed and cannot be moved). To remove a stock from the panel it is possible to click the X button or to drag the stock to the Available Stocks (Drag to the folder icon).

2.3 Configuration Panel



This panel enables you to customize the application behaviour. It is composed by 5 sections (top-bottom):

1. Let you choose the kind of repainting. With "table-based repaint" the entire table is repainted for each update, while with the "row-based redraw" only updated cells are repainted. The first approach could be heavier on the CPU but keeps the entire table sorted based of the latest updates. Note that the highlighting is not available. On the contrary with the "row-based repaint" approach cells are high lightened on any update but if the table is sorted by clicking on a column, it could lose its sorting as soon as a new update arrives.

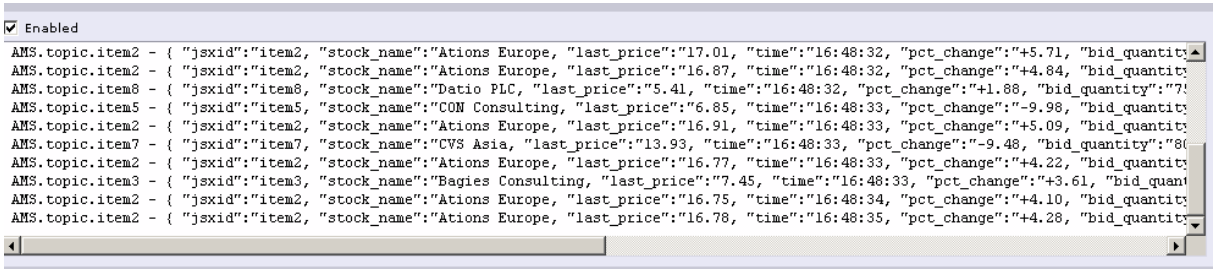
2. When “row-based” repaint is selected this second section enables you to choose the highlighting policy. Three options are available: a simple highlight that changes the background of the updated cell for a while and then returns back its original background; a fade highlight that changes the background of the updated cell for a moment and then fades the cells until it reaches its original background (disabled at the moment); a no highlighting policy that let you disable highlighting in order to slow down CPU requirements.

3. The first slider lets you control the refresh rate of the update panel. It is possible to choose between a delayed update of the view (from 1 to 4 seconds of delay) to let the application aggregate different updates arrived in a time slice in a single repaint on the page, or a real-time update so that each update arrived from the server is immediately rendered on screen. This second approach could be CPU consuming but lets you see at any time the latest data available on the client with no added latency (to set to real-time update move the slider completely to the left).

4. Different columns are available in the table. This slider let you choose how many columns to show on the view.

5. The last slider lets you control the bandwidth used to push the data from the server. This value is handled on the server side (so you have to wait a moment for the new value to be applied).

2.4 Monitor Panel



You can show or hide this panel by clicking on the checkbox located on the top left of the panel itself. If enabled, the panel will show all the messages exchanged between different parts of the application through the PageBus. You can see all the update traffic, subscribtion and unsubscription requests, connection status changes and so on. Note that when disabled, the monitor is not visible at all.

3 How The Application Works

The application includes three JavaScript frameworks (GI, AMS Web Client, PageBus), some application JavaScript files (amspublisher.js, giclient.js, monitor.js) and Lightstreamer Server (plus a common web server).

3.1 Workflow Overview

The application includes by five actors:

⇒ The **GI Matrix** (can be assimilated to the entire GUI) listens to user interactions in order to raise events on the giclient object.
⇒ The **giclient** object listens to GUI events and transforms user interactions in subscription requests. The giclient interacts with the PageBus, publishing subscription and unsubscription requests and listening to update events.
⇒ The **PageBus** is a complete client-side Pub/Sub system. It takes care of the dispatching of events between publishers and subscribers.
⇒ The **amspublisher** object makes a bridge between AMS client API and the PageBus. It subscribes to "publisher.*" subjects to receive orders from the GUI and publishes update events.
⇒ The **AMS Web Client** implements the network connections with Lightstreamer Server to subscribe/unsubscribe items and receive updates in order to dispatch them to the amspublisher.

The following sequence diagrams depicts a typical interaction among the five actors, when a subscription to "item5" is requested by a user gesture.

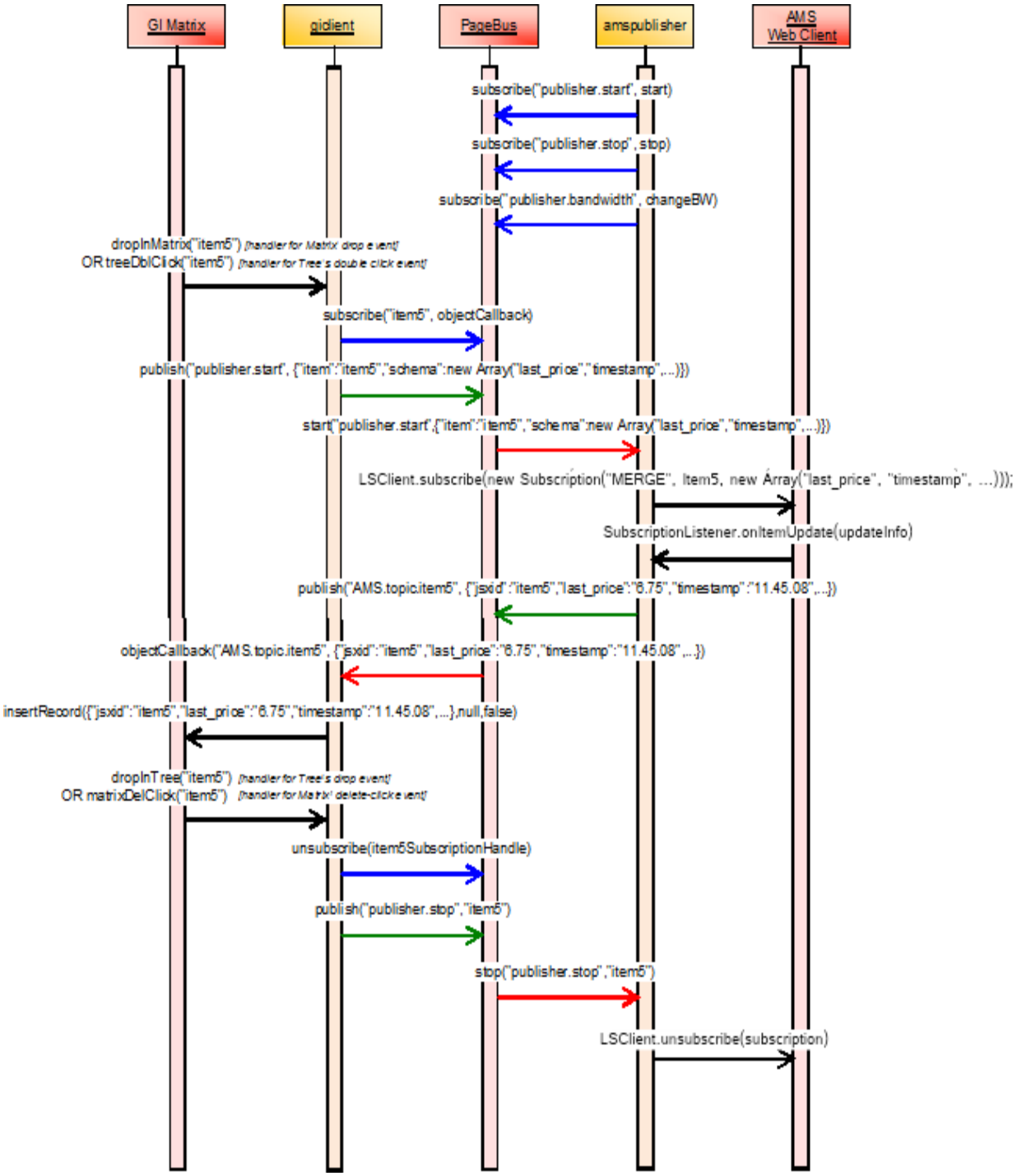
Sequence diagram that shows the flow of messages from application startup to subscription and unsubscription ot "item5".

3.2 amspublisher.js and AMS Web Client

The AMS client is used to receive updates from the Lightstreamer Server. Initially it starts without any subscription to the server. Then it subscribes to three PageBus subjects: "publisher.start", "publisher.stop" and "publisher.bandwidth". Events for such subject are published by the GUI handler (the GI application).

A "**publisher.start**" message carries an item name and a schema array. In order to receive any updates the AMS client subscribes to those item/schema on the Lightstreamer Server. Each time an update arrives from the server it is published to the "AMS.topic.<itemName>" topic (<itemName> is the name of the item so that there is a different topic per each subscribed item) on the PageBus.

A "**publisher.stop**" message carries an item name. This item will be unsubscribed from the Lightstreamer Server so that no more updates for it will arrive.



A "**publisher.bandwidth**" message carries a bandwidth value. The AMS client will issue a request to the Lightstreamer Server in order to increase or reduce the bandwidth used to push the updates to this client.

The AMS client publishes messages on another couple of subjects. Those message have not a direct consumer, but will be logged by the monitor panel, if active. Those subjects are **AMS.error**, **AMS.warning** and **AMS.status**. The latter reports changes on the status of the connection between the AMS client and the Lightstreamer Server.

3.3 GI GUI (giclient.js)

The GI client starts without subscribing to any subject on the PageBus. It just listens to event from the GUI. When an event that requests a new stock in the Matrix is raised, the client subscribes to the relative topic ("**AMS.topic.<itemName>**") and then publishes on the "**publisher.start**" topic a message requesting the AMS client to subscribe on the Lightstreamer Server to the new stock. In the same way, if an item is removed from the Matrix, the GI client unsubscribe from the pertaining subject and sends a message on the "**publisher.stop**" topic to request the unsubscription from the stock on the Server.

The GUI has some controls:

1. **"Table-based repaint" vs "Row-based redraw"**
   A radio button to control the approach used to update the view of the matrix object. The table-based approach is implemented by delegating to the GI library the repainting of the entire data set (using the repaintData method). This means that each time an update to the view is performed, the Matrix is repainted and, if needed, resorted. In the other case the update method obtains a reference to the DOM object containing a single data and updates directly the DOM. This way the resorting is not performed.

2. **Highlighting type**
   If in "Row-based redraw", it is possible to choose between two highlight policies (or to disable highlight at all). When the "simple highlight" policy is selected each time the value inside a DOM node is changed, its background is changed too. A function called after a timeout will return back the background to its original color. The difference between the simple and the fade highlight is that the latter returns back the original color not at once but fading it from the highlighted color.

3. **Refresh rate.**
   There are two ways the refresh rate is handled. When its value is 0 (real-time) values are changed in the view within the same thread of the PageBus while in the other case updates are buffered and sent to the view only at set intervals.

4. **Columns number**
   If columns number is changed the view is changed to reflect the new value. A template xml is loaded to add columns and then customized to reflect its position (header and xslt could differ between columns).

5. **Bandwidth.**
   When the bandwidth slider is moved, the GI client publishes a message on the publisher.bandwidth subject on the PageBus with the new requested value that will be in turn forwarded to Lightstreamer Server.

Subscriptions and unsubscriptions are driven by the user through GUI events. Event handlers listens to drag and drop, click and double click events in order to subscribe to or unsubscribe from a stock.

3.4 Monitor (monitor.js)

The monitor contains some very simple code, so that when it is activated it subscribes to every topic ("**" subject) on the PageBus. Each update received is serialized to a string and printed to the monitor panel in the application.