



AltioLive Developer Training

Version 5.4

5. Getting Live Data using Datapools

Summary

This module explains:

- How to get live data working in AltioLive applications using datapools, and
- How to highlight updates in a database by using Rules and the Traffic Lighting properties.

Integra SP

88 Wood Street London

EC2V 7RS

United Kingdom

www.altio.com

tel: +44 (0) 20 8528 1045

Contents

Introduction to Datapools.....	2
Timestamps.....	2
Datapool Longevity	3
Datakeys: AL_ID attributes	3
Example of Datakey Properties.....	3
Creating Datapools and Data Updates for the Stock Application.....	6
Defining a Datapool	6
Adding the new STOCKS_POOL in the Designer	7
Client Update	7
Introduction to Client Update.....	7
Updating data in the Stock application.....	7
Using Traffic Lighting property and Format Rules	10
Using Format Rule.....	10
Using Update Rule.....	12

Introduction to Datapools

The Datapools are used for applications where real time data updates are required.

An AltioLive application developed without using Datapools is forced to work in a client-server or synchronous mode. When updates are made to Initial Data the application cannot reflect the changes until a Service Request is used to retrieve the new data. With Datapools, the changes can be reflected immediately.

Datapools provide the means to distribute updates from the Server to the Clients in an efficient asynchronous mode. Each Datapool is a table of time-stamped data records managed by the Presentation Server. As the XML in the Datapool changes, the Presentation Server distributes the changes from the back-end application through the Datapool mechanism.

Client(s) can poll Datapools, or can receive streamed updates from the Datapools. A Datapool is instantiated when the first client subscribes.

It is better to have several smaller Datapools for specific data rather than one all-embracing large Datapool. This allows more control over refresh rates. Different datapools can have different poll rates; therefore critical data could be handled by a datapool with a high poll rate whereas non-critical data could be updated to the client less often via another datapool.

Timestamps

Each Datapool has an associated Timestamp. This is in the form of a long integer value or in Date/Time format. The Timestamp is initially set when the first Client subscribes to the Datapool and receives its Initial Data

Thereafter, the Timestamp is updated whenever the Presentation Server receives updated data from the Back end application: whether by polling (in which case it passes its current value in the request) or through streaming. The new Timestamp is specified as an attribute in the initial tag of the response.

The interpretation of the Timestamp is determined by the Back end application and is specific to each Datapool (i.e. AltioLive does not specify whether the number is milliseconds, seconds, next sequence number etc). However, the Timestamp must get numerically larger with each update.

The Timestamp is sent in integer or date/time format. It should support the expected lifetime of an application. In the unlikely event that the Timestamp has to be reset, then the Presentation Server would need to be reset.

By default, the Back end application returns the Timestamp value as an attribute of the parent node of the data update posted by the System. Alternatively, the Timestamp can be retrieved from a specified element attribute from the data update received from the system in a specified format.

If for any reason, an update is received with a duplicate of earlier Timestamp then the Presentation Server will output a warning message.

Datapool Longevity

The Datapool effectively has an entry per timestamp. The size of the Datapool is determined through the CACHESIZE attribute.

The required size of the Datapool depends on the frequency of updates to Clients and the frequency of updates from the Back end application. It is important to ensure that all new data required by a particular client is present in the datapool, and that none of the data is archived to file prior to them receiving it. If the Client were to poll at 3-second intervals and the Back end application creates updates at a rate of 2 per second, then it should be set to at least four times the rate (i.e. $4 \times 3 \times 2 = 24$) to provide contingency.

Once the cache is full, the oldest entries are flushed from the datapool. Datapool size, how many entries to flush, and disk/memory cache allocation sizes are all configurable datapool options.

Datakeys: AL_ID attributes

All data elements in the XML Desktop need to be able to be referenced uniquely. Altiolive uses an **AL_ID** attribute to define a unique key for each element, which will be unique across all data. The Presentation Server adds these **AL_ID**s through the datakeys definition.

The **AL_ID** is created for every element. By default the **AL_ID** uses this definition:

concat(parent::*/@AL_ID, name()) where none has been specified. The **AL_ID** is created from the datakey configuration which can be accessed by the Application Manager.

Please note: to access the Datakey parameters launch the Application Manager, and expand the **Datakeys** folder. Select the datakey you want to set and display the **Properties** window by clicking on

the **Show properties** icon .

For more information on Datakeys, please refer to 2. The First Steps to Build Your Own Application, Testing the HTTP Service Function and creating Datakeys.

Example of Datakey Properties

For example: The datakey for the STOCK elements is defined as:

Property	Value	Comments
Datakey	STOCK	The name should be the same as the element name
Prefix	S-	A prefix added to each key generated from this Datakey
Key	@STOCK_ID	An attribute or a combination of attributes
Namespace URI		If you do not want to apply the Datakey to all the elements matching the STOCK name but only some of them, enter a namespace URI. The datakey will apply only to the elements which match this namespace. If this parameter is left blank the datakey will apply to all the elements matching the STOCK name.

Therefore

- the data held within the **STOCK** element has an **AL_ID** attribute that is made up of the prefix value (**S-**) and the value of the **STOCK_ID** attribute of the element, as shown below:

```
<STOCK AL_ID='S-S1' NAME='ABC' STOCK_ID='S1' ... />
```

- The **AL_ID** generated for the STOCK element is: **S-S1**

Remember, the **STOCK_ID** attribute must hold a unique value. For the Altio DB back end, new stock elements are given a new **STOCK_ID** value based on the current value held in the file **altiodbparams.xml**, which continuously increments so that a new unique **AL_ID** value can be created.

All the datapool entries include the **AL_ID** attribute to ensure that the correct data is updated at the client:

- Example of XML for an update to an existing stock element:

```
<STOCKS TIMESTAMP='2' _tableLocation='dbdata/stocks.xml'>
<STOCK TS='2' STOCK_ID='S1' NAME='ABC' AL_ID='S-S1' .../>
</STOCKS>
```

- Example of XML for a new stock element:


```
<STOCKS TIMESTAMP='19' _tableLocation='dbdata/stocks.xml'>
<STOCK TS='19' STOCK_ID='1000' NAME='AAA' AL_ID='S-1000' ... />
</STOCKS>
```

The **AL_ID** attribute can be made up of more than one attribute value to ensure that it is unique.

For example in the following data:

```
<STOCK STOCK_ID='S01' ... >
  <HISTORY HISTORY_ID='H01' ... />
  <HISTORY HISTORY_ID='H02' ... />
  <HISTORY HISTORY_ID='H03' ... />
</STOCK>
<STOCK STOCK_ID='S02' ... >
  <HISTORY HISTORY_ID='H01' ... />
  <HISTORY HISTORY_ID='H02' ... />
</STOCK>
```

In order to update the correct history it needs to have an **AL_ID** attribute made up of both the **HISTORY_ID** and the **STOCK_ID** to ensure that it is unique.

Please note: to access the Datakey parameters launch the Application Manager, and expand the **Datakeys** folder. Select the datakey you want to set and display the **Properties** window by clicking on the **Show properties** icon .

Property Value

Datakey	HISTORY
Prefix	H-
Key	concat(parent::*/@STOCK_ID, @HISTORY_ID)

With these settings the data above would therefore include the new **AL_ID** attribute:

```
<STOCK STOCK_ID='S01' ... >
```

```
<HISTORY HISTORY_ID='H01' AL_ID='H-S01H01'... />
<HISTORY HISTORY_ID='H02' AL_ID='H-S01H02'... />
<HISTORY HISTORY_ID='H03' AL_ID='H-S01H03'... />
</STOCK>
<STOCK STOCK_ID='S02' ... >
  <HISTORY HISTORY_ID='H01' AL_ID='H-S02H01' ... />
  <HISTORY HISTORY_ID='H02' AL_ID='H-S02H02'... />
</STOCK>
```

Creating Datapools and Data Updates for the Stock Application

Defining a Datapool

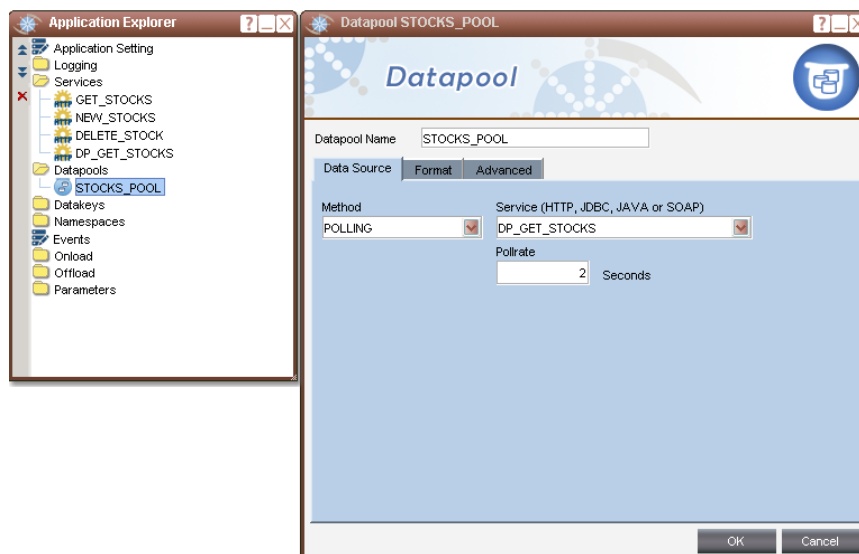
To enable live updates to our Stocks data, we must add a Datapool to the configuration. The Datapool will use a Service Function to get the data, so first we will create the Service Function:


1. Open the Application Manager.
2. Expand the **Services** folder and copy the Service Function **GET_STOCKS** by doing one of the following:
 - Right-click on the **GET_STOCK** Service Function and select **Clone GET_STOCK** in the context menu.
 - Select the **GET_STOCK** Service Function and click on **Edit | Clone GET_STOCK** menu item.
3. Double-click on the new Service function to launch the **HTTP Service** window.
4. In the **Service name** field enter **DP_GET_STOCKS**.
5. On the **Request** tab enter a new parameter mapping:

REQUEST TAB	
Name	Value
TIMESTAMP	\${datapool.timestamp}

Please note: if you want to test this service, you will need to specify a datapool timestamp parameter of **0**. This will return all the data.

6. Now create a new Datapool by right-clicking on the **Datapools** folder and selecting **New Datapool** in the context menu.
7. Double-click on the new Datapool to display the **Datapool** window.
8. Change the name to **STOCKS_POOL**.
9. On the **Data Source** tab, select our new Service Request **DP_GET_STOCKS** from the **Service** drop-down menu.




10. Click on the **OK** button.
11. Save  .

Adding the new STOCKS_POOL in the Designer

Although we have created the Datapool **STOCKS_POOL** we have to configure our application to subscribe to it. This is done within the Designer.

Returning to the AltioLive Designer we have just one more step to enable the Datapool.

1. Load the **Stocks** application in the Designer.
2. From the **View explorer** window expand the **Initial Data Services** folder.
3. Select the **GET_STOCKS** Service and click on the **Show Properties** icon .
4. Expand the **General** properties and from the **Subscribe** drop-down menu, select **STOCKS_POOL**.

Client Update

Introduction to Client Update

The Datapool updates are either polled from the Presentation Server by the Client or streamed to the Client. In the case of polling, the Client polls the Presentation Server using its current Timestamp(s) (either the Initial Data or last successful poll). In response to the poll, the Presentation Server will send all updates (for all Datapools subscribed to by the Client) subsequent to the Timestamp.

For streaming, the Presentation Server maintains a queue of Clients that are subscribed for streamed updates. As the Datapool is updated, the relevant updates are added to a queue for each Client.

The preferred mechanism is HTTP streaming as this minimizes update latency. However, as this utilizes a connection per client, HTTP polling can support a larger number of Clients where there is a less immediate update requirement. If polling is used and no connection can be established (to request new data, save user preferences, update data in the Back end application), then the Client retries after a number of seconds and the user gets an error message.

Please Note: When running from within the designer, you will be using a 5 second polling connection irrespective of the setting applied in the designer.

The Client Queue is fixed length (16 entries). In the event of a queue being full (only likely if the Back end application generates updates at a high rate and/or the Client is subscribed to many Datapools), then the Datapool updating will suspend for up to 30 seconds to allow the queue to clear. If the queue is still full after 30 seconds, the Client session is terminated. In the event of Client Queue being full/disconnected, warning messages are output the Presentation Server log.

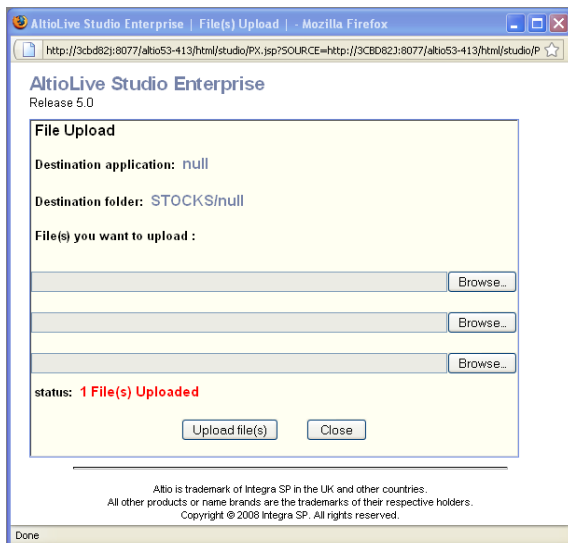
Updating data in the Stock application

When using Altio DB Manager, we can simulate the effect of data updates from a back-end database using an XML file. This would not be used in a real application; it is just a demonstration of how data would update.

The data file containing our update data is called **stocks_updates.xml**. It should be located with the training installation files. You can also download it here:

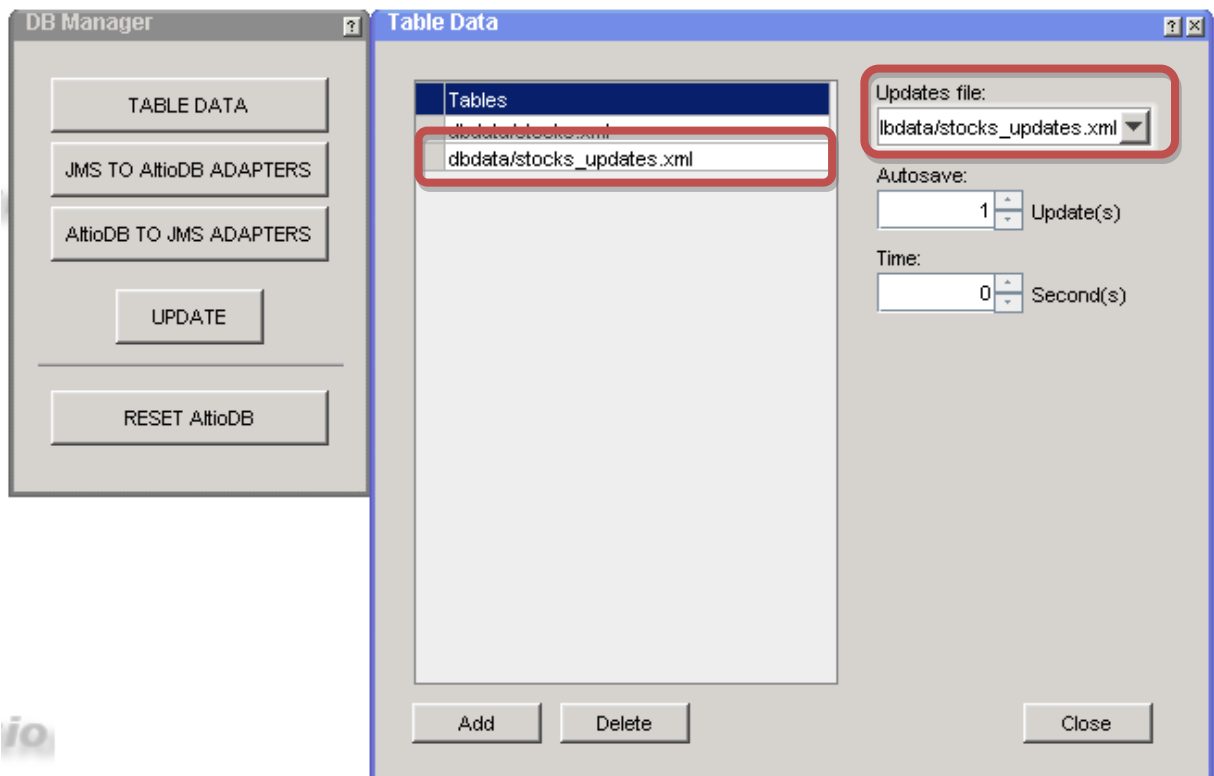
<http://www.altio.com/AltioLive/Online-Documentation.aspx>

1. From the AltioLive Studio, expand the **STOCKS** project.
2. Select the **dbdata** folder and click on the **File | Upload New File** menu item.
3. From the **AltioLive Studio** window, click on **Browse**.
4. Select the **stocks_updates.xml** and click on **Upload file(s)**.
5. Once the message **1 File(s) Uploaded** displays click on **Close**.



Now we can use the Altio DB Manager to specify this file within the application.


6. From the **AltioLive Studio**, right-click on the **STOCK** project and select **AltioDB** in the context menu.
7. From the **Altio DB** window, click on **TABLE DATA**.
8. Click on the **Add** button. In the **Tables** panel, an empty row should now be displayed.
9. Click in this row to display a drop-down menu.
10. From the drop-down menu, select the **stocks_updates.xml** file.
11. From the **Updates file** drop-down menu, select **stocks_update.xml**.



12. Click on the **Close** button.
13. Click on the **UPDATE** then **RESET AltioDB** buttons.

The data file **stocks_updates.xml** has now been integrated into the application.

We are already subscribing to the datapool **STOCKS_POOL**. This datapool will also be used to provide all the clients with the stock updates.

14. Return to the Designer and run the application by clicking on  to see the data updates. From the **Sector** drop-down menu, select **Banks** and from **Market** select **All** since all the updates have been applied to the Banking stocks. The data in the **Mid Price** column should change.

Using Traffic Lighting property and Format Rules

The appearance of a row or cell in a list control can be modified when data updates are received. To enable this feature, we define a font that is enabled if a value increases, and another that is enabled if a value decreases. This definition is called a Format Rule.

To set up a format rule on the Stocks list, we will add new fonts, define the rule and then apply the rule to the Stocks List control.

Using Format Rule

1. From the Designer, expand the **Font style** folder.
2. Create two new font styles by Cloning **CONTROL** font style.
3. For the first new font style, set its properties as follow:

Property	Value
▼ General	
Name	GREEN_CONTROL
Color	0x339966

4. For the second new font style, set its properties as follow:

Property	Value
▼ General	
Name	RED_CONTROL
Color	0x990000

5. Create a Rule by either:
 - selecting the **Rules** node and clicking **New** to create a new rule.
 - right-clicking on the **Rules** folder and selecting **New rule** in the **View explorer** window.
6. Set the properties of the rule to:

Property	Value
▼ General	
Name	HIGHLIGHT_RULE
Positive font	GREEN_CONTROL
Negative font	RED_CONTROL

7. Now look at the **Properties** for the **MID_PRICE** column of the **STOCKS_LIST**.
8. Change the properties of the column as follows:

Property	Value	Comments
▼ Traffic Lighting		
Format rule	HIGHLIGHT_RULE	The rule you have created
Format rule column	MID_PRICE	The column to which the rule is applied

- Run the application and see what effect this has on the appearance of the data. The positive values should be green and the negative ones red:



- The rule can be applied to an alternative column so that the rule is still evaluated against the **MID_PRICE** value but causes a different column to change color. In this situation **HIGHLIGHT_RULE** would be specified in a different column but the rule attribute value would remain as **MID_PRICE**.

Using Update Rule

As well as rules to permanently change the appearance of the data we can set rules that change the appearance temporarily. Here we use the Update Rule and Update Rule Attribute instead.

1. Create two new fonts by Cloning the **CONTROL** font.
2. For the first new font style, set its properties as follow:

Property	Value
▼ General	
Name	UPDATE_GREEN
Color	0xFFFFFFFF
Background color	0x339966

3. For the second new font style, set its properties as follow:

Property	Value
▼ General	
Name	UPDATE_RED
Color	0xFFFFFFFF
Background color	0x990000

4. Create another rule.
5. Set the properties of the rule to:

Property	Value
▼ General	
Name	UPDATE_RULE
Positive font	UPDATE_GREEN
Negative font	UPDATE_RED

11. Now look at the Properties for the **MID_PRICE** column of the **STOCKS_LIST**.
12. Change the properties of the column as follows:

Property	Value	Comments
▼ Traffic Lighting		
Format rule column	MID_PRICE	The column to which the rule is applied
Update rule	UPDATE_RULE	The rule you have created

6. Now in the **View Explorer**, select **View Setting** and set the properties as follows:

Property	Value	Comments
▼ Behavior		
Update timeout (secs)	2	The update format will be applied for 2 seconds

15. Run the application and select **Banks** and **All** from the **Sector** and **Market** drop-down menus. As values change, the Update fonts are used. They remain for the two-second Update Timeout period, then revert to the fonts specified in the Format Rule:

Summary of Stock

Sector: Market:

Stock List

EPIC	NAME	SECTOR	MARKET	MID_PRICE	PERC_CHANGE
ANL	Abbey National	Banks	LONDON	-111	0.5
AL	Alliance & Leic	Banks	NASDAQ	222	1.2
BSCT	Bank of Scotland	Banks	STOCKHOLM	-333	4
BARC	Barclays Bank	Banks	HANGSENG	444	-0.8
BB	Bradford & Bingley	Banks	WSTREET	555	0
EGG	Egg Wli	Banks	NIKKEI	-666	-0.3
HFX	Halifax Grp	Banks	HANGSENG	777	4.2
HSBA	HSBC Holdings	Banks	DJONES	-888	-0.3
LLOY	Lloyds-TSB	Banks	NIKKEI	999	-0.7
NWB	Nat West Bank	Banks	HANGSENG	-111	0
NRK	Northern Rock	Banks	AMEX	222	-1.8

It is also possible to extend Rules by adding Conditions to them. This enables you to set fonts for specific values in columns, rather than just positive and negative numbers.

Altio software does not perform detailed business logic, so more complex rules should be implemented at the server and special attribute values sent to the client for use in Rules. These attribute values would be put in hidden (zero length) columns and compared in the Rules.

Next Step: Using JDBC, SOAP and Java Service Functions

AltioLive Developer Training

Document Information

KEYWORDS: Datapool, Datakeys, Format rules.

Integra SP – Altio

Telephone: +44 (0) 20 8528 1045

Internet: www.altio.com

Copyright © 2010 Integra SP

Copyright in this document is vested in Integra SP. The contents of the document (wholly or in part) must not be reproduced, distributed, used or disclosed without the prior written permission of Integra SP.

Integra recognizes the trademarks or registered trademarks of any third party product or company name referenced in this document at the time of its publication.