



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start Authoring](#)[StreamSQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > Samples Guide

## Samples Guide

StreamBase provides sample EventFlow applications to demonstrate key product features and to help you learn how to use them. Each sample has its own emphasis. For example, a sample might highlight a particular type of operator, while another may describe the configuration details for a distributed environment.

### Contents

- [Loading Samples in Studio](#)
  - [Application Samples](#)
  - [Operator and Data Construct Samples](#)
  - [Client API Samples](#)
  - [Embedded Adapter Samples](#)
  - [External Adapter Samples](#)
  - [Extending StreamBase Samples](#)
  - [High Availability Samples](#)
  - [Tutorial Samples](#)
- 
- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
  - [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > Loading Samples in Studio

## Loading Samples in Studio

### Contents

[Loading Samples into StreamBase Projects](#)

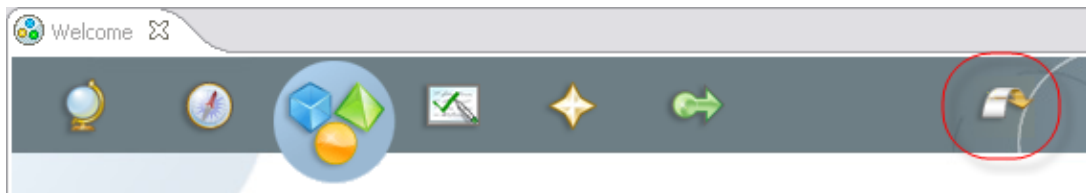
[Sample Locations](#)

[Rebuilding Samples on Windows](#)

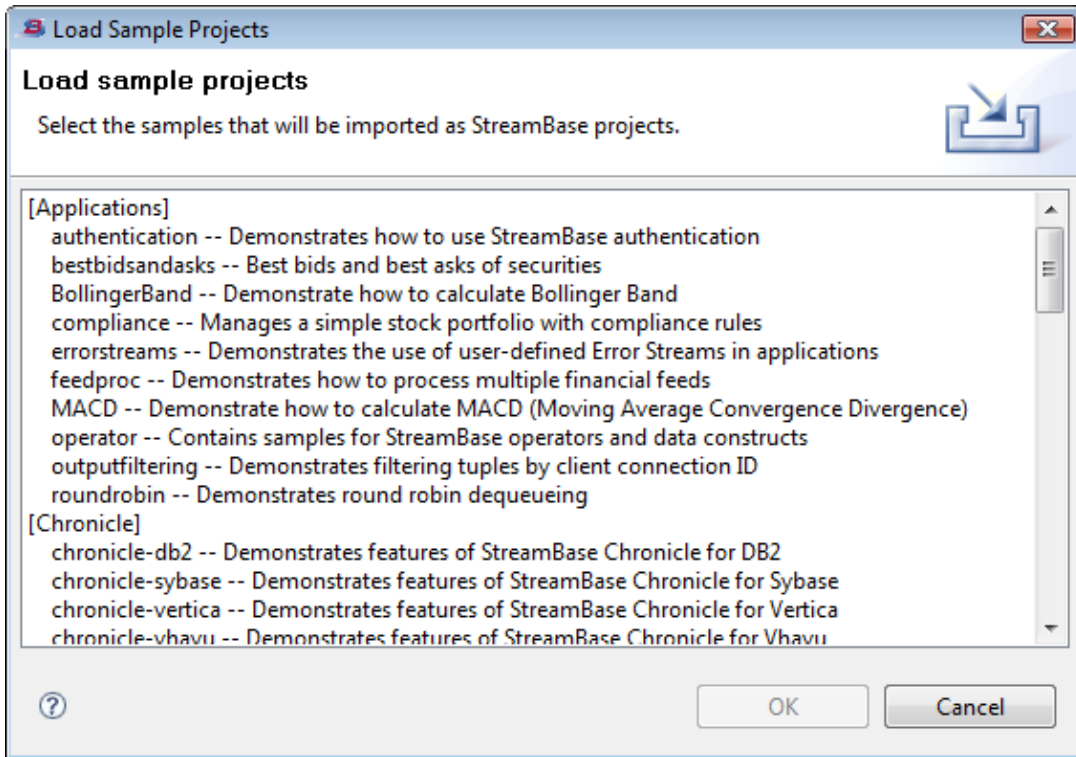
## Loading Samples into StreamBase Projects

To load a sample into StreamBase Studio:

1. If you currently have one of the Welcome pages open in Studio, click the Workbench icon to close the Welcome page and open the SB Authoring Perspective. The Workbench icon is circled in red in the following:



2. From Studio's top-level menu, select File > Load StreamBase Sample.
3. Select one or more samples from the Load Sample Projects dialog, illustrated below. Hold the **Ctrl** key while clicking to make more than one selection.



4. Click OK.

The samples you load have the following characteristics:

- StreamBase Studio creates a separate project for each sample.
- **Exception:** for the `operator` samples, Studio creates one project and all the individual operator's sample EventFlow applications are imported into the same project folder.
- You can reload any sample. If your workspace already has a copy of that sample, Studio creates a new copy, appending a number to the new copy's folder name. This lets you load several copies of the same sample to experiment in different ways.
- If a sample contains a custom function or client application, Studio automatically loads those and any specified configuration settings onto StreamBase Server when you run the sample.
- When you load a sample that contains JAR files, the JAR files are automatically used when you run the sample. The Java source files that created the JAR are included, so that you can see the code and experiment with it, but you do not need to build the JAR before running the sample. When Studio loads a JAR file as part of a sample, it automatically adds the StreamBase Client API to the project's Java build path.

## Sample Locations

By default, the StreamBase sample files are installed in:

- On Windows: `C:\Program Files\StreamBase Systems\StreamBase.n.m\sample\...`
- On UNIX: `/opt/streambase/sample/...`

When you load a sample into StreamBase Studio, Studio copies the sample project's files to your Studio workspace. StreamBase Systems recommends that you use the workspace copy of the sample, especially on

## Samples Guide

UNIX, where you may not have write access to `/opt/streambase`. In the default installation, the path to this sample in your Studio workspace is:

**UNIX:**

```
~/streambase-studio-n.m-workspace/
```

**Windows XP:**

```
C:\Documents and Settings\username\My Documents\StreamBase Studio n.m Workspace\
```

**Windows Vista:**

```
C:\Users\username\Documents\StreamBase Studio n.m Workspace\
```

## Rebuilding Samples on Windows

A number of StreamBase samples, such as `compliance` and `buffering`, have C++ and Java code that can be modified and rebuilt. In order to rebuild the C++ executables and Java JAR files on Windows, configure Microsoft Visual Studio as follows:

1. Add the following StreamBase `include` directories to the list of Include files in Tools > Options > Projects > VC++ Directories:

```
streambase-install-dir\include
```

2. Add the Sun JDK's `bin` directory to the list of executable files.

For example, add the following folder to the list:

```
C:\Program Files\Java\jdk1.6.0_04\bin
```

- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > Application Samples

## Application Samples

### Overview

This group of samples includes EventFlow applications that focus on basic StreamBase functionality and solving business problems.

- The [Authentication Sample](#) — Demonstrates what happens when you enable StreamBase authentication and then try to use several **sb\*** StreamBase commands, such as **sbadmin shutdown**.
  - [Best Bids and Asks Sample](#) — Shows an example of a common stock trading application that tracks the highest bid price and lowest ask price for each stock.
  - [BollingerBand Sample](#) — Shows an example of a common analytical tool that provides a relative definition of high and low bands in securities trading.
  - [Compliance Sample](#) — Shows an example of a common stock fund application, where the StreamBase operators are used to ensure that funds are in compliance with their business requirements.
  - [Error Streams Sample](#) — Demonstrates how to use Error Ports and Streams.
  - [FeedProc Sample](#) — Shows an example of an application that processes financial tick data from three financial feeds, performing various useful computations on the data.
  - [MACD Sample](#) — Shows an example of a momentum indicator: a program that follows trends and shows the relationship between two moving averages of prices.
  - [Output Filtering Sample](#) — demonstrates the use of output filtering to allow StreamBase client applications to receive only the tuples that they are supposed to receive, based on a keyword.
  - [Round Robin Dequeuer Samples](#) — demonstrates the use of round robin dequeuer applications, which control the destination of output tuples based on the connected clients, so that each output message is delivered to a single, subscribed client.
- 
- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
  - [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > Operator and Data Construct Samples

# Operator and Data Construct Samples

## Overview

This section introduces the samples for using operators and data constructs found in the Palette view in StreamBase Studio.

An operator is a StreamBase processing unit that performs pre-defined work on streaming data, such as aggregating windows of data streams, merging streams, or retrieving shared data from a table. A data construct is a component that can store information from a stream or from an external data source that can then be used by an associated StreamBase operator.

A good way to learn about these StreamBase components is to go through the individual samples listed here.

## Operators

Aggregate Operator:

- [Aggregate Operator Time Dimension Sample](#): Demonstrates an aggregate with a time-based dimension.
- [Aggregate Operator Tuple Dimension Sample](#): Demonstrates an aggregate with a tuple-based dimension. Calculates a moving average of price for each of four successive trades of a given stock.
- [Aggregate Operator Field Dimension Sample](#): Demonstrates an aggregate with a field-based dimension. Sums the volume of trades in a particular stock over 30 second windows, advancing every 30 seconds.
- [Aggregate Operator Predicate Dimension Sample](#): Demonstrates an aggregate with a predicate-based dimension.
- [Aggregate Operator Two-Dimension Sample](#): Demonstrates an aggregate with two dimensions

### [BSort Operator Sample](#)

Uses a Bsort operator to reorder trades by time for an input stream where trades arrive in random order. It is instructive to compare the results of running this application with 2 passes versus 6 passes.

### [Filter Operator Sample](#)

Uses a Filter operator to pass only trades of 10,000 shares or more.

### [Gather Operator Sample](#)

Uses a Gather operator to generate an alarm (a "true") for trades of 10,000 shares or more for a stock

## Samples Guide

whose price is \$200 or greater.

Heartbeat Operator:

- Heartbeat Aggregate Sample: Uses a Heartbeat operator to output the number of stock quotes received every five seconds. The actual counting is done by an Aggregate operator, but it relies on the Heartbeat to trigger output during periods when no quotes are received.
- Heartbeat Merge Sample: Uses two Heartbeat operators to prevent a Merge operator from starving when either of its inputs stop receiving tuples.

Join Operator Sample

Uses a Join operator to combine trades from a Reuters and a Comstock feed, where the stocks are the same and the price per share is greater than or equal to \$1, over a period of 60 seconds.

Java Operator

An example of using the Java operator is described in Java Operator Sample.

Lock and Unlock Operators Sample

Uses paired Lock and Unlock operator to protect shared data against concurrent writes. The sample is intended only to demonstrate how to protect the data, and does not include other concurrent operations which might access the data. Other concurrent operations could derive from data on different streams sharing the Query Table, or they could be generated by tuples on the same stream, which would have to wait for the tuple being protected by the Lock clearing the Unlock. (There would be additional operator between the Lock and Unlock operator to process the data.) This example uses the same key for both the key into the Query Table and the key on which to lock for exclusive access.

Map Operator Sample

Uses a Map operator to accept tuples from a comma-separated list, and demonstrates the use of the Map operator to reorder and drop output fields, as well as calculate new output fields.

Merge Operator Sample

Uses a Merge operator to interlace trades from a Reuters and a Comstock feed, based on trade time.

Metronome Operator Sample

Uses a Metronome operator to output the current share price of all stocks every five seconds. The current prices are stored in a query table.

Pattern Operator Sample

Uses a Pattern operator to pass in only market data for securities that have already been traded.

Query Operator:

- Query Sample: Uses a Query Table data construct and Query operators to provide lookup capabilities for the name and description of a stock in the Nasdaq100 index, given its symbol.
- Query TopN Sample: Uses a Query operator to repeatedly read all the rows in a Query Table, and then uses a series of other operators to process the result, compare prices across stock symbols, and compute the top  $n$ .

Sequence Operator Sample

Uses a Sequence operator to generate unique ID numbers for generic product orders, allowing for easy querying when updating orders.

Split Operator Sample

Uses a Split operator to explicitly declare the order in which copies of a tuple are processed by downstream components. This type of configuration is seen when a component has a single output port, with multiple output arcs connected to separate downstream paths.

Union Operator Sample

Uses a Union operator to interlace trades from a Reuters and a Comstock feed.

## Data Constructs

JDBC Table Data Construct Sample

## Samples Guide

Contains a JDBC data construct and several Query operators that perform operations on an external database.

### Materialized Window Data Construct Sample

Uses a Materialized Window data construct to view and query data.

- Copyright © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
- Contact Us



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > Client API Samples

## Client API Samples

### Overview

These samples demonstrate how to work with the StreamBase-provided APIs to create client programs:

- [Buffering Sample](#) — Highlights the tuple buffering options for producer or enqueue clients.
- [Client Sample](#) — Provides sample enqueue and dequeue clients; that is, enqueueing (adding) tuples onto a named stream, and dequeuing (reading) processed tuples from a named stream. The samples include both Java or C++ sources.
- [StreamBase .NET Client Sample](#) — Provides sample .NET enqueue, dequeue clients, and a monitor client that displays system, operator, and thread information. This sample requires the StreamBase .NET Client API, which is installed only for Windows.
- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > Embedded Adapter Samples

## Embedded Adapter Samples

### Overview

StreamBase provides both input and output embedded adapters. Input adapters convert data from external resources into StreamBase tuples, while output adapters convert from StreamBase tuples to external formats. Embedded adapters run in the same server process that is hosting and processing your StreamBase application's requests. See [Adapters Guide](#) for descriptions of each adapter. Sample applications to illustrate the use of embedded adapters are installed in

`streambase-install-dir/sample/adapter/embedded`.

### Embedded Input Adapters

The following pages describe how to run the samples for the embedded input adapters:

- [ActivFeed Input Adapter Sample](#)
- [Alpha Trading System EMAPI Market Data Feed Adapter Sample](#)
- [Alpha Trading System EMAPI Order Entry Adapter Sample](#)
- [Comstock Input Adapter Sample](#)
- [CSV File Reader Input Adapter Sample](#)
- [CSV Socket Reader Input Adapter Sample](#)
- [Deutsche Bank AutobahnFX Trading System Adapter Sample](#)
- [FIX Adapter Sample](#)
- [Hotspot FX Trading System Adapter Sample](#)
- [HTTP Reader Input Adapter Sample](#)
- [InfoReach TMS Adapter Sample](#)
- [IRC Reader Input Adapter Sample](#)
- [JMS Input Adapter Sample](#)
- [Lime Citrius Quote Input Adapter Sample](#)
- [Once Input Adapter Sample](#)
- [Opentick Input Adapter Sample](#)
- [POP3 Reader Input Adapter Sample](#)
- [Regular Expression File Reader Input Adapter Sample](#)
- [Regular Expression Socket Reader Input Adapter Sample](#)
- [Reuters RMDS Subscribing Input Adapter Sample](#)

- [Reuters Subscribing Input Adapter Sample](#)
- [RSS Reader Input Adapter Sample](#)
- [SMTP Reader Input Adapter Sample](#)
- [StreamBase to StreamBase Input Adapter Sample](#)
- [TIBCO Rendezvous Adapter Sample](#)

## Embedded Output Adapters

The following pages describe how to run the samples for the embedded output adapters:

- [Alpha Trading System EMAPI Order Entry Adapter Sample](#)
  - [Adobe Flex Output Adapter Sample](#)
  - [CSV File Writer Output Adapter Sample](#)
  - [CSV Socket Writer Output Adapter Sample](#)
  - [Deutsche Bank AutobahnFX Trading System Adapter Sample](#)
  - [E-mail Sender Output Adapter Sample](#)
  - [FIX Adapter Sample](#)
  - [Hotspot FX Trading System Adapter Sample](#)
  - [Log Output Adapter Sample](#)
  - [Reuters RMDS Publishing Output Adapter Sample](#)
  - [TIBCO Rendezvous Adapter Sample](#)
  - [XML File Writer Output Adapter Sample](#)
  - [XML Over HTTP Output Adapter Sample](#)
- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > External Adapter Samples

## External Adapter Samples

### Overview

External adapters are those that run in independent processes separate from the StreamBase Server process. External adapters are provided in separate installation kits.

The following pages describe how to run the samples for certain StreamBase external adapters:

- [Microsoft Excel Adapter Sample](#)
- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > Extending StreamBase Samples

## Extending StreamBase Samples

### Overview

You can extend StreamBase by writing your own custom clients, functions, embedded adapters, and Java operators. The following samples are provided:

- [Custom Java Simple Function Sample](#) — Demonstrates how to use a custom simple function written in Java.
- [Custom C++ Simple Function Sample](#) — Demonstrates how to use a custom simple function written in C++.
- [Custom Java Aggregate Function Sample](#) — Demonstrates how to use a custom aggregate function written in Java.
- [Custom C++ Aggregate Function Sample](#) — Demonstrates how to use a custom aggregate function written in C++.
- [Java Operator Sample](#) — Demonstrates how to use a custom operator written in Java. Shows a simple `StringCase` class that extends the StreamBase Client API's `Operator` class.
- [Custom Embedded Adapter Sample](#) — Illustrates the use of a custom-written embedded adapter.
- [UTF-16 Sample](#) — Demonstrates StreamBase's support for UTF-16 character data, and includes documentation about the `utf16` custom Java functions.
- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[StreamSQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > High Availability Samples

# High Availability Samples

## Overview

The samples in this part demonstrate best practices for implementing highly available clusters of StreamBase Servers. They show how to separate the high availability logic from the rest of the application logic, as well as design patterns that you can use to address specific high availability requirements.

- [High Availability Sample](#) — This sample shows two StreamBase Server instances running together in an active-active high availability cluster. The sample runs the same simple application in two servers, one designated primary and the other designated secondary. The servers coordinate their behavior through the HA Heartbeat adapter to provide highly available access to the output of the application.
- [High Availability Sample 2: Shared Disk Access](#) — This sample shows a design pattern different from the HA sample. For instance it does not use an HA HeartBeat adapter. The HA2 sample shows:
  - How to ensure that an application is running on one or the other of two servers, where the application requires access to a disk-based query table.
  - How to test for the presence of an application in another container and start the application if it is not found.
  - How to use the External Process Command Line operator to run commands specific to high availability design patterns.

The HA2 sample requires StreamBase Enterprise Edition with a license that enables the use of disk query tables.

- [Query Table Replication Sample](#) — This sample shows how to replicate a StreamBase query table on servers in a high availability (HA) cluster. The schemas in the sample table are order schemas typically used in an application for trading.
- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Samples Guide](#) > Tutorial Samples

## Tutorial Samples

### Overview

The samples in this part are provided to support the following tutorials:

- [First Application Sample](#) — Use this sample while stepping through the tutorial in [Getting Started](#). The tutorial is designed for first-time StreamBase Studio users. It steps through the mechanics of using Studio, defining a simple application, submitting generated test data to it, and observing the results.
- [Copyright](#) © 2004-2009 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)