

# Addressing Data Volume, Velocity, and Variety with IBM InfoSphere Streams V3.0

Performing real-time analysis on big data by using System x

Developing a drag and drop Streams application

Monitoring and administering Streams



Mike Ebbers  
Ahmed Abdel-Gayed  
Veera Bhadrans Budhi  
Ferdiansyah Dolot  
Vishwanath Kamat  
Ricardo Picone  
João Trevelin

**Redbooks**





International Technical Support Organization

**Addressing Data Volume, Velocity, and Variety with IBM  
InfoSphere Streams V3.0**

March 2013

**Note:** Before using this information and the product it supports, read the information in “Notices” on page ix.

**First Edition (March 2013)**

This edition applies to IBM InfoSphere Streams V3.0.

**© Copyright International Business Machines Corporation 2013. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	ix
Trademarks .....	x
<b>Preface</b> .....	xi
The team who wrote this book .....	xi
Now you can become a published author, too! .....	xiv
Comments welcome .....	xiv
Stay connected to IBM Redbooks .....	xiv
<b>Chapter 1. Think big with big data.</b> .....	1
1.1 Executive summary .....	2
1.2 What is big data? .....	3
1.3 IBM big data strategy .....	5
1.4 IBM big data platform .....	7
1.4.1 InfoSphere BigInsights for Hadoop-based analytics .....	7
1.4.2 InfoSphere Streams for low-latency analytics .....	8
1.4.3 InfoSphere Information Server for Data Integration .....	9
1.4.4 Netezza, InfoSphere Warehouse, and Smart Analytics System for deep analytics	10
1.5 Summary .....	13
<b>Chapter 2. Exploring IBM InfoSphere Streams.</b> .....	15
2.1 Stream computing .....	16
2.1.1 Business landscape .....	19
2.1.2 Information environment .....	22
2.1.3 The evolution of analytics .....	26
2.2 IBM InfoSphere Streams .....	28
2.2.1 Overview of Streams .....	29
2.2.2 Why use Streams .....	34
2.2.3 Examples of Streams implementations .....	36
<b>Chapter 3. InfoSphere Streams architecture</b> .....	41
3.1 Streams concepts and terms .....	42
3.1.1 Working with continuous data flow .....	42
3.1.2 Component overview .....	43
3.2 InfoSphere Streams runtime system .....	45
3.2.1 SPL components .....	45
3.2.2 Runtime components .....	47
3.2.3 InfoSphere Streams runtime files .....	49
3.3 Performance requirements .....	51
3.3.1 InfoSphere Streams reference architecture .....	52
3.4 High availability .....	55
<b>Chapter 4. IBM InfoSphere Streams V3.0 new features</b> .....	57
4.1 New configuration features .....	58
4.1.1 Enhanced first steps after installation .....	58
4.2 Development .....	59
4.3 Administration .....	59
4.3.1 Improved visual application monitoring .....	59
4.3.2 Streams data visualization .....	59

4.3.3	Streams console application launcher	59
4.4	Integration	60
4.4.1	DataStage integration	60
4.4.2	Netezza integration	60
4.4.3	Data Explorer integration	60
4.4.4	SPSS integration	60
4.4.5	XML integration	60
4.5	Analytics and accelerators toolkits	61
4.5.1	Geospatial toolkit	61
4.5.2	Time series toolkit	61
4.5.3	Complex Event Processing toolkit	61
4.5.4	Accelerators toolkit	61
<b>Chapter 5.</b>	<b>InfoSphere Streams deployment</b>	<b>63</b>
5.1	Architecture, instances, and topologies	64
5.1.1	Runtime architecture	64
5.1.2	Streams instances	66
5.1.3	Deployment topologies	69
5.2	Streams runtime deployment planning	72
5.2.1	Streams environment	72
5.2.2	Sizing the environment	73
5.2.3	Deployment and installation checklists	73
5.3	Streams instance creation and configuration	80
5.3.1	Streams shared instance configuration	80
5.3.2	Streams private developer instance configuration	84
5.4	Application deployment capabilities	88
5.4.1	Dynamic application composition	89
5.4.2	Operator host placement	91
5.4.3	Operator partitioning	96
5.4.4	Parallelizing operators	96
5.5	Failover, availability, and recovery	99
5.5.1	Restarting and relocating processing elements	100
5.5.2	Recovering application hosts	102
5.5.3	Recovering management hosts	103
<b>Chapter 6.</b>	<b>Application development with Streams Studio</b>	<b>105</b>
6.1	InfoSphere Streams Studio overview	106
6.2	Developing applications with Streams Studio	107
6.2.1	Adding toolkits to Streams Explorer	107
6.2.2	Using Streams Explorer	111
6.2.3	Creating a simple application with Streams Studio	114
6.2.4	Build and launch the application	125
6.2.5	Monitor your application	131
6.3	Streams Processing Language	136
6.3.1	Structure of an SPL program file	137
6.3.2	Streams data types	138
6.3.3	Stream schemas	139
6.3.4	Streams punctuation markers	140
6.3.5	Streams windows	140
6.4	InfoSphere Streams operators	144
6.4.1	Adapter operators	145
6.4.2	Relational operators	147
6.4.3	Utility operators	148

6.4.4 XML operators . . . . .	151
6.4.5 Compact operators . . . . .	151
6.5 InfoSphere Streams toolkits . . . . .	152
6.5.1 Mining toolkit . . . . .	152
6.5.2 Financial toolkit . . . . .	153
6.5.3 Database toolkit . . . . .	155
6.5.4 Internet toolkit . . . . .	155
6.5.5 Geospatial toolkit . . . . .	156
6.5.6 TimeSeries toolkit . . . . .	156
6.5.7 Complex Event Processing toolkit. . . . .	157
6.5.8 Accelerators toolkit . . . . .	158
<b>Chapter 7. Streams integration considerations . . . . .</b>	<b>161</b>
7.1 Integrating with IBM InfoSphere BigInsights . . . . .	162
7.1.1 Streams and BigInsights application scenario . . . . .	162
7.1.2 Scalable data ingest . . . . .	163
7.1.3 Bootstrap and enrichment. . . . .	164
7.1.4 Adaptive analytics model . . . . .	164
7.1.5 Streams and BigInsights application development . . . . .	165
7.1.6 Application interactions. . . . .	166
7.1.7 Enabling components . . . . .	167
7.1.8 BigInsights summary. . . . .	168
7.2 Integration with IBM SPSS . . . . .	168
7.2.1 Value of integration . . . . .	168
7.2.2 SPSS operators overview . . . . .	169
7.2.3 SPSSScoring operator . . . . .	169
7.2.4 SPSSPublish operator . . . . .	169
7.2.5 SPSSRepository operator. . . . .	170
7.2.6 Streams and SPSS examples. . . . .	170
7.3 Integrating with databases . . . . .	172
7.3.1 Concepts. . . . .	172
7.3.2 Configuration files and connection documents . . . . .	173
7.3.3 Database specifics . . . . .	175
7.3.4 Examples . . . . .	180
7.4 Streams and DataStage . . . . .	182
7.4.1 Integration scenarios. . . . .	182
7.4.2 Application considerations . . . . .	182
7.4.3 Streams to DataStage metadata import . . . . .	183
7.4.4 Connector stage . . . . .	184
7.4.5 Sample DataStage job . . . . .	185
7.4.6 Configuration steps in a Streams server environment . . . . .	185
7.4.7 DataStage adapters . . . . .	186
7.5 Integrating with XML data . . . . .	187
7.5.1 XMLParse Operator . . . . .	187
7.5.2 XMLParse example. . . . .	188
7.6 Integration with IBM WebSphere MQ and WebSphere MQ Low Latency Messaging . . . . .	190
7.6.1 Architecture. . . . .	190
7.6.2 Use cases . . . . .	191
7.6.3 Configuration setup for WebSphere MQ Low Latency Messaging in Streams. . . . .	192
7.6.4 Connection specification for WebSphere MQ. . . . .	194
7.6.5 WebSphere MQ operators . . . . .	195
7.6.6 Software requirements . . . . .	196
7.7 Integration with Data Explorer. . . . .	196

7.7.1 Usage .....	197
<b>Chapter 8. IBM InfoSphere Streams administration .....</b>	<b>199</b>
8.1 InfoSphere Streams Instance Management .....	200
8.1.1 Instances Manager .....	201
8.1.2 Creating and configuring an instance .....	201
8.2 InfoSphere Streams Console .....	208
8.2.1 Starting the InfoSphere Streams Console .....	209
8.3 Instance administration .....	210
8.3.1 Hosts .....	212
8.3.2 Permissions .....	215
8.3.3 Applications .....	219
8.3.4 Jobs .....	221
8.3.5 Processing Elements .....	222
8.3.6 Operators .....	224
8.3.7 Application streams .....	225
8.3.8 Views .....	226
8.3.9 Charts .....	228
8.3.10 Application Graph .....	230
8.4 Settings .....	233
8.4.1 General .....	233
8.4.2 Hosts Settings .....	235
8.4.3 Security .....	238
8.4.4 Host tags .....	239
8.4.5 Web Server .....	239
8.4.6 Logging and Tracing .....	244
8.5 Streams Recovery Database .....	247
<b>Appendix A. Installing InfoSphere Streams .....</b>	<b>249</b>
Hardware requirements for InfoSphere Streams .....	250
Software requirements for InfoSphere Streams .....	251
Operating system requirements for InfoSphere Streams .....	251
Required Red Hat Package Manager for InfoSphere Streams .....	253
RPM minimum versions and availability .....	254
Installing InfoSphere Streams .....	258
Before you begin .....	258
Procedure .....	258
IBM InfoSphere Streams First Steps configuration .....	267
Configure the Secure Shell environment .....	269
Configure InfoSphere Streams environment variables .....	269
Generate public and private keys .....	270
Configure the recovery database .....	271
Verify the installation .....	272
Creating and managing InfoSphere Streams instances .....	273
Developing applications with InfoSphere Streams Studio .....	273
Migrate InfoSphere Streams configuration .....	275
<b>Appendix B. IBM InfoSphere Streams security considerations .....</b>	<b>277</b>
Security-Enhanced Linux for Streams .....	278
Unconfined PEs .....	278
Confined PEs .....	278
Confined and vetted PEs .....	279
SELinux advance setting .....	279
User authentication .....	280



User authorization .....	280
Audit log file for Streams .....	281
<b>Appendix C. Commodity purchasing application demonstration .....</b>	<b>283</b>
Application overview .....	284
High-level application design .....	285
Detail application design .....	286
Application demonstration .....	289
Adding missing toolkit .....	291
Running CommodityPurchasing application .....	293
<b>Glossary .....</b>	<b>303</b>
<b>Related publications .....</b>	<b>305</b>
IBM Redbooks .....	305
Other publications .....	305
Online resources .....	305
Help from IBM .....	306



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

BigInsights™	IBM®	Redbooks (logo)  ®
Cognos®	Informix®	Smarter Cities®
Coremetrics®	InfoSphere®	Smarter Planet®
DataStage®	Intelligent Cluster™	solidDB®
DB2®	Intelligent Miner®	SPSS®
developerWorks®	OmniFind®	System x®
GPFS™	POWER7®	Unica®
Guardium®	PureData™	WebSphere®
IBM Watson™	Redbooks®	X-Architecture®

The following terms are trademarks of other companies:

Netezza, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

We live in an era where the functions of communication that were once completely the purview of landlines and home computers are now primarily handled by intelligent cellular phones. Digital imaging replaced bulky X-ray films, and the popularity of MP3 players and eBooks is changing the dynamics of something as familiar as the local library.

There are multiple uses for big data in every industry, from analyzing larger volumes of data than was previously possible to driving more precise answers, to analyzing data at rest and data in motion to capture opportunities that were previously lost. A big data platform enables your organization to tackle complex problems that previously could not be solved by using traditional infrastructure.

As the amount of data that is available to enterprises and other organizations dramatically increases, more companies are looking to turn this data into actionable information and intelligence in real time. Addressing these requirements requires applications that can analyze potentially enormous volumes and varieties of continuous data streams to provide decision makers with critical information almost instantaneously.

IBM® InfoSphere® Streams provides a development platform and runtime environment where you can develop applications that ingest, filter, analyze, and correlate potentially massive volumes of continuous data streams based on defined, proven, and analytical rules that alert you to take appropriate action, all within an appropriate time frame for your organization.

This IBM Redbooks® publication is written for decision-makers, consultants, IT architects, and IT professionals who are implementing a solution with IBM InfoSphere Streams.

## The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.



**Mike Ebbers** is a Project Leader and Consulting IT Specialist at the ITSO, Poughkeepsie Center. He worked for IBM since 1974 in the field, education, and as a manager. He has been a project leader with the ITSO since 1994.



**Ahmed Abdel-Gayed** is an IT Specialist at the Cairo Technology Development Center (CTDC) of IBM Egypt since early 2006. He has worked on various business intelligence applications with IBM Cognos® Business Intelligence. He also implemented many search-related projects by IBM OmniFind® Enterprise Search. He also has experience in application development with Java, Java Platform, Enterprise Edition, Web 2.0 technologies and IBM DB2®. Ahmed's current focus area is Big Data. He holds a bachelor's degree in Computer Engineering from Ain Shams University, Egypt.



**Veera Bhadrans Budhi** is an IT Specialist in IBM US with expertise in the Information Management brand, which focuses on Information Server, IBM Guardium®, and Big Data. He has 17 years of IT experiences working with clients in the Financial Market Industry such as Citigroup, American International Group, New York Stock Exchange, and American Stock Exchange. He is responsible for designing architectures that supports the delivery of proposed solutions, and ensuring that the necessary infrastructure exists to support the solutions for the clients. He also manages Proof of Concepts to help incubate solutions that lead to software deployment. He works with technologies such as DB2, Oracle, Netezza®, MSSQL, IBM Banking Data Warehouse (BDW), IBM Information Server, Informatica, ERWIN (Data Modeling), Cognos, and Business Objects. He holds a Masters Degree in Computer Science from University of Madras, India.



**Ferdiansyah Dolot** is a Software Specialist at IBM Software Group Information Management Lab Services, Indonesia. He worked with the Telecommunication and Banking industries. His area of expertise includes IBM InfoSphere Streams, IBM Information Server, Database Development, and Database Administration. He also specialized in scripting and programming application development, including Perl, Shell Script, Java, and C++. Currently, he is focusing on Big Data, especially IBM InfoSphere Streams application design and development. He has bachelor degree in Computer Science, Universitas Indonesia.



**Vishwanath Kamat** is a Client Technical Architect in IBM US. He has over 25 years of experience in data warehousing and worked with several industries, including finance, insurance, and travel. Vish is leading worldwide benchmarks and proofs of concept by using IBM Information Management products such as Smart Analytics systems and IBM PureData™ Systems for Analytics.



**Ricardo Picone** is an IBM Advisory IT Specialist in Brazil since 2005. He has expertise with business intelligence architecture and reporting projects. He currently supports a project that provides budget sharing information for the IBM sales teams across the world. His areas of expertise include SQL, ETL jobs, IBM DB2 database, data modeling, and IBM Cognos dashboard/scorecards development. Ricardo is an IBM Certified Business Intelligence Solutions professional and is helping the Brazilian Center of Competence for Business Intelligence (COC-BI) in areas such as IBM Cognos technical support and education. He holds a bachelor's degree in Business Administration from the Universidade Paulista, Brazil.



**João Trevelin** is an Advisory IT Specialist in IBM Brazil. He is an Oracle Certified Associate and an Oracle Certified SQL Expert with over 13 years of experience. His areas of expertise include advance SQL, Oracle Database, SOA Integration, data modeling, and crisis management. He is working on Oracle Hyperion BI solutions with a focus on Hyperion Planning. He worked in several industries, including food, government, airline/loyalty, bioenergy, and insurance. He holds a bachelor's degree in Computer Science from the Universidade Federal do Mato Grosso do Sul Brazil and a post-graduate degree in Marketing of Services at Fundação Armando Alvares Penteado Brazil.

Thanks to the following people for their contributions to this project:

- ▶ Kevin Erickson
- ▶ Jeffrey Prentice  
IBM Rochester
- ▶ Steven Hurley  
IBM Dallas
- ▶ Roger Rea  
IBM Raleigh
- ▶ Manasa K Rao  
IBM India
- ▶ James R Giles  
IBM Watson™
- ▶ Susan Visser  
IBM Toronto

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:  
<http://ibm.com/redbooks/residencies.html>

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks® publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form that is found at this website:

<http://ibm.com/redbooks>

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>





# Think big with big data

In this chapter, we describe the accelerating explosion of big data (also known as Big Data), the definition of big data, the characteristics of big data, and the platform that is used for the harnessing big data.

There are multiple uses for big data in every industry – from analyzing larger volumes of data than was previously possible to drive more precise answers, to analyzing data at rest, data in motion to capture opportunities that were previously lost. A big data platform enables your organization to tackle complex problems that previously could not be solved by using traditional infrastructure.

## 1.1 Executive summary

Today, enterprises are exploring big data to discover facts they did not know before. This exploration is an important task right now because the recent economic recession forced deep changes into most businesses, especially those businesses that depend on mass consumers. By using advanced analytics, businesses can study big data to understand the current state of the business and track still-evolving aspects such as customer behavior.

If you really want more information about what is happening in your business, you need large volumes of highly detailed data. If you truly want to see something you never saw before, it helps to tap into data that is not tapped for business intelligence (BI) or analytics. Some of the untapped data is foreign to you and comes from sensors, devices, third parties, web applications, and social media. Some big data sources feed data unceasingly in real time. Put all of these facts together and you see that big data is not only about giant data volumes, it is also about an extraordinary diversity of data types that are delivered at various speeds and frequencies.

This new information, effectively captured, managed, and analyzed, has the power to enhance profoundly the effectiveness of the government. Imagine a world with an expanding population but a reduced strain on services and infrastructure; dramatically improved healthcare outcomes with greater efficiency and less investment; intensified threats to public safety and national borders, but greater levels of security; more frequent and intense weather events, but greater accuracy in prediction and management. Imagine a world with more cars, but less congestion; more insurance claims but less fraud; fewer natural resources, but more abundant and less expensive energy. The impact of big data has the potential to be as profound as the development of the Internet itself.

The great paradox is that, as big data emerges as a new resource, we struggle to keep pace. We find it difficult to discover, understand, and take advantage of the information it contains, to find those true nuggets of knowledge that can improve the lives of every citizens and change the world. Although there is more data available, our ability to comprehend this data is reduced. The challenge lies in capturing the streams of big data that we need, effectively managing them, and extracting new and relevant insights.

The good news is that not only is it possible to extract value from big data, but the path is relatively straightforward. Leaders across government, academia, and private industry made investments and demonstrated success. So now there are best practices from which we can define the path forward.

Perhaps as important, the path to harnessing the value of big data is now affordable. It is this convergence of the availability of big data, the ability to harness it, and the affordability of doing so that brings companies to an inflection point. Big data complements the existing analysis systems; it does not replace them. So, the time to act is now.

## 1.2 What is big data?

For years, decisions that were made in the business world were driven by access to data and technology. But more recently, the decision factors for many organizations revolve around the use of analytics and big data.

Analytics produces the best results when the information that is brought in is of high quantity and high quality. The more information that you bring in, the better the decision. The explosion of data is measured in petabytes<sup>1</sup>, and soon in zettabytes. Over the years, we are more proficient in how to handle structured data by using different types of massive databases, data warehouses, and data marts. But today, data from new sources is becoming increasingly unstructured. There is a big challenge for organizations on how to handle such huge volumes of data, both structured and unstructured. The ability to handle these volumes of data is what big data is all about.

We are now at the crux of a data explosion with more sources continuously generating data. Where is this data coming from? In Figure 1-1, we show a few examples of the items and sources of this data explosion.

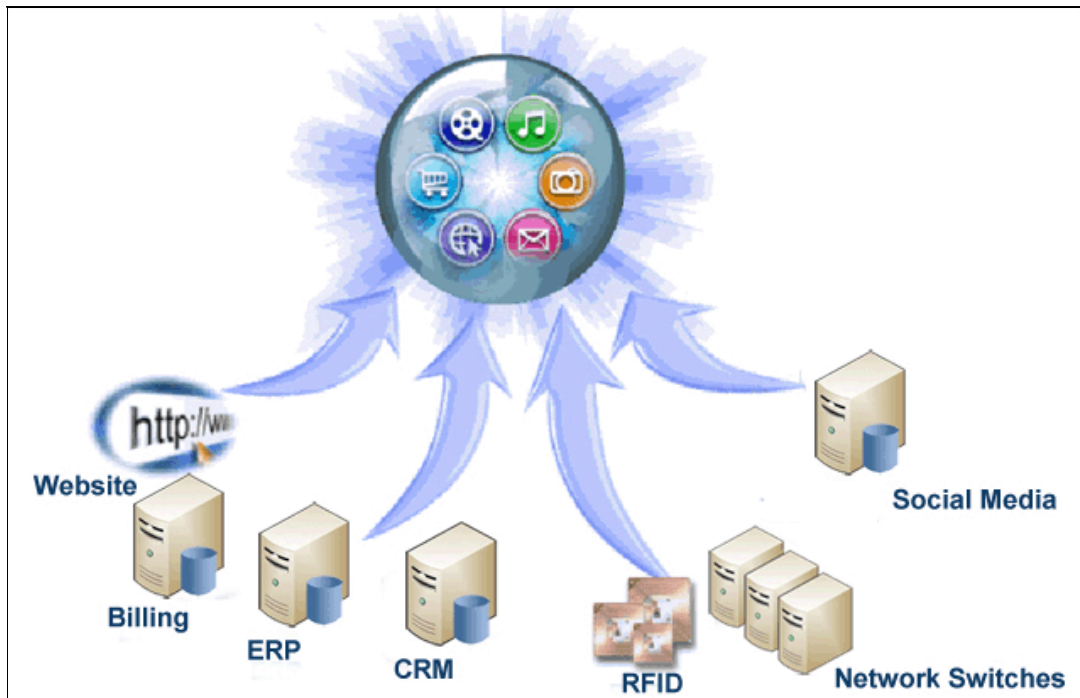


Figure 1-1 Big data explosion

For professionals who work in the realm of information management, there is a good chance that you are aware of the term big data. It is becoming popular to incorporate big data in data management discussions. In a similar way, it was previously popular to bring the advent of service-oriented architecture (SOA) and Web 2.0.

To define the term big data, we explore the evolution of data, along with enterprise data management systems, to ultimately arrive at a clear understanding of not only what big data is, but why you should care. This data comes from everywhere, including sensors that are used to gather climate information, posts to social media sites, digital pictures and videos, purchase transaction records, and cell phone GPS signals.

<sup>1</sup> A quadrillion, or 2 to the 50th power bytes.

What are the characteristics of big data? There is a simple concept that is known as *the Vs*, as shown in Figure 1-2. This concept was first proposed by Gartner group<sup>2</sup> and was widely adopted.

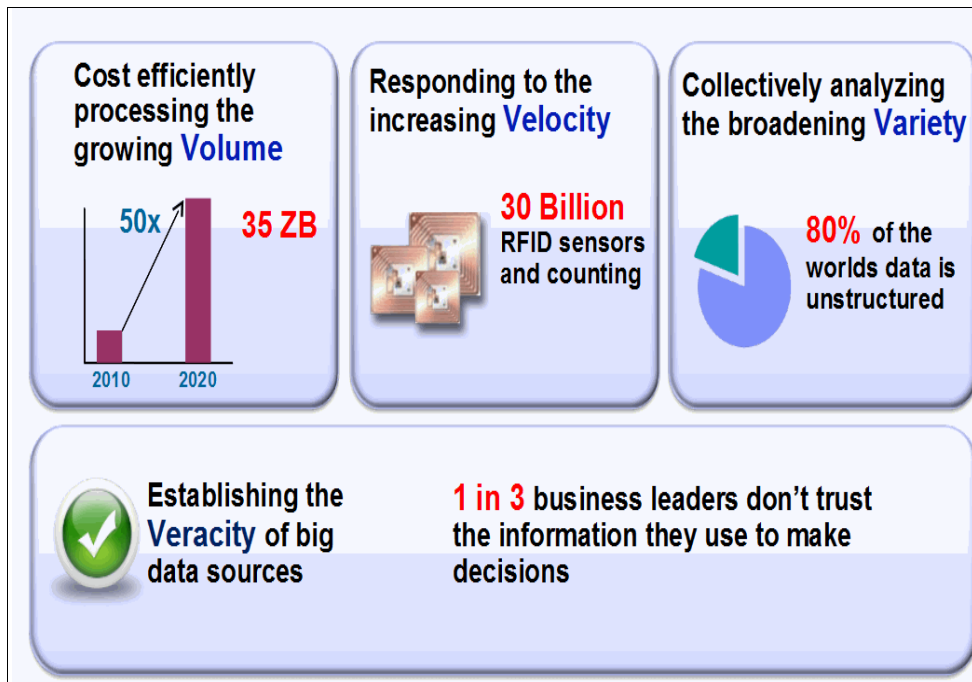


Figure 1-2 Characteristics of big data

The first V is volume. The more data volume that you receive, the harder it becomes for the traditional infrastructure to handle it. Managing that huge flow of data with your current budget might not be economically feasible. Enterprises are awash with ever-growing data of all types, easily amassing terabytes (even petabytes) of information.

The second V is velocity. When this large volume is handled with instruments such as radio frequency identification (RFID) sensors, cellular phones, and social media, the question is what can we do to improve the response time. Some of the data is highly perishable, and its usefulness is over in a short time. Sometimes two minutes is too late. For time-sensitive processes, such as catching fraud, big data must be processed immediately as it streams into your enterprise to maximize its value.

The third V is variety. Big data is any type of data: structured and unstructured data, such as text, sensor data, audio, video, click streams, and log files. New insights are found when these data types are analyzed together. One of the things about data in the world is that we are looking at about 1.6 zettabytes of data. To put that in context, if one cup of coffee was a gigabyte, the Great Wall of China would be 1 zettabyte. So, all of the data in the world is equal to one and one half Great Walls of China. Now, of that 1.6 zettabytes, 80% of that data is unstructured, so there are many novel sources of data that traditional technologies cannot manage or analyze.

There is a notion that, by having this much data in the world and trying to manage and analyze that data, you must be able to trust the resulting information. Bad input equals bad output. In fact, in surveys that are conducted with the CIO, CFO, and CEO of companies, some of the information that was provided by these business leaders is surprising. One in

<sup>2</sup> For more information, see this website:  
<http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>

three business leaders think they make frequent decisions with a lack of trust in the information, and one in two say that they do not have access to information that they need to do their jobs. So, 50% said they cannot do their job as well as they could; not because they are not getting the information that they need for the perfect analysis, but because even though we think that we gave them all of the information that they need, they do not think so. Also, 83% of the CIOs that were surveyed think and see analytics and business intelligence as part of their future roadmap. But how can you act upon information if you do not trust it? Establishing trust in big data presents a huge challenge because the variety and number of sources grows. Veracity becomes the fourth V around big data.

So to understand big data, a simple framework to use is to analyze the Vs.

**Important:** According to the characteristics of big data, currently the officially used Vs are Volume, Velocity, and Variety (3Vs) and Veracity is not yet another official “V” for big data, but it holds true that veracity of data is as important to big data solutions as any prior data management solutions. Veracity refers to the quality and trustworthiness of the data. Big data is so vast that the quality issues are a reality, and veracity is what we generally use to refer this problem domain.

Harvesting and analyzing all of this data to provide a competitive advantage is the challenge that is facing businesses today and in the future. How does an organization extract insight from an immense volume, variety, velocity, and veracity of data in a timely and cost-effective manner? This question is the question and the challenge that is posed by big data.

## 1.3 IBM big data strategy

Big data is more than a matter of size. It is an opportunity to find insights into new and emerging types of data and content to make your business more agile, and to answer questions that were previously considered beyond your reach. Until now, there was no practical way to harvest this opportunity. Today, IBM's platform for big data uses state of the art technologies, including patented advanced analytics to open the door to a world of possibilities.

Organizations must be able to fully use all sources of data and content for insight. Executives must make decisions that are based not only on operational data and customer demographics, but also on customer feedback, details in contracts and agreements, and other types of unstructured data or content. How can you manage all of this data and give executives access to the visually compelling information that they need to make timely, informed decisions?

IBM's big data strategy moves the analytics closer to the data. New analytic applications drive the requirements for a big data platform.

The big data strategy includes some of the following key factors:

- ▶ Integrate and manage the full variety, velocity, and volume of data
- ▶ Apply advanced analytics to information in its native form
- ▶ Visualize all available data for ad hoc analysis
- ▶ Create a development environment for building new analytic applications
- ▶ Workload optimization and scheduling
- ▶ Security and governance

Figure 1-3 shows IBM's big data strategy of moving and integrating the analytics closer to the data. This integration helps the companies by enhancing decision making by bringing all of the information together from multiple sources.

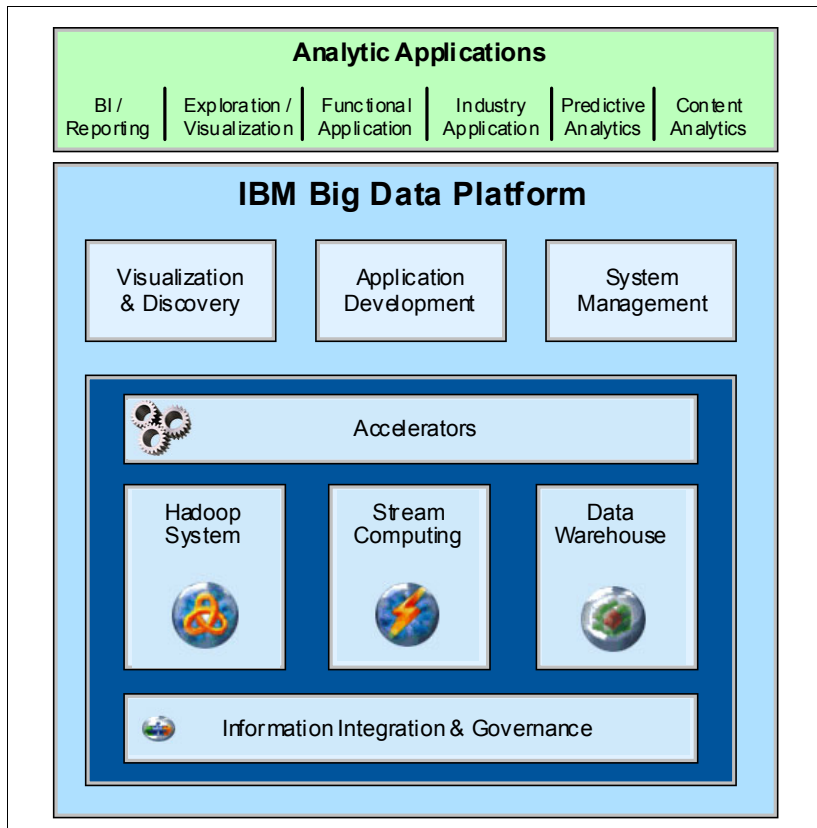


Figure 1-3 Big data strategy

IBM's approach is multi-faceted. IBM is incorporating new capabilities into an existing infrastructure to enable the enterprise to efficiently store and analyze virtually any variety, volume, or velocity of data. No single infrastructure can solve all big data problems. Rather than creating a single product, IBM added products where the addition of new technology can complement the existing data warehouse (DW) architecture and thus provide added value to the enterprise.

## 1.4 IBM big data platform

Big data is not just a technology, it is a business strategy for capitalizing on information resources. Success at each entry point is accelerated by products within the big data platform, which helps in building the foundation for future requirements by expanding further into the big data platform.

The components that comprise the IBM big data platform are shown in Figure 1-4.

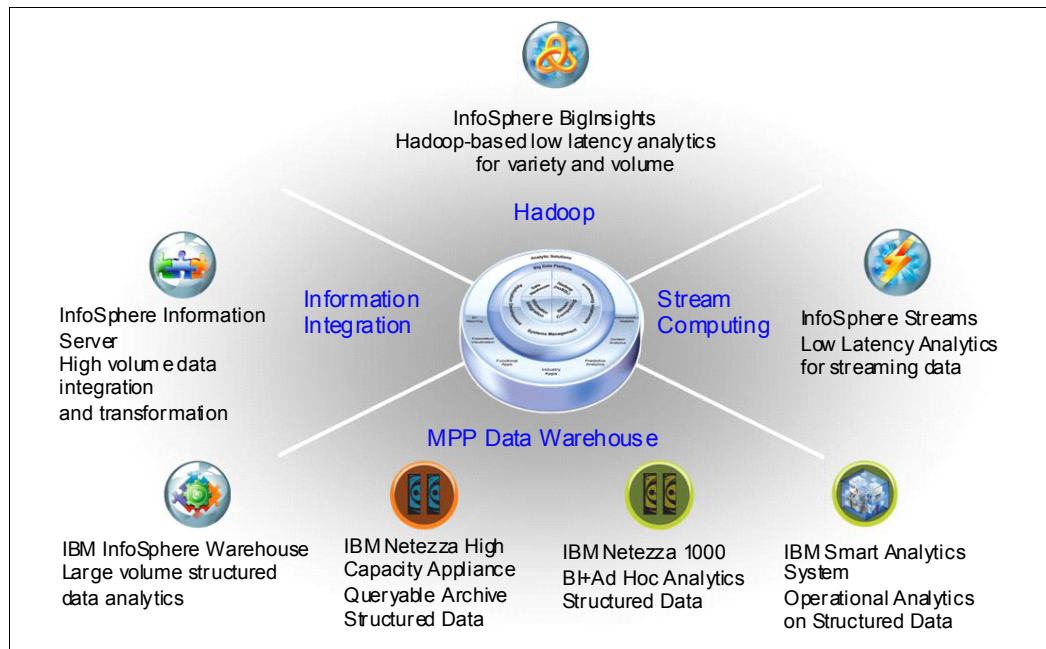


Figure 1-4 IBM big data platform components

### 1.4.1 InfoSphere BigInsights for Hadoop-based analytics

Working with big data is becoming a part of the enterprise data strategy at many firms. A number of organizations want to deploy a software platform, such as IBM BigInsights™ so that they can manage big data from the moment it enters their enterprise. After the raw data is stored in BigInsights, firms can manipulate, analyze, and summarize the data to gain new insights and feed downstream systems. In this manner, the original (raw) data and modified forms of data are accessible for further processing.

IBM incorporated Apache Hadoop into its big data platform. Hadoop is a software framework that is often implemented on a cluster of commodity-based hardware servers to perform distributed computational operations across the hardware in the cluster. Unlike traditional DWs, Hadoop does not require a physical data model and schema to be defined before data is incorporated into the Hadoop cluster. Therefore, it is able to store virtually any data format within its file system, which is known as *Hadoop Distributed File System* (HDFS). To make this option feasible for the enterprise, IBM developed a product called InfoSphere BigInsights. Figure 1-5 on page 8 shows the components of InfoSphere BigInsights.

One potential deployment approach involves the use of BigInsights as a source for a data warehouse. BigInsights can sift through large volumes of unstructured or semi-structured data and capture relevant information that can augment existing corporate data in a warehouse. Figure 1-5 on page 8 shows such a scenario, which offers firms the ability to broaden their analytic coverage without creating an undue burden for their existing systems.

After BigInsights is in the warehouse, traditional business intelligence and query and report writing tools can work with the extracted, aggregated, and transformed portions of raw data that are stored in BigInsights.

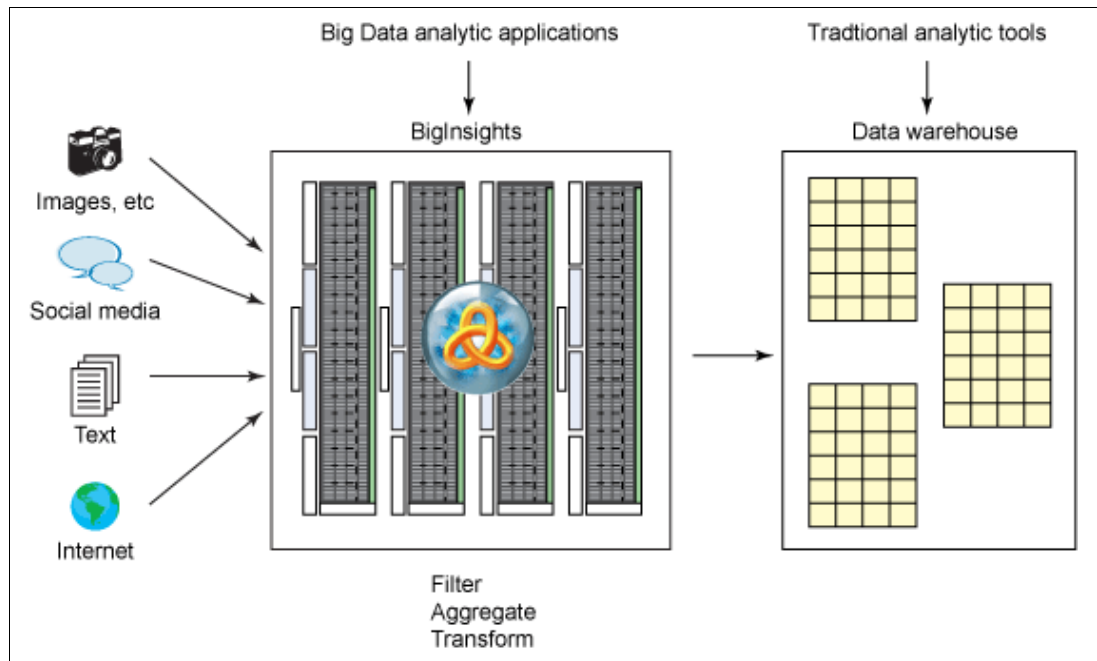


Figure 1-5 Using InfoSphere BigInsights to filter and summarize big data for warehouse

This offering provides a packaged Hadoop distribution, a greatly simplified installation of Hadoop, and corresponding open source tools for application development, data movement, and cluster management. BigInsights also provides other options for data security, which is frequently a point of concern for anyone who is contemplating the incorporation of new technology into their data management ecosystem. BigInsights is a component of the IBM big data platform and as such provides potential integration points with the other components of the platform, including the DW, data integration and governance engines, and third-party data analytics tools. The stack includes tools for built-in analytics of text, natural language processing, and spreadsheet-like data discovery and exploration.

### 1.4.2 InfoSphere Streams for low-latency analytics

IBM InfoSphere Streams is a software platform that enables the development and execution of applications that process information in data streams. InfoSphere Streams enables continuous and fast analysis of massive volumes of moving data to help improve the speed of business insight and decision making.

Streams performs analytics on continuously streaming data before it lands inside the DW. In addition to volume, our definition of big data includes the velocity of data. Streams computing is ideal for high-velocity data where the ability to recognize and react to events in real time is a necessary capability. While there are other applications that are aimed at providing stream-computing capability, the Streams architecture takes a fundamentally different approach to continuous processing, which differentiates it from other platforms. Its distributed runtime platform, programming model, and tools for developing continuous processing applications promote flexibility, development for reuse, and unparalleled performance.



The overview of InfoSphere Streams is shown in Figure 1-6.

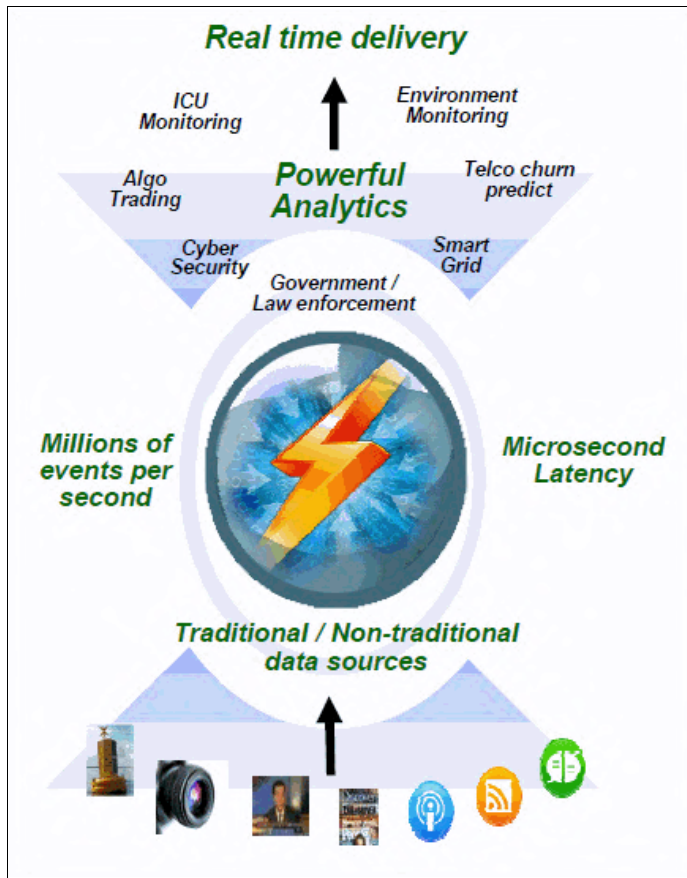


Figure 1-6 InfoSphere Streams

When Streams processing is discussed, it is often associated with Complex Event Processing (CEP). However, Streams and CEP are truly different. Aside from the fact that they operate on real-time data, feature ultra-low latency, and provide event-based, stream processing, InfoSphere Streams potentially outperforms CEP in other aspects.

CEP provides analysis on discrete business events, is rule-based with correlation only across certain event types, features modest data rates, and operates only on structured data. Streams provides simple and complex analytics on continuous data streams, can scale for computational complexity, and supports a wide range of relational and non-relational data types. Streams supports higher data rates and a much broader range of data types than CEP. For example, there are data sources that are consumable by Streams, including but not limited to sensors, cameras, video, audio, sonar or radar inputs, news feeds, stock tickers, and relational databases.

### 1.4.3 InfoSphere Information Server for Data Integration

By using IBM InfoSphere Information Server for Data Integration, you can transform data in any style and deliver it to any system, ensuring faster time to value.

IBM InfoSphere Information Server is a market-leading data integration platform that helps you understand, cleanse, transform, and deliver trusted information to your critical business initiatives, such as big data, master data management, and point-of-impact analytics.

Creating trusted information requires collaboration between IT and business, by using a unified platform to define and integrate data. Organizations must ensure that data is accessible, authoritative, consistent, timely, and in context for analysis and reporting. InfoSphere Information Server delivers features to help you rationalize your increasingly complex environment and meet your evolving information integration needs.

By using InfoSphere Information Servers end-to-end information integration capabilities, you can understand your data, cleanse, monitor, transform, and deliver data, and to collaborate to bridge the gap between business and IT. Since its inception, InfoSphere Information Server provides a highly scalable and flexible integration platform, providing massively parallel processing (MPP) that handles small to large data volumes. Figure 1-7 shows the overall picture of InfoSphere Information Server.

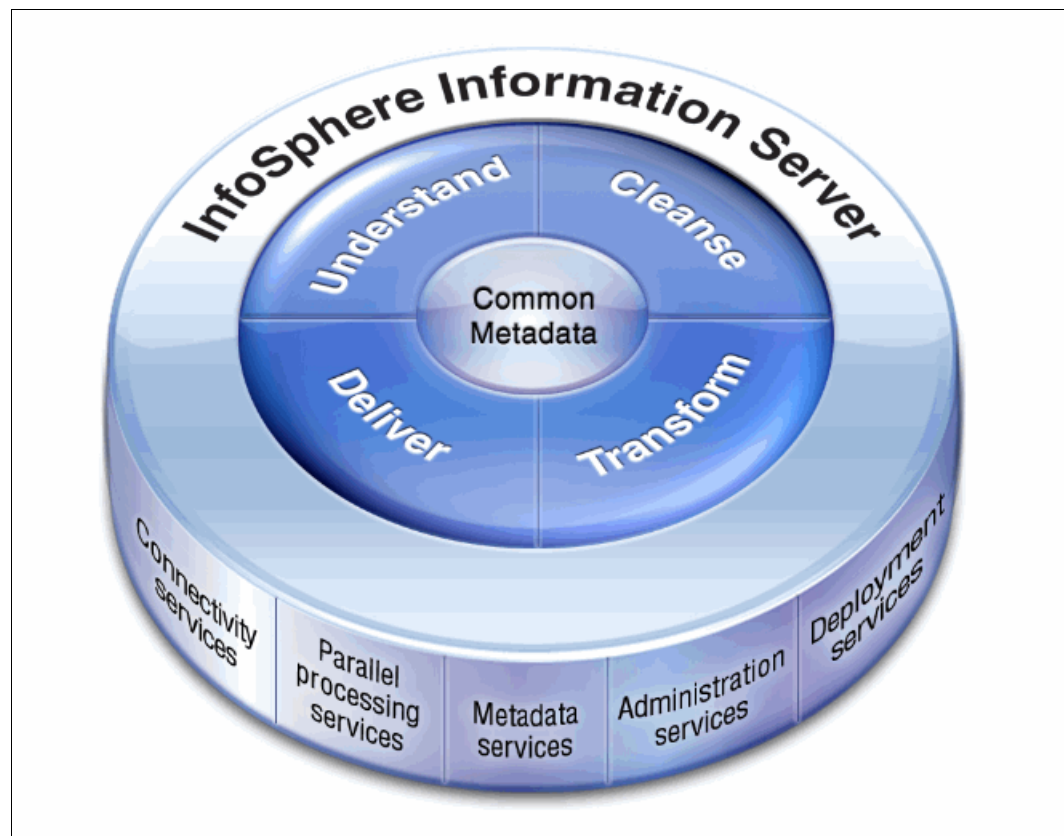


Figure 1-7 InfoSphere Information Server

#### 1.4.4 Netezza, InfoSphere Warehouse, and Smart Analytics System for deep analytics

##### Netezza

Netezza transforms the data warehouse and analytics landscape with a platform that is built to deliver extreme, industry-leading price performance with appliance simplicity. It is a new frontier in advanced analytics, with the ability to carry out monumental processing challenges with blazing speed without barriers or compromises. For users and their organizations, it means the best intelligence for all who need it, even as demands for information escalate. The Netezza data warehouse and analytics appliance's revolutionary design provides exceptional price performance. As a purpose-built appliance for high speed analytics, its strength comes not from the most powerful and expensive components, but from having the correct components assembled and working together to maximize performance.

MPP streams combine multi-core CPUs with Netezza's unique Field Programmable Gate Arrays (FPGA) with Accelerated Streaming Technology (FAST) engines to deliver performance that in many cases exceeds expectations. As an easy-to-use appliance, the system delivers its phenomenal results ready to use, with no indexing or tuning required. Appliance simplicity extends to application development, enabling organizations to innovate rapidly and bring high performance analytics to the widest range of users and processes.

The Netezza appliances integrate database, processing, and storage in a compact system that is optimized for analytical processing and designed for flexible growth. The system architecture is based on the following core tenets that are a hallmark of Netezza leadership in the industry:

- ▶ Processing close to the data source
- ▶ Balanced massively parallel architecture
- ▶ Platform for advanced analytics
- ▶ Appliance simplicity
- ▶ Accelerated innovation and performance improvements
- ▶ Flexible configurations and extreme scalability

### **InfoSphere Warehouse**

InfoSphere Warehouse represents the IBM offering for implementing integrated business intelligence (BI) solutions. The BI framework enables the transformation of the DW from a static repository that is primarily used for batch reporting into an active end-to-end platform for BI solutions. InfoSphere Warehouse integrates design-time tooling and runtime infrastructure for Online Analytical Processing (OLAP) and data mining, and the delivery of embedded Inline analytics on a common platform that is based on the IBM DB2 Relational Database Server and the IBM WebSphere® Application Server.

InfoSphere Warehouse removes cost and complexity barriers, and enables delivery of powerful analytics to an enterprise. The integrated tooling enables a low total cost of ownership (TCO) and provides improved time-to-value for developing and delivering analytics enterprise-wide.

InfoSphere Warehouse is a suite of components that combines the strength of the DB2 database engine with a powerful business analytics infrastructure from IBM. InfoSphere Warehouse provides a comprehensive business analytics platform with the tools that are required so that you can design, develop, and deploy analytical applications in your enterprise. InfoSphere Warehouse can be used to build a complete data warehousing solution that includes a highly scalable relational database engine, data access capabilities, business analytics, and user analysis tools. It integrates core components for data warehouse administration, data mining, OLAP, inline analytics, reporting, and workload management.

### **IBM Smart Analytics System**

Nowadays, enterprises recognize the value of business analytics and are moving to apply these capabilities to add business value. However, implementing a data warehouse solution requires resources and expertise in business intelligence software, server hardware, storage, and the help of professional services. The traditional system implementation method for this complex integration effort costs a company time and money.

IBM Smart Analytics System, taking advantage of the appliance architecture, is a pre-integrated analytics system that is designed to deploy quickly and deliver fast time to value. Because the software is already installed and configured in the server, clients are able to have their systems up and running in days instead of months. Engineered for the rapid deployment of a business-ready solution, the IBM Smart Analytics System includes the following features:

- ▶ A powerful data warehouse foundation
- ▶ Extensive analytic capabilities
- ▶ A scalable environment that is integrated with IBM servers and storage
- ▶ Set-up services and single point of support

To add capacity over time, you can mix various generations of hardware with which you can protect your investment in the long term.

Figure 1-8 shows the concept of IBM Smart Analytics System modules.



Figure 1-8 IBM Smart Analytics System module concept

Every IBM Smart Analytics System offers a set of resources to support a complete data warehousing solution. At the heart of the IBM Smart Analytics System is a data warehouse that is based on DB2 Enterprise Server Edition software and the Database Partitioning Feature that incorporates best practices that are based on decades of IBM experience with designing and implementing data warehouses. The analytics, workload management, and

performance analysis capabilities that are provided by the InfoSphere Warehouse software depend on the specific edition of the software that your offering includes. In most cases, the software includes the following features:

- ▶ Data modeling and design that is provided through Design Studio
- ▶ Data movement and transformation that is provided through the SQL Warehouse Tool
- ▶ OLAP functions provided through Cubing Services
- ▶ OLAP visualization provided through Alphablox
- ▶ In-database data mining that is provided through IBM Intelligent Miner® and MiningBlox
- ▶ Data Mining Visualization provided through Intelligent Miner Visualization
- ▶ Unstructured text analysis that is provided through Text Analytics
- ▶ Integrated workload management that is provided through DB2 workload manager
- ▶ Deep compression for data, index, temporary tables, and XML provided by
- ▶ DB2 Storage Optimization feature
- ▶ Performance tuning and analysis through DB2 Performance Expert

The analytics capabilities that are provided by the optional IBM Cognos 8 BI software include reporting, query, and dashboarding capabilities. By using these capabilities, you can perform complex analysis on your data to identify trends in business performance and represent your insight visually through reports or at-a-glance dashboards.

## 1.5 Summary

This chapter provided a general framework to identify big data and analytics. We introduced the terms volume, variety, velocity, and veracity to give you an understanding of the concepts and, in turn, a better understanding of the capabilities of big data and its opportunities. The big data era is all about enhancing analytics through a purpose-built big data platform that introduces new methods and techniques and synergizes them with as important traditional solutions you have running today. These approaches complement one another, and like the

athlete who coordinates multiple skills for superb performance, so too does the IBM big data platform power up the instantiation of your successful and business-differentiating big data initiatives.

Now that you know about big data and its characteristics and platform, we provide more information about InfoSphere Streams V3.0 in the following chapters:

- ▶ Chapter 2, “Exploring IBM InfoSphere Streams” on page 15 explores how Stream Computing is used for big data. We describe IBM InfoSphere Streams and its capabilities.
- ▶ Chapter 3, “InfoSphere Streams architecture” on page 41 explains the architecture of IBM InfoSphere Streams.
- ▶ Chapter 4, “IBM InfoSphere Streams V3.0 new features” on page 57 lists the new features of IBM InfoSphere Streams V3.0.
- ▶ Chapter 5, “InfoSphere Streams deployment” on page 63 describes the runtime deployment of Streams.
- ▶ Chapter 6, “Application development with Streams Studio” on page 105 describes application development and deployment on InfoSphere Streams and introduce the new features of Streams Studio.
- ▶ Chapter 7, “Streams integration considerations” on page 161 explains how to integrate Streams with other systems such as IBM InfoSphere BigInsights, Netezza, SPSS®, DB2, and others.
- ▶ Chapter 8, “IBM InfoSphere Streams administration” on page 199 provides details on how to perform monitoring and administration in Streams.
- ▶ The Appendix section contains the prerequisites, installation details, security considerations, and the Commodity Purchasing Application demonstration performed on IBM InfoSphere Streams.



# Exploring IBM InfoSphere Streams

In this chapter, we provide background information about an exciting analytic paradigm called stream computing. We introduce you to the concepts of streaming data from the business landscape and informational environment perspectives, and the emergence of real-time analytic processing (RTAP). We then build on this information to show how IBM is actively involved in this extreme shift in the process of decision making. The result of years of research into how to meet this challenge is IBM InfoSphere Streams (Streams). To help position this shift, we give you some history about Streams and how it is designed to provide an optimal platform for applications that use streaming data in the decisions you must make. These decisions translate into actions that you must take for the health of your business and the quality of your life.

This chapter provides insight into some of the following key questions and concepts:

- ▶ What is streaming data?
- ▶ What can be learned from it to improve the functions of our businesses?
- ▶ How can developing streams-based applications enable you to become more proactive in the decision making process?
- ▶ How does Streams enable you to harness the power in the plethora of information that is available from traditional and non-traditional sources of data?

## 2.1 Stream computing

We live in an era where the functions of communication that were once completely the purview of landlines and home computers are now primarily handled by intelligent cellular phones, where digital imaging replaced bulky X-ray films, and the popularity of MP3 players and ebooks are changing the dynamics of something as familiar as the local library.

For years, self-professed information evangelists took every opportunity to expound on the explosive increase in the amount of available data and the challenge to glean key useful information from the deluge in time to take action. Actually, we heard these proclamations so often we became desensitized to the true impact of such a statement. At a time when this rate of increase is surpassing everyone's projections, many of us still feel the challenge to do something with this tsunami of information to make it relevant to our jobs. Unfortunately, we rarely can. The flood of data is pertinent to all aspects of our lives. We cannot treat all that information as so much background noise any longer. It is time to be more than aware that all that information is there. We must integrate it into our businesses, governments, educational processes, and lives. We must use all of this information to make things happen, not just document and review what happened.

This desire probably is not new. We always wanted to use available information to help us make more informed decisions and take appropriate action in real time. The realization of such a lofty goal as truly real-time analysis is challenged by many things over the years. There were limitations in the capabilities of the IT landscape to support the delivery of large volumes of data for analysis in real time, such as processing power, available computing memory, network communication bandwidth, and storage performance. Though some of these constraints still exist, there are significant strides in minimizing or alleviating some or all of these constraints. One of the biggest challenges to presenting data in real time is not directly related to the capabilities of the data center. One of the main challenges is the ability to acquire the actual information in a way that makes it available to the analysis process and tools in time to take appropriate action.

Even today, typical analytical processes and tools are limited to the use of stored (and often structured) data. To accommodate this requirement, data acquisition historically required several time-consuming steps, such as collection through data entry or optical scanning, cleansing, transformation, enrichment, and loading this data into an appropriate data store. The time that it takes for these steps to complete results in a delay before the data is available to be analyzed and used to drive any actionable decisions. Often, this delay is enough that any action that is taken from the analysis is more reactive than proactive in nature.



If there are challenges to acquiring information in real time, we can expect the situation only to get worse. Our world is becoming more instrumented. Because of this phenomenon, traditionally unintelligent devices are now a source of intelligent information. Tiny processors, many with more processing power than the desktop computers of years ago, are infused in the everyday objects of our lives. Everything from the packages of products you buy, to the appliances in our homes, to the cars we drive feature the capability to provide us with information we could use to make more informed decisions. An example of this situation is shown in Figure 2-1.



Figure 2-1 Information is available from formerly inanimate sources

Along with the proliferation of instrumentation in everyday products and processes, we are experiencing a surge in interconnectivity. Not only is the actual data available right from the sources, but those sources are also interconnected in such a way that we can acquire that data as it is generated. The acquisition of data is no longer limited to the realm of passive observation and manual recording. The information that is produced and communicated by this massive wave of instrumentation and interconnectivity makes it possible to capture what is actually happening at the time it happens. Data can be acquired at the source and made available for analysis in the time it takes a machine to communicate with another machine. Where we once assumed, estimated, and predicted, we now can know.

These advances in availability and functionality are the driving force in the unprecedented, veritable explosion in the amount of data that is available for analysis. Are we ready to use this information? Even with the recent improvements in Information Technology, the predicted steady increase in the amount of traditional stored data that is available was a considerable challenge to realizing true real-time analytics. The world can now generate inconceivable amounts of data. This unexpected increase in volume is likely to still outstrip the potential gains we made in technology, unless we begin to change the way we approach analysis. Just to get that amount of data stored for analysis could prove to be an insurmountable task.

**Imagine:** In the next few years, IP traffic is expected to total more than half a zettabyte (that is, one a trillion gigabytes, or a one followed by 21 zeros).

Another challenge is new sources of data generation. The nature of information today is different from information in the past. Because of the profusion of sensors, microphones, cameras, and medical and image scanners in our lives, the data that is generated from these instruments is the largest segment of all information that is available, and about 80% of this information is unstructured. One advantage of this shift is that these non-traditional data sources are often available while they are en-route from their source to their final destination for storage, which means that they are available the moment they happen (before they are stored).

Consider data feeds from local or global news agencies or stock markets. Throughout their active hours of operation, these data feeds are updated and communicated as events happen. Most of us have at least one of the addresses (URLs) for these live data feeds bookmarked on our browser. We are confident that every time we access this address, we can review current data. Conversely, we probably do not have any idea where the historic information goes to be stored. More importantly, there might be no need to know where it is stored because we are able to analyze the data before it comes to rest.

This data that is continuously flowing across interconnected communication channels is considered streaming data or *data in motion*.

To automate and incorporate streaming data into our decision-making process, we must use a new paradigm in programming called *stream computing*. Stream computing is the response to the shift in paradigm to harness the awesome potential of data in motion. In traditional computing, we access relatively static information to answer our evolving and dynamic analytic questions. With stream computing, you can deploy a static application that continuously applies that analysis to an ever changing stream of data. As you look at Figure 2-2, think of the difference between doing research for a report on a current event from the printed material in a library versus the online news feeds.



Figure 2-2 Comparing traditional with stream computing

To help you better understand this concept, consider for the moment that you are standing on the bank of a rapidly flowing river into which you are planning to launch a canoe. You have a wet suit but are not sure if you need to bother with wearing it. You first must determine how cold the water is to decide whether you should put on the wet suit before you get into the canoe. There is certainly no chance of capturing all of the water and holding it while you determine the average temperature of the full set of the water. In addition, you want to leave your canoe while you go to retrieve a newspaper that might have a listing of the local water temperatures. Neither of those options are practical. A simpler option is to stand on the edge of the creek and put your hand or foot into the water to decide whether it was warm enough to forego the wet suit. In fact, you could also insert a thermometer for a more accurate reading, but that level of accuracy is probably not necessary for this scenario. The water's journey was not interrupted by our need to analyze the information; on the contrary, someone else could be testing those same waters farther downstream for their own informational needs.

Much like its watery counterpart in this example, streaming data can be accessed from its edges during its travels without deterring further use downstream or stopping its flow to its final repository.

Looking at the different programming approaches to analysis, you can see correlations to the river, although historically you could capture all of the information before analysis. In traditional computing, you could produce a report, dashboard, or graph that shows information from a data warehouse for all sales for the eastern region by store. You could take notice of particular items of interest or concern in your high-level analysis and drill down into the underlying details to understand the situation more clearly. The data warehouse that is supplying information to these reports and queries is probably loaded on a set schedule; for example, weekly, daily, or even hourly, and is not modified much after the data is committed.

So, the data is growing over time, but historical data is mostly static. If you run reports and queries today, tomorrow, or a month from now on the same parameters, the results are basically the same. Conversely, the analysis in this example is evolving and changing from a summary to details. You could, and surely would, take action that is based on this analysis. For example, you could use it to project replenishment and merchandising needs, but because the analyst must wait until the data is loaded to use, their ability to take those actions in real time is limited. With some analyses and actions, this model is a perfectly acceptable, for example, making plans to update the production process or design a new line that is based on the success of another line. With stream computing, you can focus on an analysis that results in immediate action. You could deploy an analytic application that continuously checks the ever-changing inventory information that is based on the feeds from the cash registers and the inventory control system and send an alert to the stock room to replenish items more quickly before the data is loaded.

### **2.1.1 Business landscape**

As previously stated, the potential for data acquisition for analysis is taking place in real time more than ever before. Still, most of our decisions as business leaders or even as individuals are made based on information that is historic in nature or limited in scope. Although most decision makers believe that more current data leads to better decisions, stored data was the only source of data that was readily available for analysis. This situation drove expectations about how the analytic process and the tools that support analysis should work. Historically, businesses must wait until after the actual data was committed to storage before they can run any analysis. This analysis is then limited to the use of historical (and structured) data to predict the best actions for the future. By acquiring actual data in real time, businesses hope to analyze and take action in real time, but they need new products and tools that are designed to do so.

Figure 2-3 shows the results of a 2011 IBM study of 3,018 CIOs worldwide. The source of this data is the IBM Institute for Business Value Global CIO Study 2011. The survey found that there are four CIO mandates: Leverage, Expand, Transform, and Pioneer. Nearly one-quarter of the CIOs IBM polled support organizations that operate with a Transform mandate. Organizations with this mandate see IT primarily as a provider of industry-specific solutions to change the business.

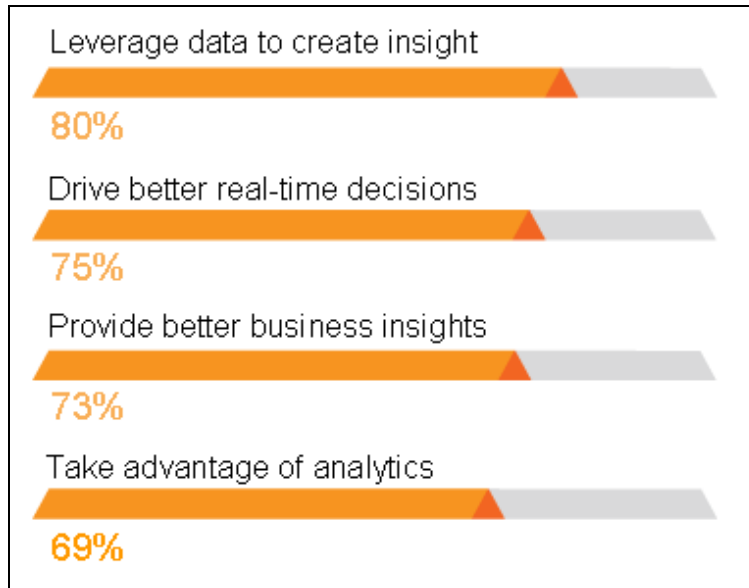


Figure 2-3 Enabling the intelligence enterprise

Beyond the delivery of basic IT services and business process improvements, Transform mandate CIOs are helping their public and private sector organizations fundamentally rethink the way that they understand and interact with customers and partners. One key to extending the enterprise's reach is the use of *big data*, that is, the vast volumes that are captured and analyzed, such as data from sensors, RFID tags, or real-time, web-based transactions that can be analyzed to drive better, real-time decisions. Big data, what to do with it, and how to use it, are top issues for Transform mandate CIOs.

Why is it that businesses continue to feel an urgency to seek an effective use of analytics? Because before you can determine an effective use strategy, the definition of what it takes to be effective changes. The rapid shift in the complexion of the data causes companies, governments, and even individuals to continually find themselves trying to determine how best to use the new data landscape.

An excess of good data might sound like a good problem to have, but it means that with enterprise data projected to double every 18 months, it is difficult to keep up. Also, with 80% of data growth that is driven by unstructured and non-traditional data sources, such as email, text, VoIP, and video, it is difficult to even know for which types of data to plan. With the fast pace of today's business environment, industry leaders are constantly called upon to make innovative decisions in real time to gain a competitive edge, or to simply remain competitive.

The time is near (if not already now) when these images and streaming data are far more prevalent than their historic structured counterparts. Businesses are challenged by not only the volume and velocity<sup>1</sup> of available data, but by the ability to interpret the broad range of sources.

<sup>1</sup> For more information, see this website:

<http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>

**Continually producing data:** A city the size of London could have tens of thousands of security cameras. Roads are often equipped with thousands of sensors for an area as limited as one bridge. Medical diagnostic equipment that creates digitized medical images makes up, or will soon make up, almost a third of all the data in the world. Most if not all of these devices are connected to the World Wide Web, where they continually produce data.

The list of sources of available data for analysis keeps growing. The increased reach of instrumentation brings with it the availability of a volume of data that businesses never dreamed might exist. We have smart appliances that can keep smart meters aware of energy usage in small increments for all appliances, which allows the utility company to make capacity adjustments at any level, even within a single home. Even something as basic as a tractor can capture and transmit soil condition, temperature, and water content and directly relay that information to a specific location to be used to monitor and manage water usage for irrigation systems, while simultaneously alerting the local dealership of the fact that the machine needs service. From the potential of smart grids, smart rail, smart sewers, and smart buildings, we can see significant shifts in the way decisions are made in the fabric of our lives.

Simultaneously, as individuals, we are becoming increasingly aware of the value of real-time information, whether to feel more connected in a society where being in the same physical locality is not always possible between family members and friends, or as a forum to have our opinions heard. The rise in popularity of the social networks indicates that there is also a considerable personal value to real-time information. In this forum, millions of people (and therefore customers, students, patients, and citizens) are voicing their opinion about everything from good and bad experiences they had with products or companies to their support for current issues and trends.

Businesses that can analyze what is said and align themselves with the popular desires are a powerful force.

In some situations, businesses tried to build applications to provide functionality beyond traditional analytics tools to use non-traditional or high-volume information. Unfortunately, many of these businesses find that their in-house built applications struggle to scale well enough to keep up with the rapidly growing data throughput rates. The reality is that even as the old obstacles to real-time analytics are removed, business leaders are still finding themselves limited in their ability to make real-time decisions. Unless they embrace a new analytical paradigm, the dynamic growth in data volumes result in increasing the time that is needed to process data, which potentially leads to missed opportunities and lowering (or losing) competitive advantage.

Businesses are keenly aware that knowing how to effectively use all known sources of data in a way that allows future sources to be incorporated smoothly can be the deciding factor that serves to fuel competitive, economic, and environmental advantages in real time. Those companies and individuals that find themselves positioned to use data in motion find themselves with a clear competitive edge. The time that it takes to commit the data to persistent storage can be a critical limitation in highly competitive marketplaces. More often we see that the effective key decisions must come from the insights that are available from traditional and non-traditional sources.

## 2.1.2 Information environment

You can already see that the changes in the business landscape are beginning to blur the lines between the information needs of business and the needs of the individual. Data that was once only generated, collected, analyzed, enriched, and stored in corporate data centers is now transmitted by some of the most innocuous devices. Data that was once only attainable by using the power that was provided by a business's information environment can now be accessed by devices as personal as intelligent cellular phones and personal computer tablets. With greater availability of data comes the potential for a new breed of analyst to enrich the information content. Thanks to the increases in instrumentation and interconnectivity, these new analysts and their new content can now be heard worldwide in mere seconds. The lines that historically delineated the information environment into its use by industries, business, government, or individuals are becoming all but invisible. Not only is the world becoming more instrumented and interconnected, but the entire planet is becoming more intelligent, or at least has that potential.

We are in the dawn of the IBM Smarter Planet® era in information evolution. As you can see in Figure 2-4, the Smarter Planet era builds from earlier styles of computing, and is driven by the rapid rise in the level of sophisticated computing technology that is delivered into hands of the real world.

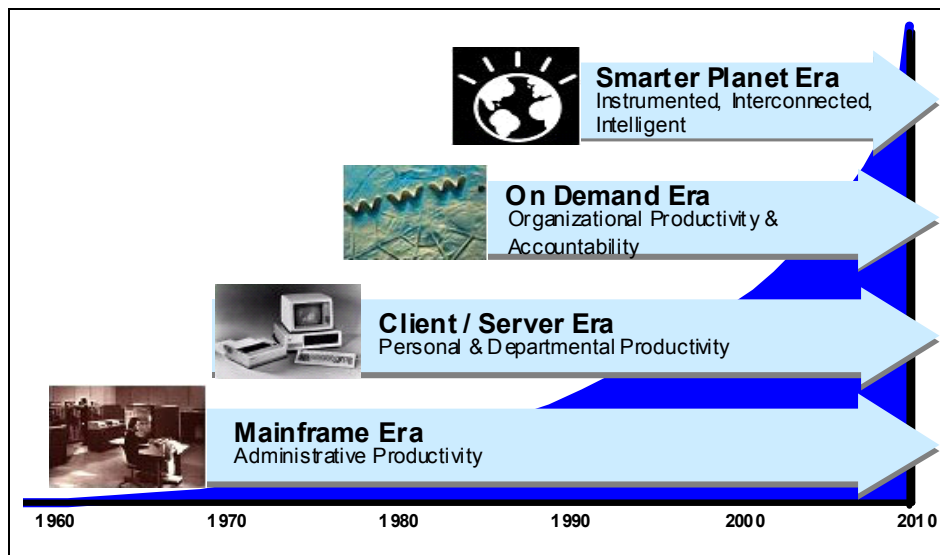


Figure 2-4 Eras of information technology evolution

The need for an automated information environment began in the mid-1960s with the Mainframe Era. A wonderful environment for automating the back-office functions, it is still widely in use today. Because all businesses feature back offices, the information environment of the Mainframe Era became prolific in all businesses and all industries. Not designed to do things such as plant floor manufacturing, departmental computing, or personal computing, this environment and the data it creates and stores are solely designed for and used by the specific enterprise. The purposes of the applications in this environment are to conduct the day-to-day business. The data that it generates is only truly used by the processes of those day-to-day operations, such as billing, inventory control, accounting, and sales analysis.

The awareness of the value of analysis came from the departmental units within the business. The information environment of the Mainframe Era was well-suited for creating and storing data, but not as well-suited to analyzing that information to make improvements in the way to do business. As departmental decision makers were pressed to improve their bottom lines, they found themselves needing an information environment that was flexible enough to support any question they needed to ask.

The Client/Server Era was the first information environment that was fashioned to support the business analyst. Unfortunately, as it rose from departmental demands, it was an information environment that is isolated from the data needs of the particular departmental unit. Still, it was quick and effective, and like its predecessor, it still exists in organizations today for targeted needs, such as campaign management.

But soon analysis could no longer be isolated to the departmental domain. The On-Demand Era was born out of the need to blend (and validate) the departmental analysis to provide insight and recommendations at the enterprise level. The On-Demand Era started integrating these islands of automation into an information environment that could serve the analytical needs of the entire enterprise. The rise of the Internet, with its open, standards-based, and widely available access, provided a framework to share departmental information within the business and the foundation for sharing information throughout the world at a pace we could barely imagine.

The growth of the environments to support information is relatively steady over the years and gradually growing. Still, the sources of data and the results of analysis were primarily the property of industry, government, and academic science. These enterprise data centers controlled the information because only they had the computing power to acquire and analyze it. As such, they reviewed and verified information before it was communicated to the rest of the world. But with computing power prevalent outside of the walls of industry, we must rethink what a computer really is and who might be analyzing the data.

**Generating and communicating data:** There were 4.6 billion mobile phone subscribers as of February 2010, and there will be 1.3 billion Radio Frequency Identification (RFID) tags produced globally in the next year. Sensors are embedded across entire ecosystems, such as supply chains, healthcare networks, electric grids, cities, and even natural systems such as rivers. These sensors and phones all can be considered computers that can generate and communicate data.

In the past, the information that was available on the Internet was generated and communicated only by those large data centers. Now it is estimated that not only will there likely be over 2 billion people that are connected to the Internet, but there will be as many as one trillion objects that are connected. This Internet of things, show in Figure 2-5, shows some examples of how computers are moving out of the data centers and into our everyday world.

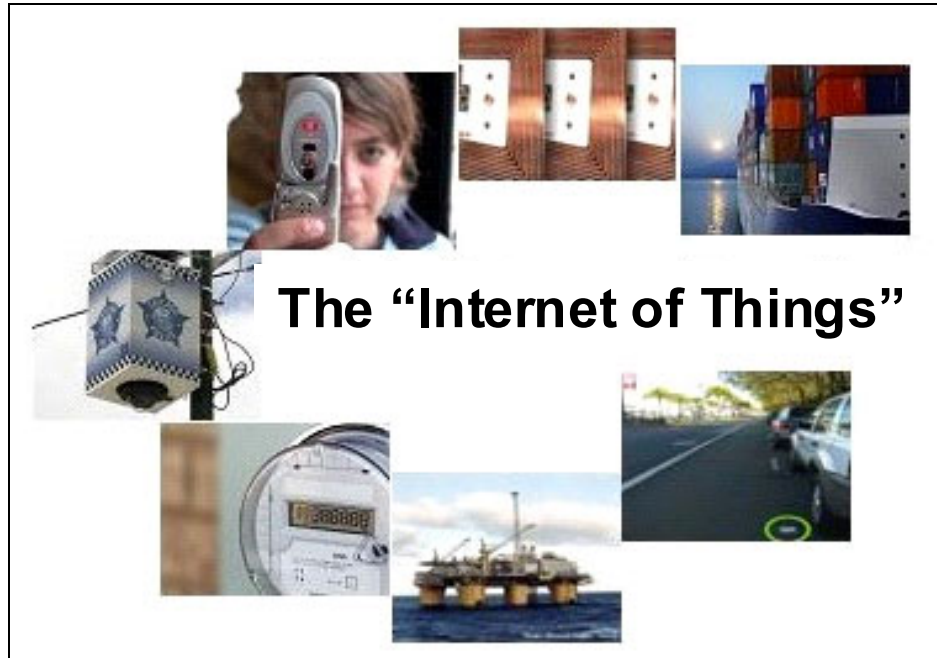


Figure 2-5 Computers are moving out of the data center into the world

The proliferation of these sensors and devices, which are embedded in everything from smart grids to rail cars and even chickens, help drive as much as a 10 times increase in the amount of data in the world. The ability to tap into that data for a new kind of intelligence can help drive vast improvements in our systems.

Welcome to the Smarter Planet Era. Driven by the surge in instrumentation, interconnectivity, and intelligence that is infused in all aspects of our lives, this new information environment holds the potential to infuse even complex analysis in the routine aspects of our lives. This new environment enables situations such as having traffic and security cameras in major cities help alert police and other first responders to the logistics of an incident far faster and more precisely than ever before. The information from sensors in your car could alert your garage-of-choice that your car needs maintenance so they can schedule an appointment for you and prompt you to confirm the appointment through a text message on your cell phone.

In-home monitors that transmit key health factors could include the capability that allows an elderly relative to continue living in their own home. Automated quality analysis that is installed on a factory line could identify defects in products before they leave the line to allow operators to take corrective measures.



Although access to data sooner might save lives, lower crime, save energy, and reduce the cost of doing business and therefore products, one of the by-products of the widespread instrumentation is the potential for feeling as though we live in a surveillance society. Although it is easy for people and businesses to see the value of access to the variety of data and smarter systems, it is also easy to see how they might be increasingly uncomfortable having so much information known about them. Individuals and businesses alike find themselves concerned that this essential environment might be only as secure and reliable as the average person's notebook or PDA.

**Big brother:** An article a few years ago reported that the London flat in which George Orwell wrote *1984* features 32 closed-circuit cameras within 200 yards, scanning every move. Those cameras were not put there specifically to spy on that flat; they were installed to scan traffic and provide security for local businesses. Still, they are there and as they are most likely connected to the Internet, the irony and the potential concern is easy to see.

Along with the sheer mechanics of analyzing the massive volumes of information in real time comes the challenge of how to provide security and privacy. The information environment of the Smarter Planet Era gets its reaction speed from accessing data in motion, data outside of the usual framework of data repository authorization and encryption features. The sheer volume of the available data in motion requires a high-performance environment for stream computing, but this environment also needs to be able to employ in-stream analysis to determine the credibility of sources and protect the information without adversely compromising the speed. When the information environment uses and allows widespread access to analyze and enrich data in motion, it also takes on some of the responsibility to protect the results of that analysis and enrichment of the data and determine the analysis is based on credible and trusted sources.

The potential of the information environment for this new Smarter Planet Era is great, but so are its challenges. Providing an environment that can deliver valuable and insightful information for real-time analytics without compromising on quality and security is the standard with which we measure the success of this era.

### 2.1.3 The evolution of analytics

As the business landscape and information environments are evolving, the approaches to effective analysis are at the dawn of a major evolution. We might think we are drowning in data, but in fact, we now can turn that data into useful information. Advanced software analytic tools and sophisticated mathematical models can help us identify patterns, correlations of events, and outliers. With these new tools, we can begin to anticipate, forecast, predict, and make changes in our systems with more clarity and confidence than ever before. We stand on the brink of the next generation of intelligence: analysis of insightful and relevant information in real time, which is the real value of a Smarter Planet. The evolution of the analytic process is shown in Figure 2-6.

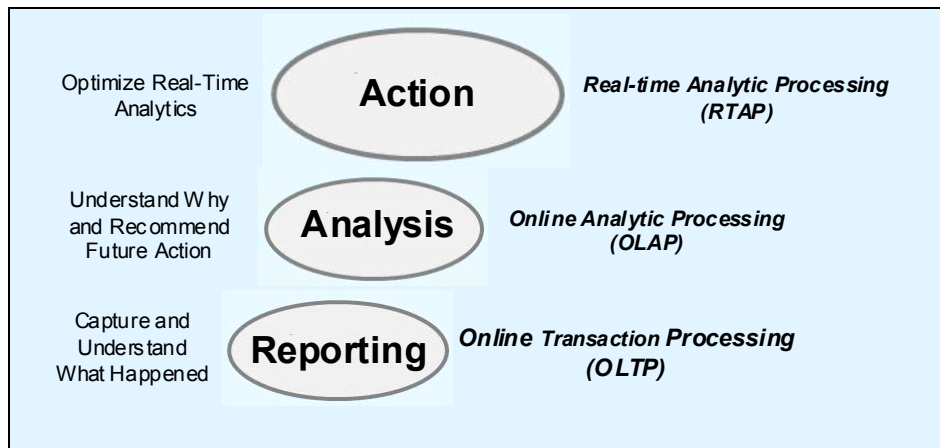


Figure 2-6 The next generation of business intelligence

Hierarchical databases were invented in the 1960s and still serve as the foundation for online transaction processing (OLTP) systems for all forms of business and government that drive trillions of transactions today. Consider a bank as an example. It is likely that even today in many banks that information is entered into an OLTP system, possibly by employees or by a web application that captures and stores that data in hierarchical databases. This information then appears in daily reports and graphical dashboards to demonstrate the current state of the business and to enable and support appropriate actions. Analytical processing here is limited to capturing and understanding what happened.

Relational databases brought with them the concept of data warehousing, which extended the use of databases from OLTP to online analytic processing (OLAP). By using our example of the bank, the transactions that are captured by the OLTP system were stored over time and made available to the various business analysts in the organization. With OLAP, the analysts can now use the stored data to determine trends in loan defaults, overdrawn accounts, income growth, and so on. By combining and enriching the data with the results of their analysis, they could do even more complex analysis to forecast future economic trends or make recommendations on new investment areas. Additionally, they can mine the data and look for patterns to help them be more proactive in predicting potential future problems in areas such as foreclosures. The business can then analyze the recommendations to decide whether they must take action. The core value of OLAP is focused on understanding why things happened to make more informed recommendations.

A key component of OLTP and OLAP is that the data is stored. Now, some new applications require faster analytics than is possible when you have to wait until the data is retrieved from storage. To meet the needs of these new dynamic applications, you must take advantage of the increase in the availability of data before storage, otherwise known as streaming data. This need is driving the next evolution in analytic processing called real-time analytic processing (RTAP). RTAP focuses on taking the proven analytics that are established in OLAP to the next level. Data in motion and unstructured data might be able to provide actual data where OLAP had to settle for assumptions and hunches. The speed of RTAP allows for the potential of action in place of making recommendations.

So, what type of analysis makes sense to do in real time? Key types of RTAP include, but are not limited to, the following analyses:

► Alerting

- The RTAP application notifies the user or users that the analysis identified that a situation (based on a set of rules or process models) occurred and then optionally provides recommendations and options for the appropriate actions.
- Alerts are useful in situations where the application should not be automatically modifying the process or automatically taking action. They are also effective in the following situations where the action to be taken is outside the scope of influence of the RTAP application:
  - A market surveillance application that is monitoring a local exchange for suspect activity notifies someone when such a situation occurs.
  - A patient monitoring application alerts a nurse to take a particular action, such as administering more medicine.

► Feedback

- The RTAP application identifies that a situation (based on a set of rules or process models) occurred and makes appropriate modifications to the processes to prevent further problems or to correct the problems that already occurred.
- Feedback analysis is useful, as an example, in manufacturing scenarios where the application determined that defective items were produced and takes action to modify components to prevent further defects.

An example is a manufacturer of plastic containers that might run an application that uses the data from sensors of the production line to check the quality of the items throughout the manufacturing cycle. If defective items are sensed, the application generates instructions to the blending devices; for example, to adjust the ingredients to prevent further defects from occurring.

► Detecting failures

- The RTAP application is designed to notice when a data source does not respond or generate data within a prescribed period.
- Failure detection is useful in determining system failure in remote locations or problems in communication networks, including the following examples:
  - An administrator for a critical communications network deploys an application to continuously test that the network is delivering an adequate response time. When the application determines that the speed drops below a certain level, or is not responding at all, it alerts the administrator, wherever they happen to be.
  - An in-home health monitoring application determines that the motion sensors for a particular subscriber were activated today and sends an alert to a caregiver to check on the patient to determine why they are not moving around.

You can see that if you consider the improvements that were made in chip capacity, network protocols, and input/output caching with the advances in instrumentation and interconnectivity and the potential of stream computing, we stand poised and ready to present appropriate information to decision makers in time for proactive action to be taken. The days of solely basing our decisions on static data are yielding to the use of streaming data or data in motion.

## 2.2 IBM InfoSphere Streams

In April of 2009, IBM made available a revolutionary product named IBM InfoSphere Streams (Streams). Streams is a product that is designed specifically to help clients continuously analyze massive volumes of streaming data at extreme speeds to improve business insight and decision making. Based on ground-breaking work from an IBM Research team that was working with the US government, Streams is one of the first products that was designed specifically for the new business, informational, and analytical needs of the Smarter Planet Era.

As seen in Figure 2-7, Streams is installed in over 150 sites across six continents. Hundreds of applications were developed, which helped IBM understand more about client requirements for this new type of analysis.

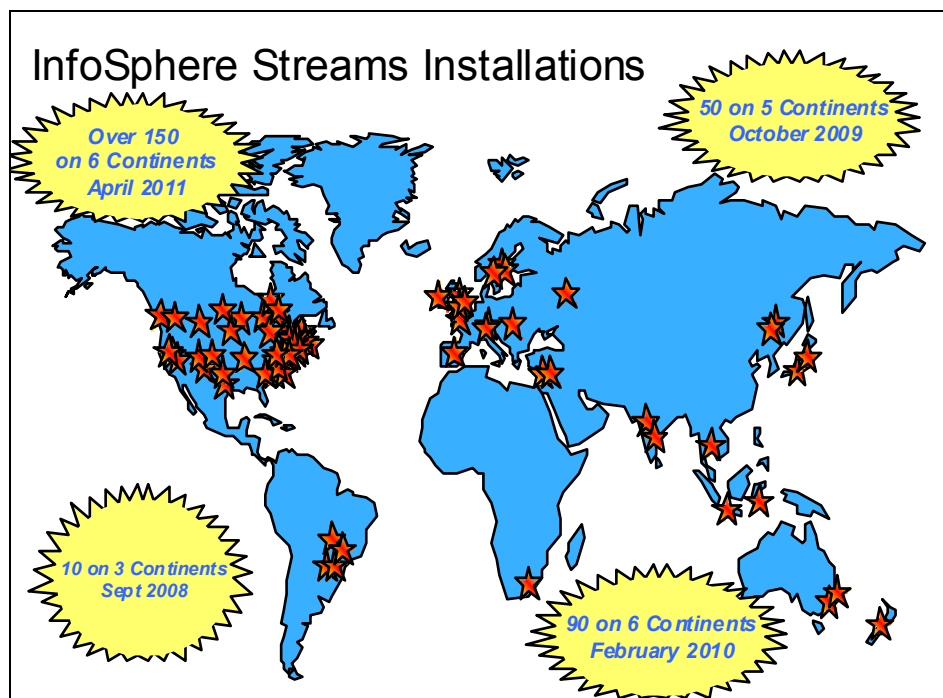


Figure 2-7 Streams installations as of February 2010

Streams is based on nearly a decade of effort by the IBM Research team to extend computing technology to handle advanced analysis of high volumes of data quickly. How important is their research? Consider how it would help crime investigation to analyze the output of any video cameras in the area that surrounds the scene of a crime to identify specific faces of any persons of interest in the crowd and relay that information to the unit that is responding. Similarly, what a competitive edge it could provide by analyzing 6 million stock market messages per second and execute trades with an average trade latency of only 25 microseconds (far faster than a hummingbird flaps its wings). Think about how much time, money, and resources could be saved by analyzing test results from chip-manufacturing wafer testers in real time to determine whether there are defective chips before they leave the line.

**System S:** While at IBM, Dr. Ted Codd invented the relational database. In the defining IBM Research project, it was referred to as System R, which stood for Relational. The relational database is the foundation for data warehousing that launched the highly successful Client/Server and On-Demand informational eras. One of the cornerstones of that success was the capability of OLAP products that are still used in critical business processes today.

When the IBM Research division again set its sights on developing something to address the next evolution of analysis (RTAP) for the Smarter Planet evolution, they set their sights on developing a platform with the same level of world-changing success, and decided to call their effort System S, which stood for Streams. Like System R, System S was founded on the promise of a revolutionary change to the analytic paradigm. The research of the Exploratory Stream Processing Systems team at T.J. Watson Research Center, which was set on advanced topics in highly scalable stream-processing applications for the System S project, is the heart and soul of Streams.

Critical intelligence, informed actions, and operational efficiencies that are all available in real time is the promise of Streams. InfoSphere Streams helps us realize the promise of a Smarter Planet.

## 2.2.1 Overview of Streams

For the purposes of this overview, it is not necessary to understand the specifics of Streams, as those specifics are described in more detail in subsequent chapters. The purpose here is only to demonstrate how Streams was designed to focus on the ability to deliver RTAP of exceedingly large volumes of data by using a flexible platform that is positioned to grow with the increasing needs of this dynamic market.

As the amount of data available to enterprises and other organizations dramatically increases, more companies are looking to turn this data into actionable information and intelligence in real time. Addressing these requirements requires applications that can analyze potentially enormous volumes and varieties of continuous data streams to provide decision makers with critical information almost instantaneously. Streams provides a development platform and runtime environment where you can develop applications that ingest, filter, analyze, and correlate potentially massive volumes of continuous data streams that are based on defined, proven, and analytical rules that alert you to take appropriate action, all within an appropriate time frame for your organization.

The Streams product goes further by allowing the applications to be modified dynamically. Although there are other systems that embrace the stream computing paradigm, Streams takes a fundamentally different approach to how it performs continuous processing and therefore differentiates itself from the rest with its distributed runtime platform, programming model, and tools for developing continuously processing applications. The data streams that are consumable by Streams can originate from sensors, cameras, news feeds, stock tickers, or various other sources, including traditional databases. The streams of input sources are defined and can be numeric, text, or non-relational types of information, such as video, audio, sonar, or radar inputs. Analytic operators are specified to perform their actions on the streams.

A simple view of the components of Streams is shown in Figure 2-8.

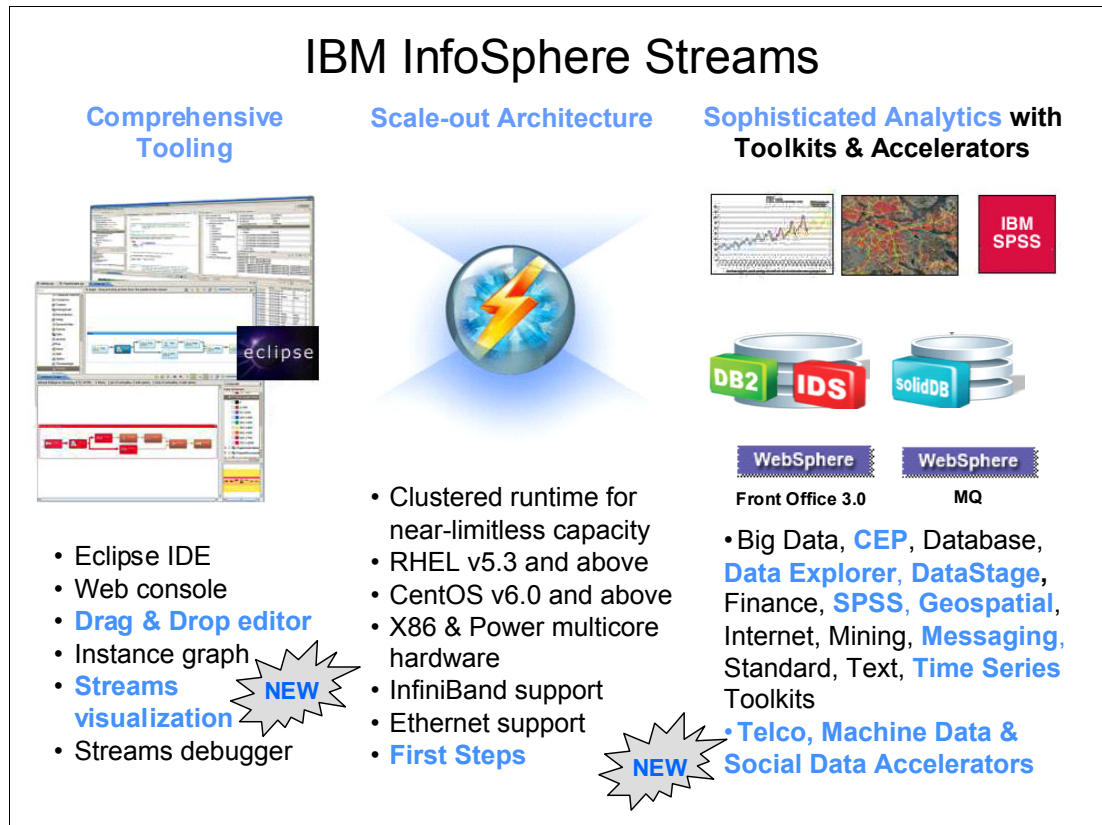


Figure 2-8 The components of InfoSphere Streams

Applications are developed in the InfoSphere Streams Studio by using IBM Streams Processing Language (previously called Streams Processing Application Declarative Engine), which is a declarative language that is customized for stream computing. After the applications are developed, they are deployed to Streams Runtime environment. By using Streams Live Graph, you can monitor the performance of the runtime cluster from the perspective of individual machines and the communications between them.

Virtually any device, sensor, or application system can be defined by using the language. But, there are predefined source and output adapters that can further simplify application development. As examples, IBM delivers the following adapters:

- ▶ TCP/IP, UDP/IP, and files
- ▶ IBM WebSphere Front Office, which delivers stock feeds from major exchanges worldwide
- ▶ IBM solidDB® includes an in-memory, persistent database that uses the Solid Accelerator API
- ▶ Relational databases, which are supported by using industry standard ODBC

Applications, such as the one shown in Figure 2-9, almost always feature multiple steps.

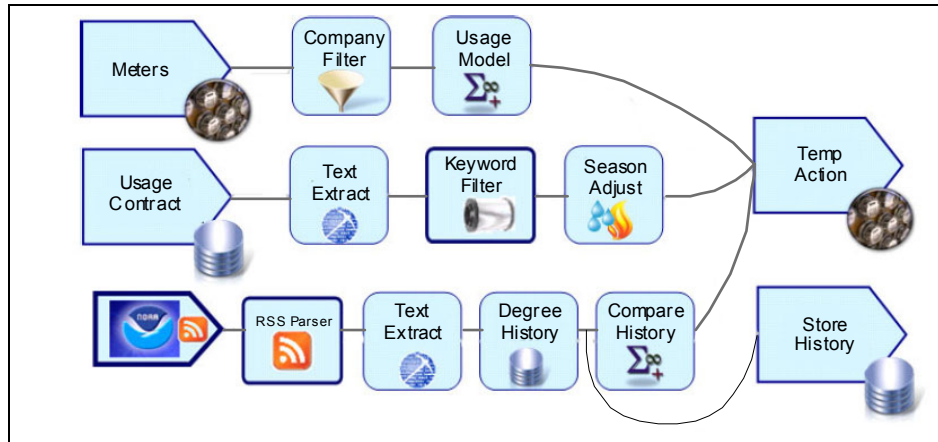


Figure 2-9 Application graph of a Streams application

For example, some utilities began paying customers who sign up for a particular usage plan to have their air conditioning units turned off for a short time, allowing the temperature to be changed. An application to implement this plan collects data from meters and might apply a filter to monitor only for those customers who selected this service. Then, a usage model must be applied that was selected for that company. Up-to-date usage contracts then need to be applied by retrieving them, extracting text, filtering on key words, and possibly applying a seasonal adjustment. Current weather information can be collected and parsed from the US National Oceanic & Atmospheric Administration (NOAA), which has weather stations across the United States. After the correct location is parsed for, text can be extracted and temperature history can be read from a database and compared to historical information. Optionally, the latest temperature history could be stored in a warehouse for future use. Finally, the three streams (meter information, usage contract, and current weather comparison to historical weather) can be used to take actions.

Streams delivers an integrated development environment (IDE) that is based on the Open Source Eclipse project, as shown in Figure 2-10. Developers can specify operators to be used in processing the stream, such as filter, aggregate, merge, transform, or more complex mathematical functions, such as Fast Fourier Transforms. The developer also specifies the data input streams to be used (source operators) and data output streams (sink operators). Some of the source and sink adapters are included within the Streams product and the environment also allows the developer to define custom adapters, including custom analytics or operators that are written in C++ or Java. Existing analytics can also be called from Streams applications. For more information about these functions, see Chapter 5, “InfoSphere Streams deployment” on page 63, Chapter 6, “Application development with Streams Studio” on page 105, and Chapter 7, “Streams integration considerations” on page 161.

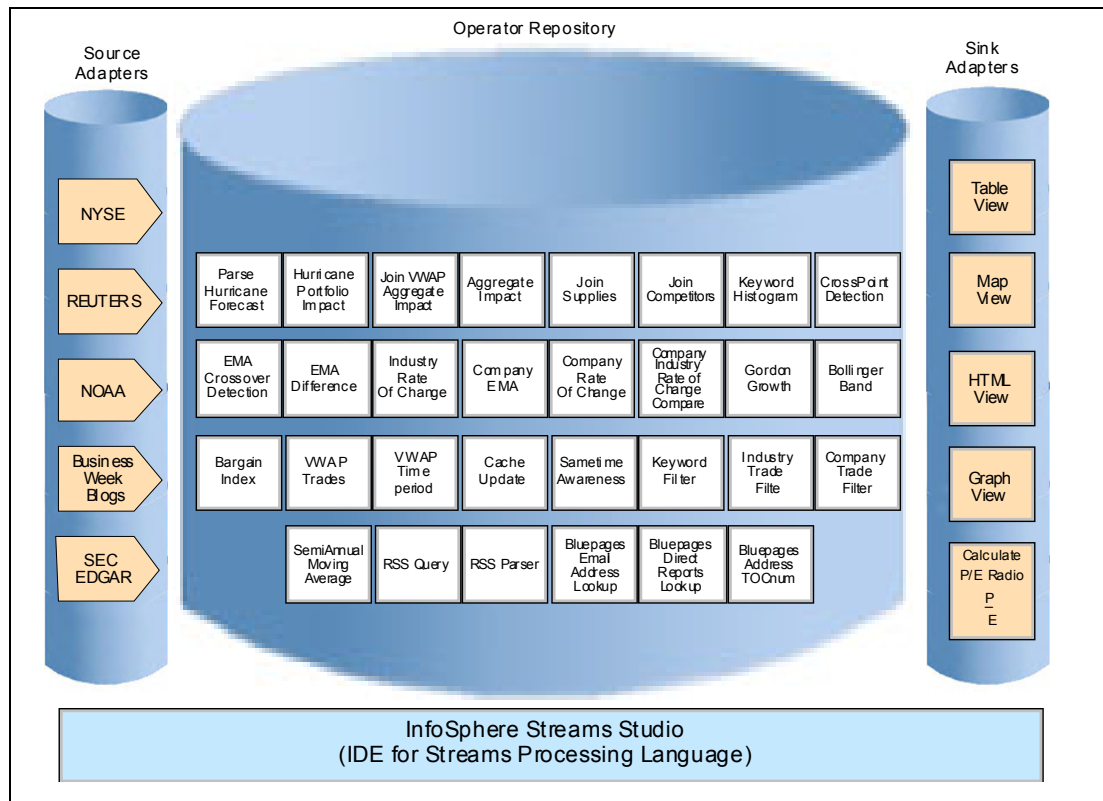


Figure 2-10 The Streams integrated development environment

InfoSphere Streams Studio not only helps you develop applications, but includes the powerful Streams Debugger tool. As with many IDEs, you can set breakpoints and stops in your application to facilitate testing and debugging. In addition, the Streams Debugger works with the Streams Runtime and the Streams Live Graph tool so that you can find a breakpoint even after the application is deployed. You can inspect or inject new data to ensure that the application is doing what you want it to do.



An optimizing compiler translates the application to executable code. The compiler makes many decisions to optimize application performance, such as buffer sizes that are required, which functions should be run on a single machine, and when new execution threads can be initiated. When deployed, the Streams Runtime works with the optimizing compiler to allocate the application across the runtime configuration, which is similar to the example that is shown in Figure 2-11.

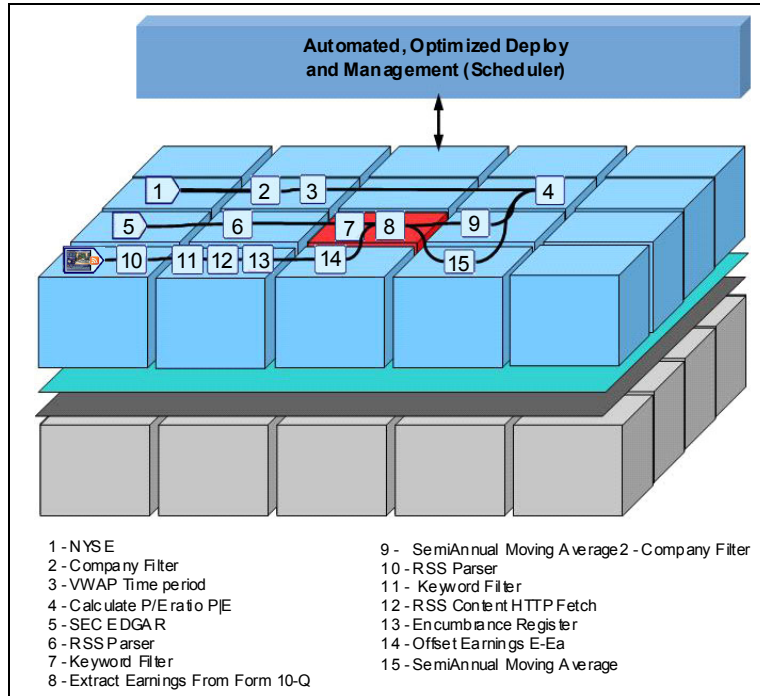


Figure 2-11 A typical runtime deployment of a Streams application

The Streams Runtime establishes communications for the streams of data as they are processed by the application. This task is accomplished by setting up connections to route data into the Streams Runtime, between the various operators, and out of the Streams Runtime cluster. Because the Streams Runtime is handled in memory across systems within a high speed infrastructure, the overall application experiences low latency.

The Streams Runtime monitors the environment's performance and if it detects poor performance or hardware failure, the administrator can take corrective actions. For example, the administrator can move an analytic function to another processor in the cluster. By using the Streams Runtime, you can add or remove applications without bringing down the cluster. You can also add or remove computer nodes from the cluster while it is running.

## 2.2.2 Why use Streams

The architecture of Streams represents a significant change in the organization and capabilities of a computing platform. Typically, stream computing platforms are designed with the lofty goal of achieving success of the following objectives:

- ▶ Respond in real time to events and changing requirements.
- ▶ Continuously analyze data at volumes and rates that are orders of magnitude greater than existing systems.
- ▶ Adapt rapidly to changing data formats and types.
- ▶ Manage high availability, heterogeneity, and distribution for the new stream paradigm.
- ▶ Provide security and information confidentiality for shared information.

Although there are academic and commercial initiatives that are focused on delivering on technical challenges in isolation, Streams attempts to simultaneously address all of these challenges. Streams is on a mission to break through a number of fundamental barriers to enable the creation of solutions that are designed to deliver on all of these critical objectives.

To do so, the Streams development environment provides a graphical representation of the composition of new applications to make them easier to understand. New applications or new rules for analysis can be created dynamically, mapped to various hardware configurations, and then deployed from the development environment to provide the wanted flexibility and agility of administration that is needed within this new paradigm. As analytical needs or rules change or the priorities and value of data sources shift, the affected portions of the applications can be modified and redeployed incrementally with minimal impact to the rest of the application. As hardware resource availability grows within the data center, Streams is designed to scale from a single node to one or more high-performance clusters that include an unlimited number of processing nodes within a cluster. The runtime component continually monitors and adapts to the state and utilization levels of its computing resources and the data velocity.

There are other products that are engineered to focus on some of the objectives that are described here, many of which center around the analysis of events. You should think of an event as something that happens, such as a temperature reading, a purchase, or an insurance claim. Events might be simple events, such as a purchase transaction, or complex events, such as a combination of purchases with a specific time span.

A data stream is different. Streaming data is more of a continuous set of never-ending readings; for example, the data from an EKG, the output of a video camera, or the sound that is captured by a microphone. Data streams can be viewed as a continuous stream of events that are happening quickly and as such, there are often correlations and comparisons between stream processing and complex event processing.

IBM is aware of the potential of the use of business event processing (processing of simple and complex events) for quite some time. To get a better idea of exactly how to use this dynamic, IBM classified the following major opportunities for business event analysis and determined the likely corresponding entry points to consider business event processing systems such as Streams:

- ▶ Business Activity Monitoring: Communicating key performance indicators (KPIs) and alerts to business stakeholders
- ▶ Information-Derived Events: Identifying correlating events across data stores and data streams
- ▶ Business Logic Derived Events: Identifying correlating events across transactions and business processes
- ▶ Active Diagnostics: Identifying potential issues and taking action
- ▶ Predictive Processing: Identifying future issues that are based upon business events in context and over time
- ▶ Business Service Management: Ensuring IT health so service level agreements are met
- ▶ High Volume Stream Analytics: Analysis on high volume data streams in real time

Complex Event Processing (CEP) is an evolution of business event processing. It is primarily a concept that deals with the task of processing multiple business events with the goal of identifying the meaningful characteristics within the entire scenario. In addition, Business Event Processing (BEP) is defined by IBM and handles simple and complex events within a business context.

The goal is always to take advantage of an appropriate action plan in real time. Because of a similar goal within CEP, CEP tools often include the same or similar marketing messages as Streams. However, the proof points behind those messages are often different. At the core, these goals of ultra-low latency and real-time event stream processing in real time are true for both. While CEP tools usually achieve response times of under a millisecond, Streams achieved response times below 100 microseconds. Both types of tools are able to use parallel processing platforms and techniques, such as partitioning and pipelining, to achieve performance. However, the extent of that achievement is far from the same when a typical CEP tool is compared to Streams.

Streams features some similarities to other CEP tools on the market, but Streams is built to support higher data volumes and a broader spectrum of the data sources that are used for input. Streams can successfully process millions of messages per second while other CEP tools often can process only hundreds of thousands of messages per second. Although CEP tools are able to handle discrete events, Streams can handle discrete events and real-time data streams, such as video and audio. As we previously described, Streams also provides an infrastructure to support the need for scalability and dynamic adaptability, by using scheduling, load balancing, and high availability techniques that certainly are key as this information environment continues to evolve.

Like Streams, CEP tools and other products use techniques such as the detection of complex patterns within many events, event correlation and abstraction, event hierarchies, and relationships between events to identify events of events. These complex events allow for analysis across all the layers of information to determine their impact. CEP tools often employ if/then/else-type rules-based analysis. With Streams, you are able to incorporate powerful analytics, such as signals processing, regressions, and clustering algorithms.

A few characteristics of Streams and CEP are listed in Table 2-1 to help clarify and differentiate the products.

Table 2-1 Characteristics comparison of Streams and CEP tools

Complex Event Processing	InfoSphere Streams
Analysis on discrete business events.	Analytics on continuous data streams.
Rules-based processing (that uses if/then/else) with correlation across event types.	Supports simple to complex analytics and can scale for computational intensity.
Only structured data types are supported.	Supports an entire range of relational and non-relational data types.
Modest data rates.	Extreme data rates (often an order of magnitude faster).

As you can see, Streams is fully aligned to the challenges of the Smarter Planet Era. The optimized runtime compiler is designed to manage the high volume of data that makes up the current challenges and makes up the future business landscape by providing the low latency that is required by real-time analytical processing. The development environment and toolkits are focused on the use of a client's existing analytics. In addition, the environment easily allows for the creation of new and complex analytics to find credibility and value for multiple traditional and even non-traditional information sources that are becoming the key elements in our ever-broadening information environment.

### 2.2.3 Examples of Streams implementations

Even throughout its research and development phase, Streams was demonstrating success by delivering cutting edge commercial and scientific applications. Many of these applications showed clients a way to capture the formerly unattainable competitive edge. In addition, many of these early applications are playing a significant role in the evolution of the Smarter Planet.

From the beginning, the intent for the Streams infrastructure was to provide a solid foundation for this broad spectrum of practical applications because they were designed specifically to address the fundamental challenges of analysis on streaming data. By featuring the ability to continually deliver exceptional performance (for all types of analyses across massive amounts of data) while remaining flexible enough to offer interoperability (with the existing application infrastructures), Streams is positioned for any solution that benefits from analyzing streaming data in real time. The uses are as varied as the imagination of the analysts that are involved in almost every industry and initiative.

Some of the early adopters of Streams developed solutions with life-changing potential. To demonstrate some of the types of solutions you can create with the Streams platform, the following sections provide brief examples.

Over the past several years, hundreds of applications were developed by using InfoSphere Streams. The following sections provide a summary of a few applications and highlight the types of usage that are supported by InfoSphere Streams.

## Example 1: Telecommunications: Call Detail Record mediation and analytics

The challenge of closing certain technology and business gaps is especially apparent for cellular service providers in Asia. Chips that are embedded in cell phones enable email, text messaging, pictures, videos, and information sharing by using social sites such as Facebook. For each phone call, email, web browse, or text message, cellular phone switches emit call detail records (CDRs). To ensure that no data is lost, the switches emit two CDRs for each transaction, which later must be deduplicated for the billing support systems. A rising volume of data caused mediation of CDRs to become ever more difficult to perform in a timely manner. Phone number portability enabled subscribers to move to a competitor at any time. Some sophisticated users even switch between multiple providers at different times within a specific day to take advantage of certain promotions. Providers not only needed to reduce the window of processing CDRs to near real time, but also perform real-time analytics in parallel to predict which customers might leave for a competitor, which is known in the industry as *churn*. With this real-time insight into customer behavior, providers could take action to retain a higher percentage of customers.

InfoSphere Streams, with its agile programming model, enabled customers to handle their huge volume of CDRs with low latency, while churn analysis is provided on the data at the same time. At one company, a peak rate of 208,000 CDRs per second is processed with an end-to-end processing latency of under 1 second. Each CDR is checked against billions of existing CDRs with duplicates eliminated in real time, effectively cutting in half the amount of data that is stored in their databases.

This example illustrates a key Streams use case: simultaneous processing, filtering, and analysis in real time. High Availability, automated fault tolerance and recovery, along with real-time dashboard summaries, also are currently in use and improving IT operations. Real-time analysis of CDRs is leading to improved business operations by performing tasks such as churn prediction and campaign management to improve the retention rate of their current customers, as shown in Figure 2-12.

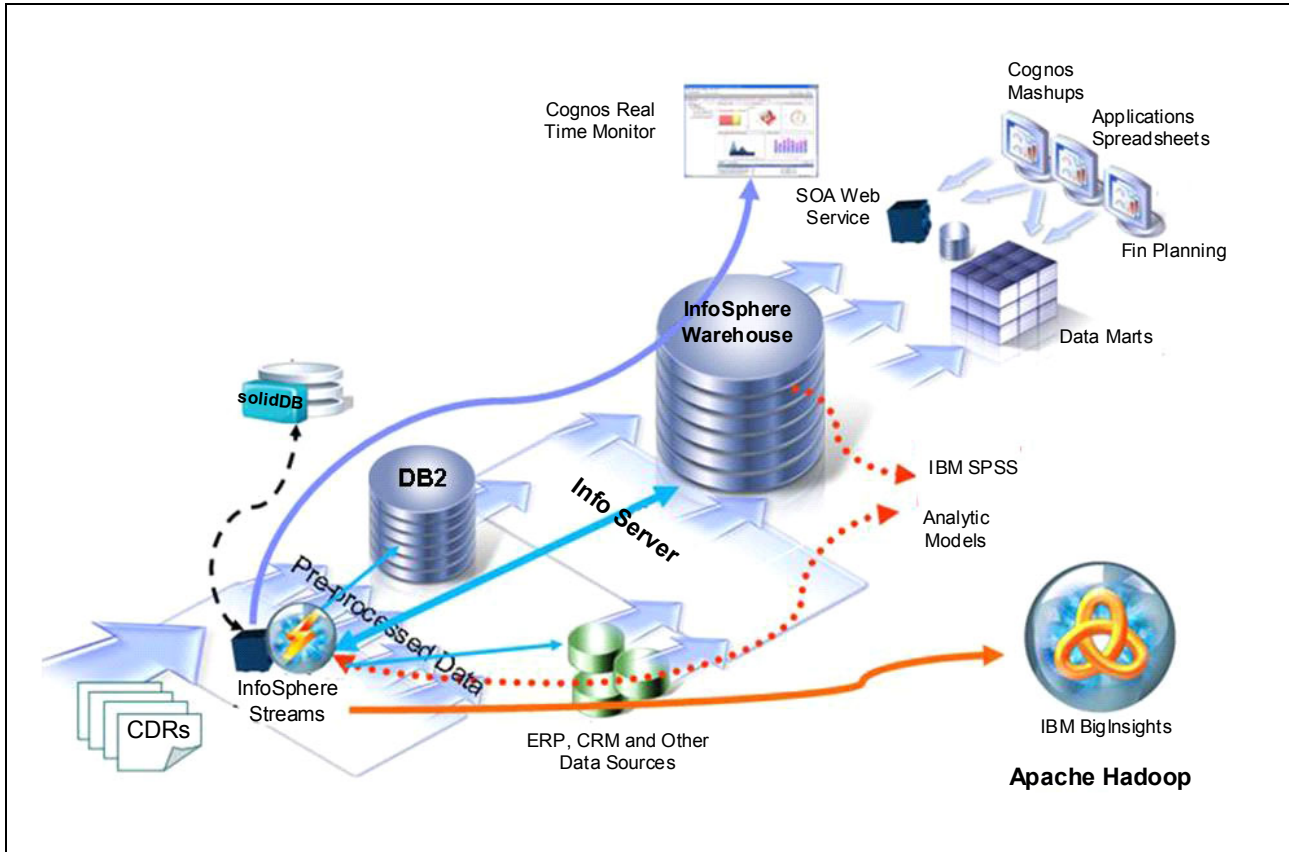


Figure 2-12 InfoSphere Streams Call Detail Record processing architecture

## Example 2: Government: Maintaining cybersecurity in a hostile environment

A key strength of Streams is the ability for Streams to perform analytics on data-intensive streams to quickly and accurately identify the typically small number of items that merit deeper investigation. One example of this use case can be found in the domain of cybersecurity. A *botnet* is a network of software agents or robots that run autonomously and automatically. Botnets respond to Command and Control (C&C) machines, where an underground economy sprung up to provide per pay access to the botnets for criminal activity. These botnets are evolving rapidly to make it difficult to detect the bots because of fast fluxing networks and encryption. The Shadowserver Foundation tracks known bots, and they estimate that there are thousands of C&C machines, and tens of thousands of bots in action today.

InfoSphere Streams was used to analyze IP traffic at a data rate of over 100 megabytes per second of IP traffic and over 10 million domain name server (DNS) queries per hour to generate fast fluxing botnet alerts. InfoSphere Streams uses machine learning models and models that were created by using historic data in IBM InfoSphere Warehouse for botnet alerts. SPSS Modeler is used to create these historic models. InfoSphere Streams also monitors for model drift. When the attack patterns change, Streams issue requests to have models updated against the historic data to ensure that all parts of the solution are using up-to-date detection models. This use case shows the value of continuously updating the data mining models for historic and real-time analysis of the data.

### **Example 3: Financial Services: Finding arbitrage opportunities faster than the competition**

Many segments of the financial services industry rely on rapidly analyzing large volumes of data to make near-real time business and trading decisions. Today, these organizations routinely use market data at rates that exceed one million messages per second, twice the peak rates they experienced only a year ago. This dramatic growth in market data is expected to continue for the foreseeable future, outpacing the capabilities of many current technologies. Industry leaders are extending and refining their strategies by including other types of data in their automated analysis; sources range from advanced weather prediction models to broadcast news. IBM developed an InfoSphere Streams that is based trading prototype that runs on a single 16 core x86 computer that processes OPRA data feeds at up to 5.7 million options messages per second, with latencies of under 30 microseconds. Recompiling the application allows it to scale even further when it is deployed across a cluster of computers.

### **Example 4: Health monitoring: Predicting the onset of illness earlier**

Stream computing can be used to better perform medical analysis while the workload on nurses and doctors is reduced. Privacy-protected streams of medical-device data can be analyzed to detect early signs of disease, correlations among multiple patients, and efficacy of treatments. There is a strong emphasis on data provenance within this domain, which is the tracking of how data are derived as they flow through the system. The “First of a Kind” collaboration between IBM and the University of Ontario Institute of Technology uses InfoSphere Streams to monitor premature babies in a neonatal unit. Data was collected for more than 18 months from a hospital in Toronto, Canada. Remote telemetry from a US hospital was operational for a year by using the same analytic routines. And earlier this year, more hospitals in China and Australia began implementing this solution. For more information, see *University of Ontario Institute of Technology: Leveraging key data to provide proactive patient care* at this website:

<http://public.dhe.ibm.com/common/ssi/ecm/en/odc03157usen/ODC03157USEN.PDF>

### Example 5: Transportation: Getting from here to there faster

IBM is working on an application in the IBM Smarter Cities® Technology Centre in Dublin, Ireland where InfoSphere Streams receives GPS data once per minute from buses. A real-time display shows all buses as they move through the city. The pilot is working to extend this solution by providing real-time predictions that are related to arrival times for each bus at each bus stop, which enables bus riders to better plan when to arrive at the bus stops, thus reducing waiting times. In the future, a personal travel planner could allow riders to receive recommendations that are based on real-time traffic monitoring. Such a system is shown in Figure 2-13.



Figure 2-13 Smarter transportation

In addition, other use cases of InfoSphere Streams are fast emerging. These domains include environmental monitoring and control (wildfire detection and water flow monitoring), the energy and utilities industry (synchrophasor monitoring of smart grids and prediction of wind power generation), radio astronomy, x-ray diffraction that uses synchrotrons, and fraud prevention.





## InfoSphere Streams architecture

This chapter describes the InfoSphere Streams Architecture and how it enables the development and execution of applications that process massive volumes of moving data due to its ability to be scalable and parallelized. We present the main components and how they are interconnected to produce high-performance results. It is a depth perspective of InfoSphere Streams architecture.

InfoSphere Streams is designed to address the following objectives:

- ▶ Parallel and high performance stream processing software platform that can scale over a range of hardware environments.
- ▶ Automated deployment of stream processing applications on configured hardware.
- ▶ Incremental deployment without restarting to extend stream processing applications.
- ▶ Secure and auditable execution environment.

This chapter contains the following topics:

- ▶ In “Streams concepts and terms” on page 42, we compare stream processing with a conventional database system and visualize this new implementation perspective and its challenges.
- ▶ In “InfoSphere Streams runtime system” on page 45, we describe the structure of an application and how it structured the architecture of components and services to work with huge volumes of data in motion.
- ▶ In “Performance requirements” on page 51, we describe the main points to observe when you are designing your big data cluster topology. We also describe how IBM System x® features can help you improve availability and performance at minimum cost.

## 3.1 Streams concepts and terms

The InfoSphere Streams architecture represents a significant change in computing system organization and capability. It provides a runtime platform, development model, and tools for applications that are required to process continuous data streams, as we see in 2.2.2, “Why use Streams” on page 34. The need for such applications arises in environments where information from one to many data streams can be used to alert users or other systems, or to populate knowledge bases for later queries, for example.

### 3.1.1 Working with continuous data flow

Standard database servers generally offer static data models and data and support dynamic queries. So, the data is stored somewhere (most often on a computer hard disk), and is persistent. Queries can arrive at any time and answer any question that is contained inside the historical data and data model. However, in an IBM InfoSphere Streams environment, you do not have to store data before it is analyzed. In fact, with Streams it is expected that the observed data volume can be so large that you cannot afford to make it persistent. Streams supports the ability to build applications that can support these huge volumes of data that include the critical requirement to be analyzed and acted upon with low latency in close to or in real time. For this situation, static models are not sufficient; you need to analyze the data while it is in motion. As an example, Streams can process 12 million messages per second with results returned in just 120 microseconds. To handle these types of speeds, a new processing paradigm is required. That paradigm is delivered by InfoSphere Streams.

Standard database servers generally feature a static data model and data, and dynamic (albeit often long running) queries. IBM InfoSphere Streams supports a widely dynamic data model and data. The Streams version of a query runs continuously without change. These two approaches to data analysis are shown in Figure 3-1.

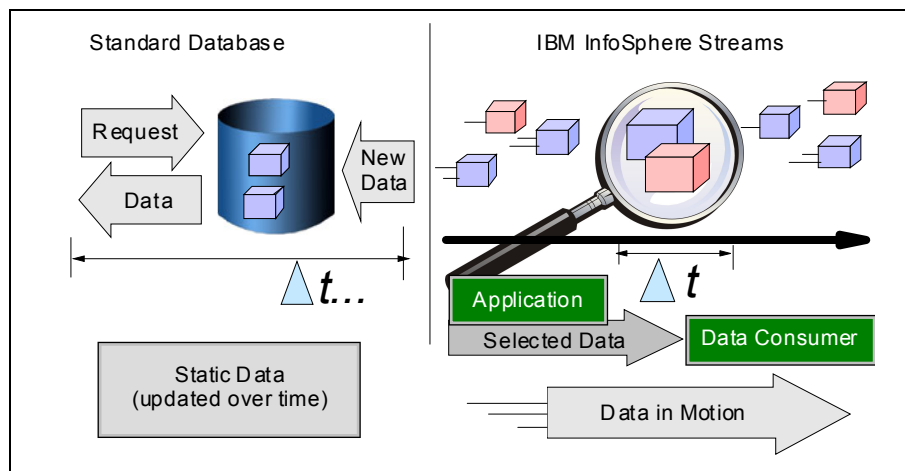


Figure 3-1 A standard relational database that is compared to IBM InfoSphere Streams

The traditional approach, which is shown on the left in Figure 3-1, involves storing data in a standard database, updating that static data over time, and periodically running queries against that data. By contrast, in the Streams approach shown on the right in Figure 3-1, the data of interest within the stream is defined and then processed as it flows by the application. The data is processed while it is in motion. Because of its volume or velocity, it is typically not stored. For more information, see this website:

<http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>

Queries in IBM InfoSphere Streams are defined by the Streams application and run continuously (or at least until they are canceled). In a database environment, a hard disk often holds the data and the requester is the consumer of that data. In a Streams environment, data that meets the defined criteria is selected as it flows past the application, which then sends it to the appropriate consumer of that data.

Because Streams applications are always running and continually sending selected data that meets the defined criteria to the consumers, they are well-suited for answering *always-on* or *continuous* types of questions. These questions include: what is the rolling average, how many parts were built so far today, how does production today compare with yesterday, and other continuous, subsecond response time statistics. Because of the nature of the questions that Streams is designed to answer, it provides join-like capabilities that are well beyond those capabilities that are found in standard SQL.

### 3.1.2 Component overview

IBM InfoSphere Streams offers a combination of command-line tools and graphical tools (web-based and fat client), and tools to perform administrative, monitoring, and application development functions. InfoSphere Streams provides a runtime platform, developing and debug toolkit, and other administrative resources. InfoSphere Streams architecture represents a significant change in computing system organization and capability.

#### Runtime environment

The runtime environment system is composed of a collection of components and services, such as platform services and a scheduler to deploy and monitor Streams applications across a single host or set of integrated hosts. It is well-integrated with the operating system for high performance. This environment is our execution environment where our InfoSphere Streams Applications run. The underlying runtime environment automatically determines how to best organize processing resources to meet the needs of running jobs and newly submitted jobs. The system continually monitors and adapts to the state and use of its computing resources. Results that come out of the run time are acted upon by processes that are running externally to Streams such as an alert to the administrator or data to visualize on a map. For more information, see Chapter 3.2, “InfoSphere Streams runtime system” on page 45.

## Programming and debug environment

As shown in Figure 3-2, IBM InfoSphere Streams Studio is an Eclipse-based developers and administrator's workbench. By using this tool, you can create, edit, compile, visualize, test, debug, and run Streams applications.

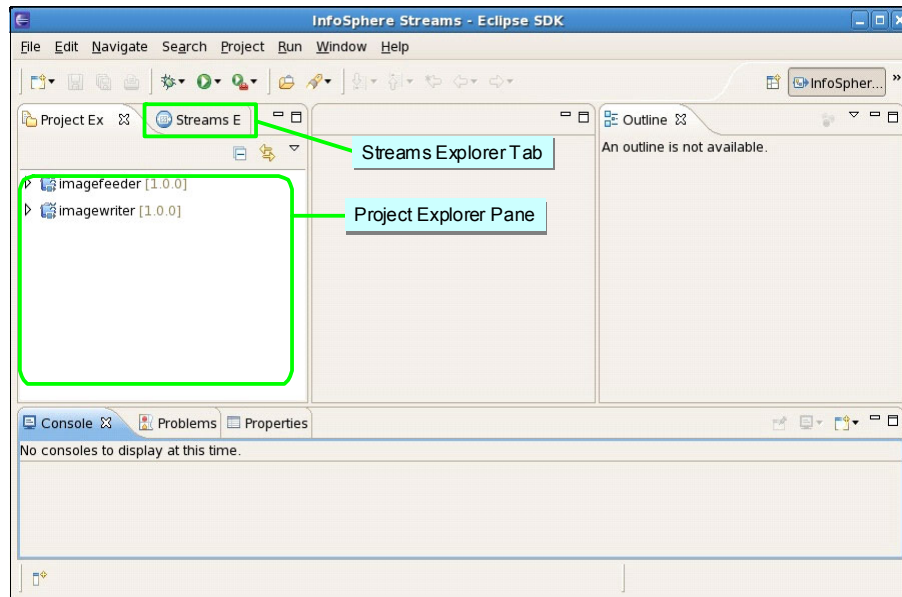


Figure 3-2 IBM InfoSphere Streams Studio

### **Streams Explorer tab**

From the Streams Explorer tab, you can perform the following tasks:

- ▶ Create, start, and stop instances.
- ▶ Manage jobs, see them running in an instance, or cancel a job.
- ▶ View metrics to analyze performance and view the health of running applications and flow summaries.
- ▶ Log viewing support for monitoring applications and see alerts.
- ▶ Graphs for visualizing running applications.

### **Project Explorer tab**

From the Project Explorer tab, you can manage the following applications and toolkits:

- ▶ Editors for creating new Streams applications
- ▶ Compile and deploy Streams applications
- ▶ Debugger for testing Streams applications
- ▶ Supports graph drag and drop composites

## **Tools and administrative resources**

IBM InfoSphere Streams includes a `streamtool` interface. This interface is a command-line program that is used to perform many tasks, such as listing Streams instances and starting and stopping instances. However, you are encouraged to use InfoSphere Streams Administration Console, which is a web-based GUI that is provided by the Streams Web Services (SWS). For more information, see “Management Services” on page 49.

Complete the following steps to open the Administration Console in your browser:

1. As shown in Example 3-1, from a command line, enter the following command to retrieve the web address for the Administration Console:

```
streamtool geturl -i <instance_name>
```

*Example 3-1 Shows the command to retrieve the Administration Console web address*

---

```
streamtool geturl -i streams@user1
```

---

The URL is shown.

2. Enter the URL into the web address field of your browser.
3. When you are prompted, enter the instance user ID and password as specified in the security configuration for the Streams instance.

For more information about the Administration Console, see Chapter 8, “IBM InfoSphere Streams administration” on page 199.

## 3.2 InfoSphere Streams runtime system

For a better understanding of the InfoSphere Stream runtime system, we examine the development of an application. We also observe the environment behavior and how the services can work together for best performance results.

### 3.2.1 SPL components

InfoSphere Streams offers the IBM Streams Processing Language (SPL) interface for users to operate on data streams applications. Users can create applications without needing to understand the lower-level, stream-specific operations. SPL provides numerous built-in operators, the ability to import data from outside InfoSphere Streams and export results outside the system, and a facility to extend the underlying system with user-defined operators. Many of the SPL built-in operators provide powerful relational functions, such as Join and Aggregate.

Deploying SPL applications results in the creation of a data flow graph that is supported by the underlying runtime system. As new workloads are submitted, the InfoSphere Streams runtime system determines where to best deploy the operators to meet the resource requirements of both newly submitted and already running specifications. The runtime system continuously monitors the state and usage of its computing resources. When applications are running in the InfoSphere Streams environment, they can be dynamically monitored across a distributed collection of hosts by using the Streams Studio.

Results from the running applications can be made available to applications that are running external to InfoSphere Streams by using Sink operators or edge adapters. For example, an application might use a TCP Sink operator to send its results to an external application that visualizes the results on a map, or alerts an administrator to unusual or interesting events.

SPL includes the following main components:

▶ Tuple

Tuple is an individual piece of data in a stream. It contains the data, and has a fixed set of Attributes (also called variables). This data can be structured or unstructured. Often, the data in a tuple represents the state of something at a specific time; for example, a stock ticker quote or a temperature reading from an individual sensor. The schema describes the data types in the tuple.

▶ Projects

A Streams project serves as a container for a Streams application (there is a one-to-one correspondence of project-to-application). A Streams project contains other properties and configuration settings that are related to a Streams application.

▶ Applications

A Streams application can be submitted for execution in the form of a Streams Job. Generally, there is a one-to-many relationship of applications to jobs. As examples, an application might not be running (zero jobs), running only once (one job), or there might be several copies of the same application that are running (many concurrent applications, or more accurately, many concurrent jobs). Applications can be defined to accept runtime parameters so that each of the concurrently running jobs are performing distinct processing tasks.

▶ Operators

A primitive operator is the basic building block of an SPL application. They encapsulate code in native language so you can write your own primitive operator in C++ or Java. There are a number of primitive operators that are supplied with IBM InfoSphere Streams, but these supplied operators do not cover all customer needs. By using the product, you can create other operators. Some primitive operators, such as sort, filter, join, merge, and sound familiar and do roughly similar things to what they mean in SQL.

Composite operators are written in SPL. They contain a stream subgraph of primitive and possibly other composite operators. Composite operators can include input and output ports, which allowing tuples to be passed to the composite operator and modified tuples to be returned.

We encourage you to make your composite operators reusable and put them into libraries so they can be started in different programs. You also can use parameterizable composite operators to implement distributed computing patterns. For more information, see *IBM InfoSphere Streams: Assembling Continuous Insight in the Information Revolution* (SG24-7970), and 6.1, “InfoSphere Streams Studio overview” on page 106.

**Important:** During compilation, the operators are fused into *partitions*. A partition can have one or more operators. A partition is deployed as a processing element (PE) or a process.

## 3.2.2 Runtime components

As shown in Figure 3-3, the InfoSphere Streams runtime system is a collection of components and services. They are designed to work together to provide scalability and capability on high-volume streaming data.

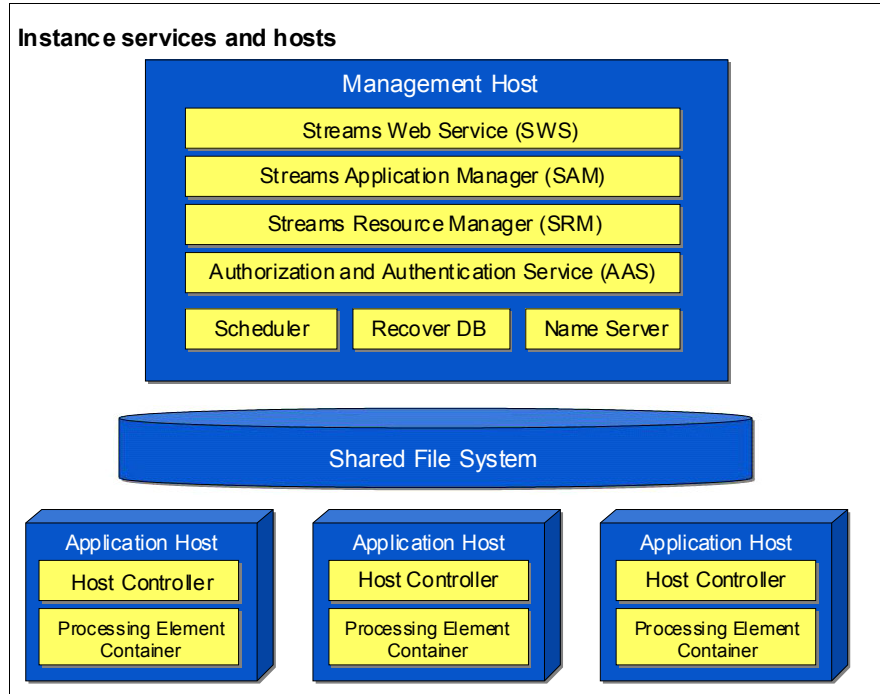


Figure 3-3 IBM InfoSphere Streams runtime components diagram

The IBM InfoSphere Streams runtime consists of the following components:

- Instances

A single instantiation of the Streams runtime system is called an instance. It is composed of a set of interacting services that are running across one or more host computers. There can be more than one instance that is created for an installation. An instance can be private or shared. Each instance is independent of the others and jobs are not shared across instances.

Streams instances are started and then stopped, and the entities that are contained inside a Streams instance are modified, compiled, submitted, deployed, and so on.

From an administrative standpoint, a Streams instance is created and then can be deleted. Creating a Streams instance is a relatively low-cost procedure.

► Hosts

A Streams host is a physical term that usually equates with a single operating system host. In certain contexts, a Streams host is also called a node. In order for it to exist, a Streams instance (a logical term) must be composed of one or more hosts (a physical term). A Streams host is one of the following types exclusively:

- A Management host runs the resident services that make up the Streams runtime environment proper. For more information, see “Management Services” on page 49.
- An Application host is dedicated to running the actual Streams applications.
- A Mixed-Use host runs resident services (all or a subset of these services) and Streams applications, and they are more common in development environments than in production environments.

► Host Controller

The Host Controller (HC) is an application service that runs on every Application host in an instance. This service carries out all job management requests that are made by the Streams Application Manager (SAM) service, which includes starting, stopping, and monitoring Process Elements (PEs). This service also collects all performance metrics from PEs and reports the results to the Streams Resource Manager (SRM) service. For more information about SAM and SRM services, see “Management Services” on page 49.

► Process Element

When you compile an SPL application, the operator and streams relationships that make up the data flow graph are broken down into a set of individual execution units that are known as PEs. Each PE can contain one or more fused (at compile time) operators, which also are known as a partition. PE is the execution container of a partition and each partition runs in one PE.

► Jobs

In addition to a set of PEs, the SPL compiler generates an Application Description Language (ADL) file that describes the structure of the application. The ADL file includes details about each PE, such as which binary file to load and run, scheduling restrictions, stream formats, and an internal operator data flow graph.

**Tip:** To run an application by command line, you submit a job to InfoSphere Streams by using the `streamtool submitjob` command and specifying the ADL file for the application.

► Recovery Database

The InfoSphere Streams recovery database is a DB2 database that records the state of instance services. When instance services fail, InfoSphere Streams restarts them and restores their state by retrieving the appropriate state data from the recovery database. This feature is optional.



## Management Services

All Management hosts run the following services:

- ▶ Streams Resource Manager  
The SRM service starts a Streams instance and aggregates system-wide performance metrics. The SRM service also is the service that interacts with the host controller of each application host.
- ▶ Streams Application Manager  
The SAM service receives and processes job submission and cancellation requests, primarily through its interaction with the Scheduler service.
- ▶ Scheduler  
The SCH service gathers runtime performance metrics from the SRM service and feeds that information to the SAM service to determine which Applications host or hosts are instructed to run given a Streams application (Processing Element Containers).
- ▶ Authorization and Administrative Service  
The Authorization and Administrative Service (AAS) provides access control lists (ACL) for administrative and user groups that are set on a user basis, such as permissions to read/write to various objects. ACLs can be added, removed, changed, and viewed. For authentication, the default setting is Pluggable Authentication Modules (PAM) with RSA where public and private keys are stored to authenticate users so that tasks can be run. You also can use LDAP Authentication. Streams also supports Secure Shell (SSH) to run various system commands.
- ▶ Streams Web Service  
The Streams Web Service (SWS) is optional and must be running if the Streams Console (a browser-based graphical interface for managing system resources and application and monitoring performance) is used.

An Application host runs the following services:

- ▶ Host Controller (HC): Manages PEs and collects PEs performance metrics.
- ▶ Processing Element Container (PEC): The PEC exists as an operating system binary program. They are started and monitored by the HC for the host on which the PE is running. PEs exist as custom-shared libraries and are loaded by PECs as required by the Streams application definitions.

**Important:** Mixed-used hosts can run all of the listed services, but are not recommended for production environments.

### 3.2.3 InfoSphere Streams runtime files

The InfoSphere Streams runtime system uses the installation, configuration, and user authentication files that are described in this section. In addition, InfoSphere Streams generates log and trace files that you can use to correct problems with the product and applications.

#### InfoSphere Streams installation files

The InfoSphere Streams installation utility copies the installation files into the path that is specified by the user who is installing the files.

**Important:** The following considerations must be remembered when InfoSphere Streams installation files are used:

- ▶ If you are running the installation utility as a non-root user, you are the owner of the InfoSphere Streams installation.
- ▶ If you are running the installation utility with root authority, the utility prompts you to specify the user and group that owns the installed files. The root user cannot own the installed files.

For root and non-root installations, the owning user ID and group for the installation must exist before the InfoSphere Streams installation utility is run. For multiple-user installations that are running shared instances, the preferred method is to use a non-personal administrator ID such as `streamsadmin`.

InfoSphere Streams does not add, remove, or modify files in the installation path. These actions occur only when you install or uninstall the product.

After the installation completes, you enter the source `<product-installationdirectory>/bin/streamsprofile.sh` command to establish the copy of InfoSphere Streams that is active and initialize the Streams environment on your session. You also can add this command to your shell initialization file; for example, `~/.bashrc`.

## InfoSphere Streams global configuration files

The global configuration files are in the `~/.streams` directory of the installation owner. InfoSphere Streams creates this directory during the installation if the directory does not exist.

If a user owns multiple InfoSphere Streams installations, all installations use the same global configuration directory. Therefore, be careful when you are editing these files because multiple installations might reference the same file. Instance configuration data is also stored in the global configuration directory and can be used by multiple installations. However, the instance can be used only by the version of InfoSphere Streams that was used to create it.

If InfoSphere Streams is uninstalled and not reinstalled, you must manually remove these files.

## InfoSphere Streams instance configuration files

When you create an InfoSphere Streams instance, configuration information for the instance is stored in your `~/.streams` directory. As you refine your instance configuration, those changes also are stored in this directory.

Although you create an instance by using the tools from an InfoSphere Streams installation, instances themselves are separate from any InfoSphere Streams installation. If you uninstall the product, the instance configuration files remain untouched. If you do not intend to use this instance in a new or different installation copy of InfoSphere Streams, run the `streamtool rminstance` command before the product is uninstalled to remove the instance and all of the configuration files.

If your instances include business value, it might be worthwhile to save your instance settings by backing up your `~/.streams` directory.

## InfoSphere Streams instance log and trace files

When an InfoSphere Streams instance is active, the runtime components and any running applications can generate log and trace data that can help you to identify, diagnose, and resolve problems.

Log and trace files are on the hosts where the InfoSphere Streams runtime services and applications are running. The log and trace files for a PE are on multiple hosts if the PE is moved during its execution.

The default root directory for all log and trace files is the `/tmp` directory. You can configure a different root directory by updating the `LogPath` property for the InfoSphere Streams instance. The names of the subdirectories under your configured root directory cannot be changed.

### 3.3 Performance requirements

One of the main goals of Streams is to deliver high throughput and low latency computation. To achieve this task, the performance requirements of the system must be well-understood. In this section, we describe a number of aspects of the performance requirements that should be addressed.

You should determine the expected volumes for each data source, and consider the following measures:

- ▶ The average throughput over a sustained period.
- ▶ The expected peak throughput in any short period.
- ▶ The growth in data volumes that is anticipated in the future.

Knowing the average and peak throughput, for example, allows you to determine the power and quantity of physical hosts that are required by your target application. Streams has a low impact on the data path in exchange for the scale up and scale out features it provides and the application agility. When an application is sized for Streams, the best possible performance from a manually coded distributed application is a good starting point.

There are cases when it is not possible to process all data by parallelizing or distributing because the data rates are too high or the required processing is too complex. In those cases, you must determine an acceptable alternative when the volume of data from a data source exceeds what the system can handle. The following alternatives are available:

- ▶ You can discard excess data. The Source and associated operators can be designed to act as a throttle, which filters the incoming data to a specific rate. It might also be possible to combine multiple records into aggregates (which reduces the number of records that are processed downstream) or to discard excess tuples by a random process of sampling, by identifying tuples of lower potential value, or by shedding the most or least recently received tuple that uses a `ThreadedPort`.
- ▶ You can buffer excess data in the expectation that this data is a peak load period and as the load declines, the system can process that excess data. The use of this approach increases the average latency (the time it takes to process each individual entry) because latency also includes the time that a tuple is buffered and waiting to be processed. As such, this approach might be undesirable in an extreme low latency focused system. However, many applications are not so sensitive and still provide better latency than alternative approaches.
- ▶ You can control the throughput of select flows by using the Throttle operator. The Throttle operator is provided in the Standard toolkit and can pace a stream at a rate that is specified in tuples per second.

Extreme performance requirements might lead you to segment the application to enable processing of smaller chunks, which can then be deployed to separate hosts and processed in parallel.

### 3.3.1 InfoSphere Streams reference architecture

InfoSphere Streams provides a computing platform to help companies turn burgeoning data volumes into actionable information and business insights. As a critical element of the IBM platform for big data, InfoSphere Streams delivers a highly scalable, agile software infrastructure. This infrastructure enables businesses to perform in-motion analytics on various relational and non-relational data types at unprecedented volumes and speeds from thousands of real-time sources. With InfoSphere Streams, your enterprise can capture and act on key business data just in time, all of the time.

Big data projects are not all alike. What works for one company might not work for another. Instead of reinventing the wheel to apply business analytics to big data, it makes business sense to take advantage of what is proven. This fact is especially true when a reference architecture that was developed by IBM is considered, a company with experience that is based on thousands of IT infrastructure engagements in wide-ranging environments for companies of all sizes in virtually every industry.

Reference architectures for big data analytics provide technical blueprints that include a well-defined scope, a complete listing of requirements, and architectural decisions that are proven in the field, including IBM Intelligent Cluster™, which helps make complex solutions simple. A reference architecture provides the value of synergy among each of the solution building blocks, with the flexibility that is needed to meet your requirements. The following advantages can be gained by using a reference architecture for big data from IBM:

- ▶ **Ease of integration**

Deployment by using a reference architecture helps ensure the new solution works with what is in place and is likely to be added later, such as data warehouse, stream compute engines, and internal and external storage devices.

- ▶ **Flexibility and simplicity**

A reference architecture from IBM helps strike a cost-effective balance between the requirements of the solution and time-to-value, which offers flexibility for your enterprise to adapt as your big data requirements evolve.

- ▶ **Faster deployment**

A reference architecture that specifies IBM pre-integrated components can also include Intelligent Cluster, an integrated, factory-built, and factory-tested cluster solution that you can roll into your data center and put to work right away. Add to this configuration an optimized software stack that relies on open source code and you can get the total solution up and running faster, with fewer headaches and much less trial and error.

- ▶ **Quicker access**

Every business is looking for fast time-to-value. There is no let up in data volumes, economic pressures, or competitive demands. The accelerators and pre-built analytics tools for developers and business users that are incorporated in the reference architecture mean solutions are working harder sooner.

#### **Design Cluster Architecture**

Streams is flexible, which means that you can start small and grow gradually. A single node can do much interesting work in Streams with good performance but does not provide the redundancy that it required for high availability.

When performance needs grow, add one or preferably, two nodes. Three nodes give an ideal level of availability at minimum cost. Add more nodes as data rates and analytical requirements grow.

Critical Streams instance configuration and name service data is maintained in directories under the instance owner's home directory. This directory must be on a shared file system so it can be accessed by all machines in the cluster. This system can be a local file system that is mounted through a network file system (NFS), but failure disables the entire instance (the instance cannot even be restarted). Replication of that file system is a key part of high availability. The use of IBM General Parallel File System™ (GPFS) is one way to accomplish replication. GPFS a robust redundant file system from IBM that has no single point of failure.

Streams management services can recover their state after failure without having to restart the entire instance if you set up a small DB2 database recovery (product and license are provided with Streams). To make the recovery database highly available, use High Availability Data Replication (HADR) (a feature of DB2) to manage redundant storage and failover across two of the cluster's nodes. To avoid capacity constraints, add one disk per node.

These points do not prevent application data loss in case of failure. This prevention requires redundancy of process flow, or some kind of unit-of-work replay capability. This requirement must be designed into the application in a way that matches the nature of the data feeds. The Streams development paradigm of a processing flow graph makes this configuration relatively easy to accomplish, but it goes beyond the topic of this section of the book, except that more than one machine is a minimum requirement.

### **System x: Reference for big data**

We see in this chapter that InfoSphere Streams provides a development platform and runtime environment that can ingest, filter, analyze, and correlate massive volumes of continuous data streams. These data streams include text, spreadsheets, images, video and audio recordings, email, instant messages, financial transactions, and customer records. InfoSphere Streams fuses these heterogeneous data types onto a powerful computing platform that enables complex data analysis with exceptional performance and impressive response times. Stream processing requires resilient, memory-rich servers to ensure continuous delivery of insight.

The core reference architecture is built with the x3550 M4 and x3690 X5. The x3630 M4 offers a cost-effective option to achieve higher performance for data-intensive workloads in a two-socket, 2U design. The x3690 X5 is a two-processor server that features the latest enterprise IBM X-Architecture® technology (eX5) for extreme flexibility and enterprise-class availability and reliability at a lower total cost of ownership. Designing for stream workloads requires an understanding of core design characteristics, such as ingest streaming rates, data packet sizes, required output rates, latency, and quality of service (QoS) requirements. These factors affect architectural design considerations, such as node design, processor speed, network bandwidth, and even server redundancy for mission-critical applications. The reference architecture approach addresses these requirements to ensure faster, lower-risk implementations. System x intelligent, industry-leading, enterprise x86 systems are designed to reduce costs and simplify your IT infrastructure. Managing energy in the data center is a growing concern because of the increasing numbers of servers, the incremental heat they generate and, ultimately, the rising cost of energy. System x servers not only help increase performance per watt, but help you budget, plan, and control power usage.

Figure 3-4 shows a medium configuration of four standard Streams nodes and nine BigInsights nodes. Two network switches at the top of the rack feature a data center-standard for redundancy and availability.

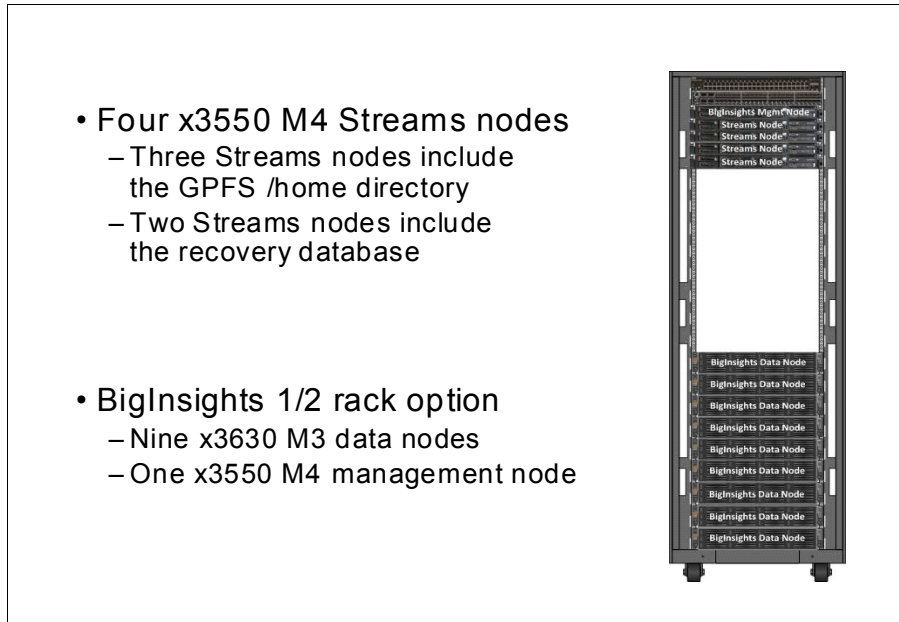


Figure 3-4 Example of Streams and BigInsights cluster

### Why IBM?

The IBM commitment to understanding, exploring, and refining analytics also includes the investment of more than USD14 billion in analytics acquisitions of market leaders, such as IBM SPSS, IBM Cognos, IBM Unica®, IBM Coremetrics®, and IBM Platform Computing. Consider the IBM investment in analytic intellectual property within Netezza or the IBM Smart Analytics System. The fact that IBM has a business analytics and optimization (BAO) division represents the kind of long-term capabilities IBM is committed to delivering for analytics in its big data platform. In May 2011, IBM announced a USD100 million investment in massive scale analytics that focused primarily on refining Hadoop. It is also likely that no other vendor can talk about and deliver analytics for big data at rest and big data in motion together. IBM can make this scale of commitment in large part because of a century-old track record of successful innovation. IBM has the single largest commercial research organization in the world, with more than 200 mathematicians who are developing breakthrough algorithms. Today, IBM has more than 9,000 experienced strategy, analytics, and technology experts and consultants. IBM provides thought leadership and practical insights from the IBM Institute for Business Value and offers proven solutions and use cases across almost every industry and functional area.

### For more information

For more information about IBM System x reference architecture solutions for big data, contact your IBM representative or IBM Business Partner, or visit this website:

<http://www.ibm.com/systems/x/bigdata>

## 3.4 High availability

InfoSphere Streams includes several features that enhance high availability and redundancy. Administrators can add or remove processor nodes to or from the cluster. The ability to seamlessly add or remove nodes to or from the cluster allows administrators perform necessary maintenance without shutting down the InfoSphere Streams application. This ability can help improve overall availability of the environment. Also, when the management nodes that control the system fail, the application nodes continue to run the InfoSphere Streams applications. Also, the management nodes can be restarted by the administrators later.

The new programmatic manipulation of streaming data makes it easier to create highly available applications by using redundant application components. New dynamic allocation of host pools and more location and isolation constraints of nodes in the runtime cluster can be used to isolate redundant application components to separate nodes in the run time. The following features are aimed at enhancing high availability aspects that are related to InfoSphere Streams:

- ▶ PE: The system provides support for applications to keep running despite crashes in parts of the application itself. The system attempts to restart a PE that crashes.
- ▶ Application Host: The system copes with faults that occur on any system that is running Streams applications.
- ▶ Management Daemon: The system maintains a coherent state and recovers when faults happen with the Management daemons or the system on which they run.
- ▶ Instance Shutdown: The system shutdown operation.







# IBM InfoSphere Streams V3.0 new features

This chapter gives a high-level overview of the following new features of IBM InfoSphere Streams V3.0:

- ▶ New configuration features
- ▶ Drag-and-drop application development
- ▶ Enhanced administration tool
- ▶ Integration with other tools
- ▶ Analytical and accelerator toolkits

## 4.1 New configuration features

Streams 3.0 provides straightforward and time-saving processes in post-installation configuration.

### 4.1.1 Enhanced first steps after installation

Streams V3.0 provides an easy post-installation configuration for the following items:

- ▶ Configure the Secure Shell (SSH) environment: Mandatory menu to set up an SSH environment by a single click of this item.
- ▶ Configure Streams environment variables: A single-click is all that is needed to set all Streams variables.
- ▶ Generate public and private keys: Provides a feature to eliminate the need to enter a password when you are running a command that requires authentication.
- ▶ Configure the recovery database: Configures DB2 database for Streams recovery needs.
- ▶ Verify the installation: Runs tests to verify the installation and runtime environment.
- ▶ Create and manage Streams Instances: Runs the instances manager to organize the runtime instances; for example, creating and starting an instance.
- ▶ Develop an application with Streams Studio: After the installation finishes, this menu installs Streams Studio, which the user can then open.
- ▶ Migrate Streams configuration: This configuration moves global configuration and runtime instances to this new version.

For more information, see Appendix A, “Installing InfoSphere Streams” on page 249.

## 4.2 Development

Previously, enterprises saw Streams as a reliable solution for data streaming needs. However, users that are not programmers had difficulties with Streams application development. Streams V3.0 provides an easy to use, drag-and-drop editor for Streams application development.

Streams now features a development environment called Streams Studio. Formerly, developers needed a deep knowledge of how to use Streams Programming Language (SPL) in Studio. Now, Streams V3.0 provides a drag-and-drop editor to develop Streams applications.

The presence of a drag-and-drop editor does not mean that SPL is gone. There is still SPL associated with the graphical design of a Streams application. The graphical design and SPL code are synchronized with each other so if you change one, the other is updated. For more information about this feature, see Chapter 6, “Application development with Streams Studio” on page 105.

## 4.3 Administration

In this section, we describe the new features of Streams V3.0 regarding enhancements to administration functions.

### 4.3.1 Improved visual application monitoring

Previously, the instance graph that includes information about Streams applications, such as job, processing element (PE), operator, and host, was available only on the Streams studio. Streams V3.0 now provides visual application monitoring on the Streams studio and on the Streams web console. This functionality easily identifies issues by using customizable views by job, PE, operator, and host. For more information about this feature, see 8.3, “Instance administration” on page 210.

### 4.3.2 Streams data visualization

One of the highlights of Streams V3.0 is data visualization. With the use of this feature, live data streaming that is processed by an application can be visualized directly on the Streams web console. Several charts are provided to visualize the data. For more information about the data visualization feature, see 8.3.7, “Application streams” on page 225.

### 4.3.3 Streams console application launcher

The Streams console application launcher feature allows a Streams developer or administrator to monitor and manage streams applications that were started by the Streams web console. Formerly, a developer or administrator needed to open Streams Studio to start or manage Streams applications. For more information about this feature, see 8.3.3, “Applications” on page 219.

## 4.4 Integration

Streams V3.0 enables integration with other platforms, such as IBM DataStage®, Netezza, Data Explorer, SPSS, and XML data types.

### 4.4.1 DataStage integration

Streams V3.0 enables integration with DataStage by using adapters to exchange data between Streams and DataStage. For more information about this feature, see 7.4, “Streams and DataStage” on page 182.

### 4.4.2 Netezza integration

By using the new Netezza integration, you can insert data into Netezza by using its native high-speed loader. For more information, see , “Netezza” on page 178.

### 4.4.3 Data Explorer integration

Streams V3.0 uses the InfoSphere Data Explorer toolkit to push streams data into the Data Explorer index. For more information about this feature, see 7.7, “Integration with Data Explorer” on page 196.

### 4.4.4 SPSS integration

By using this new feature, you can deploy models into InfoSphere Streams via IBM SPSS Statistics scoring operator. For more information about this feature, see 7.2, “Integration with IBM SPSS” on page 168.

### 4.4.5 XML integration

Streams V3.0 enables easy integration with XML formats on operations, function, standard toolkit adapter, and so on. For more information about on this feature, see 7.5, “Integrating with XML data” on page 187.

## 4.5 Analytics and accelerators toolkits

Streams V3.0 comes with several helpful toolkits and accelerators.

### 4.5.1 Geospatial toolkit

This toolkit provides high-performance analysis and processing of geospatial data. It enables location-based services, geospatial data types (such as point and polygon), and geospatial functions (such as distance between one location to another). For more information, see 6.5.5, “Geospatial toolkit” on page 156.

### 4.5.2 Time series toolkit

Time series toolkit helps you find and determine patterns and anomalies about data streaming. For more information about this toolkit, see 6.5.6, “TimeSeries toolkit” on page 156.

### 4.5.3 Complex Event Processing toolkit

The Complex Even Processing (CEP) toolkit uses patterns to detect composite events within streams of simple events. The pattern uses standard regular expressions. For more information, see 6.5.7, “Complex Event Processing toolkit” on page 157.

### 4.5.4 Accelerators toolkit

#### **Social media accelerator toolkit**

The following tasks can be performed with social media accelerator:

- ▶ Lead Generation
- ▶ Brand Management
- ▶ Micro-segmentation attributes: Segmentation that is based on several attributes, such as gender, location, parental status, marital status, and employment.
- ▶ Output: Output of measures such as intent to buy a stock (which is shown in the stock market example in 6.2.3, “Creating a simple application with Streams Studio” on page 114.

For more information about the social media accelerator toolkit, see , “Social media accelerator toolkit” on page 158.

#### **Telco accelerator toolkit**

Telco accelerators provide framework to work with Call Detail Records (CDRs). For more information, see , “Telco accelerator toolkit” on page 158.





# InfoSphere Streams deployment

In this chapter, we describe InfoSphere Streams runtime deployment, which includes planning, and configuration of the InfoSphere Streams run time on Linux clusters and configuration of Streams instances. In addition, we describe Streams application deployment capabilities.

This chapter includes the following topics:

- ▶ InfoSphere Streams runtime deployment:
  - Streams runtime architecture and services
  - Streams instances, and deployment topologies
  - Planning a Streams runtime deployment
  - Pre- and post-installation and configuration of the Streams environment
- ▶ InfoSphere Streams instance creation and configuration:
  - Streams shared instance planning and configuration
  - Streams private development instance configuration
- ▶ InfoSphere Streams application deployment capabilities:
  - Dynamic application composition
  - Operator placement
- ▶ InfoSphere Streams failover and recovery:
  - Processing element recovery
  - Streams runtime services recovery

## 5.1 Architecture, instances, and topologies

In this section, we describe the Streams runtime architecture, instances, and deployment topologies.

### 5.1.1 Runtime architecture

Much like database management systems and Java Platform, Enterprise Edition application servers, InfoSphere Streams features a runtime engine that Streams administrators define, configure, and run. The term *Streams instance* refers to the software configuration that is run on one or more hosts (servers) that provides an environment for Streams applications to be started, stopped, managed, and monitored.

A Streams instance consists of a number of service processes that interact to manage and run the Streams applications. Separate Streams instances can be created and configured independently from each other even though they might share some of the same physical hardware. Figure 5-1 shows a Streams instance with its constituent services that are described in the following sections.

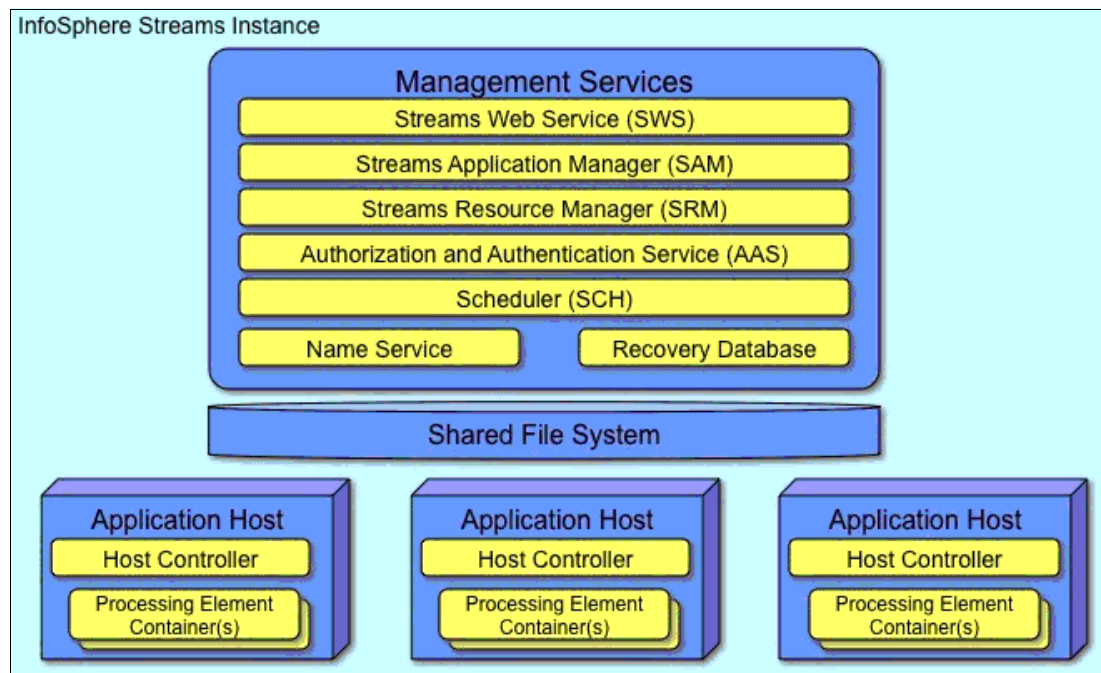


Figure 5-1 Services that are associated with the Streams instance



## Management services

There is a single set of management services for each Streams instance. If the instance is instantiated on a single host, all services are run on that single host. If the instance is spread across multiple hosts, the management services can be placed on different hosts or co-located together on a single host. The following management services are included:

- ▶ Streams Application Manager

The Streams Application Manager (SAM) is a management service that administers the applications in the Streams runtime system. The SAM service handles job management tasks, including user requests such as job submission and cancellation. This service interacts with the Scheduler to compute the placement of processing elements (PEs) that are associated with an application. It also interacts with the Host Controller to deploy and cancel the running of PEs.

- ▶ Streams Resource Manager

The Streams Resource Manager (SRM) is a management service that initializes the Streams instance and monitors all instance services. SRM collects runtime metrics on the instance hosts and Streams components. This service also collects performance metrics that are necessary for scheduling and system administration by interacting with the Host Controller.

- ▶ Scheduler

The Scheduler (SCH) is a management service that computes placement decisions for applications that are deployed in the Streams runtime system. The SCH service interacts primarily with the SAM service to handle job deployment requests and with the SRM service to obtain the set of hosts that can be used for deployments. The SCH service also interacts with the SRM service to collect runtime metrics that are necessary for computing PE placement recommendations.

- ▶ Name service

The name service (NSR) is a management service that stores service references for all of the instance components. This service is used by all instance components to locate the distributed services in the instance so that Streams components can communicate with each other.

- ▶ Authentication and Authorization Service

The Authentication and Authorization Service (AAS) is a management service that authenticates and authorizes operations for the instance.

- ▶ Streams Web Service

The Streams Web Service (SWS) is an optional management service that provides web-based access to instance services. To use the Streams Console, SWS must be running on a host in the instance.

- ▶ Recovery database

Streams instance recovery is an optional configuration setting. If recovery is configured, the Streams recovery database records the state of instance services. When instance management services fail, Streams restarts them and restores their state by retrieving the appropriate state data from the recovery database.

## Application host services

The following application host services run on each host in a Streams instance that is available to run Streams applications:

- ▶ Host Controller

The Host Controller (HC) is an application service that runs on every application host in an instance. This service carries out all job management requests that are made by the SAM service, which includes starting, stopping, and monitoring PEs. This service also collects all performance metrics from PEs and reports the results to the SRM service.

- ▶ Processing Element Container

Streams Processing Language programs consist of operators, which are grouped at compile time into partitions. The execution container for a partition is a PE. Each partition runs in one PE. PEs are loaded by the Processing Element Container (PEC), which is started and monitored by the Host Controller for the host on which the PE is running.

## Shared file system

A multi-host Streams instances configuration requires a shared file system to share instance configuration information and SPL application binaries across the hosts.

Technologies that are used to implement a shared file system must be Portable Operating System Interface for UNIX (POSIX) compliant, and include, but are not limited to, the following systems:

- ▶ Network File System (NFS)
- ▶ IBM General Parallel File System (GPFS)

### 5.1.2 Streams instances

The Streams runtime environment is configured as a Streams instance. Each instance operates as an autonomous unit and can be configured with multiple options. In this section, we look at the most commonly used options for instances. For a complete set of options, see the **streamtool man command** in *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888, and the IBM InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

You can also create new instances with a new application called the Instances Manager. The new function allows the user to create an instance. Currently, only a subset of properties can be specified by this interface. After the instance is created and started, the user can use Streams Console to change the properties of the instance.

Each Streams instance is autonomous and isolated from other instances. You can create multiple instances on the same Streams installation, but this option is not a best practice. Even though instances can share physical resources, there is no logical interference with other instances. (The use of multiple instances on the same set of physical resources can cause physical resource contention because the services for each instance are competing for the same CPU cycles and networking resources.) Streams SPL applications are run on an individual instance, and no other instances are aware of the jobs.

## Streams instance IDs

A unique instance ID identifies each Streams instance within a Streams environment. The format of a fully qualified Streams instance ID is `instance-name@userid`, where `instance-name` provides a meaningful name for an instance to identify one from another in a multi-instance environment, and `user ID` is the user name of the person who created the instance (instance owner). The “@” character is used to separate the `instance-name` from the user ID.

The instance-name must satisfy the following requirements:

- ▶ The name must be unique among the instances that are defined by you, the instance owner.
- ▶ The name must contain alphanumeric characters and start with an alphabetic character. Each of the alphanumeric characters must be one of the following ASCII characters: 0 - 9, A - Z, and a - z.
- ▶ The name can contain a dash (-), an underscore (\_), or a period (.

The following examples of instance IDs are valid and unique in a multi-instance environment:

- ▶ `streams@streamsadmin`
- ▶ `lab-inst01@streamsadmin`
- ▶ `stream_dev@mary`
- ▶ `stream_dev@john`

Use the following guidelines when you use Streams instance IDs to specify Streams instances:

- ▶ The instance owner can use the full name of the Streams instance ID, or the just the instance name. The system assumes the user ID is the user name of the current user.
- ▶ Users that are accessing an instance that was created by a different user must use the full name of the Streams instance ID (`instance-name@userid`), which specifies the user ID of the instance owner.

## Streams instance configurations

To decide how many instances you need and how they are configured, it is important to understand a couple of basic ways to categorize instances from a planning perspective. Streams instance configurations are characterized by configuration type and usage pattern or purpose. The configuration type is specified when you create a Streams instance, but the usage purpose is chosen and governed by you.

### *Instance configuration types*

You can set an instance to be private or shared, depending on the access control lists (ACLs) that are set up for that instance. You can take an instance that is created without any ACLs (private) and add ACLs so that it is shared. Private instances do not support multiple users, thus they are not often used in production.

There are templates that can be specified in the `mkinstance` command that help create the instance. There also is an option on the New function panel in the Instances Manager that can be selected to create a shared instance.

### ***Instance usage purpose***

As you plan your Streams environment, you should plan to separate the following development activities, integration and test activities, and production operations into separate instances:

- ▶ **Development instances**

Development instances often are created as private (single-user) instances, which use a single or small number of hosts. The primary purpose of these instances allows a developer to develop and test Streams applications, generic primitive operators, native functions, and so on.

It is common for multiple development instances to share physical hardware.

- ▶ **Integration and Test instances**

Integration and Test instances often are created as shared instances, which mimic a production environment. If separate Test and Integration instances are created, the Integration instance can use a smaller number of hosts than the production instance, but where possible, testing instances should provide an environment that is configured as much like the production instance as possible.

The primary purpose of Integration and Test instances are to provide an environment where multiple users can run their applications in a single environment and verify adequate resource usage and availability, sharing of imported/exported streams, and cohabitation of applications.

It is a best practice that Integration and Test instances be run on dedicated physical hardware.

- ▶ **Production instances**

Production instances are primarily created as shared instances on a dedicated set of physical resources. These instances should be monitored for resource utilization to identify CPU and I/O saturation.

### **Streams instance files**

In addition to the runtime processes that make up a Streams instance, there are file system-based components. Each Streams instance interacts with configuration, user authentication, and log files, which provide the persistent aspects of the instance.

#### ***Streams instance configuration files***

The configuration information for each instance is stored in the `~/streams/instances` directory of the instance owner (“~” is the Home directory in Linux). As configuration changes are made to the instance, these files are updated to persist the configuration.

Although instances are created by using the tools from a Streams installation, instances themselves are separate from any Streams installation. If Streams is uninstalled, the instance configuration files remain untouched.

**Backing up directories:** Each user’s `~/streams` directory should be backed up, especially the instance owners’ `~/streams` directories.

### Streams instance user authentication files

The public and private keys that Streams uses to authenticate Streams instance users are stored in the default location of `~/streams/instances/instance-name@userid/config/keys` or a directory that is configured by the instance owner. For more information, see “Security Public Key Directory” on page 79.

The authentication files also should be backed up regularly.

### Streams instance log files

When a Streams instance is active, the runtime components and any running applications can generate log file content. Log files are in `/tmp/streams.instance-name@userid`. This directory exists on each host that is part of the instance, but the contents of the directory is different for each host, depending on what management services are running on the host. Within this directory are two subdirectories: `logs` and `jobs`. The `logs` directory contains the logs for any management services that are running on the host. The `jobs` directory contains a separate directory for each job (for example, Streams application) that is running on the host. Each of the jobs subdirectories contain the log files for the processing elements that is running on that host.

When you stop a Streams application job, its corresponding log files are removed unless you use the `collectlogs` parameter in the `streamtool canceljob` command.

## 5.1.3 Deployment topologies

In preparation for planning a Streams runtime deployment, an understanding of different deployment topologies is necessary. In this section, we look at three of the most common topologies.

### Single Host Topology

The simplest topology is a single host computer that is running a single Streams instance. This configuration is suitable for learning Streams, developing applications, and possibly running lower-volume production applications. This topology is shown in Figure 5-2.

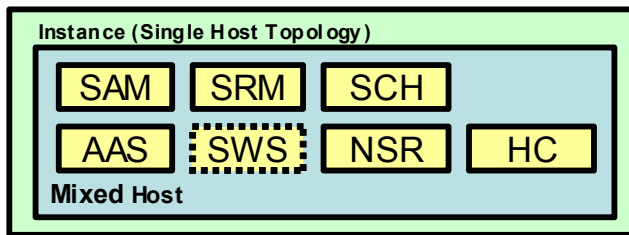


Figure 5-2 Single Host Topology

In a Single Host Topology, the host runs the management services (SAM, SRM, SCH, AAS, NSR, and, optionally, SWS) and the application host services (HC and PEs when an application is run). This type of host is referred to as a *mixed host*.

The configuration of a Single Host Topology is simplified because there are fewer cluster-related setup and configuration steps that are required when the operating system and environment are prepared.

This topology is well-suited for single-node VMWare Player/Workstation/Fusion installations.

## Multi-Host Reserved Topology

The Multi-Host Reserved Topology is the most common and recommended topology for integration, test, and production environments. The *reserved* aspect of this topology refers to the usage of a single host for running the instance management services. This type of host is referred to as a Management Host. The Multi-Host Reserved Topology is shown in Figure 5-3.

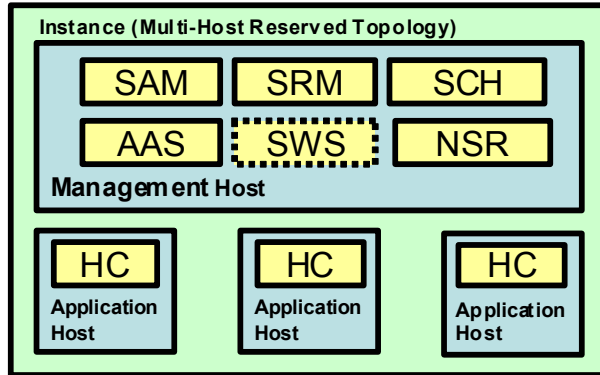


Figure 5-3 Multi-Host Reserved Topology

The additional hosts in this topology are configured to run the application host services, and are referred to as *application hosts*. See 5.5.2 for more details.

The advantage of this topology is the isolation between the management hosts and the applications hosts. Application hosts feature dedicated resources for Streams application execution. In addition, a poorly written Streams application is not as likely to impact the stability of the entire instance. The management services are co-located on a single machine, and because all of the management services are required for the instance to be operational, it is considered a best practice to have a single point of failure, rather than multiple single points of failure.

The management services can be configured with a recovery database that allows them to be restarted on another host if they crash.

In environments with two to three hosts, the Host Controller (HC) service should be added to the management host to maximize resource utilization for applications that require it. This task can be accomplished by running `streamtool addservice`.

## Multi-Host Unreserved Topology

The Multi-Host Unreserved Topology spreads the management and application host services across the nodes. In this configuration, most nodes are considered mixed hosts, although on large clusters there are application hosts because there is a fixed number of management services. A sample Multi-Host Unreserved Topology is shown in Figure 5-4.

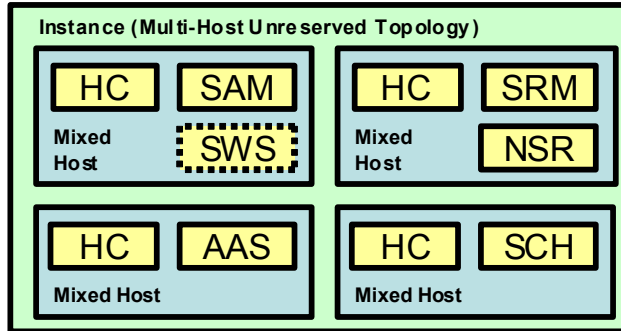


Figure 5-4 Multi-Host Unreserved Topology

This topology is not recommended for most configurations. It is found most commonly in environments where management service recovery is configured and the management services were recovered onto different hosts.

## Other topology options

The Single Host and two multi-host topologies are examples of single instance topologies. If resources are limited, multiple instances can be configured to share resources. A configuration in which two instances are running on a single host is shown in Figure 5-5.

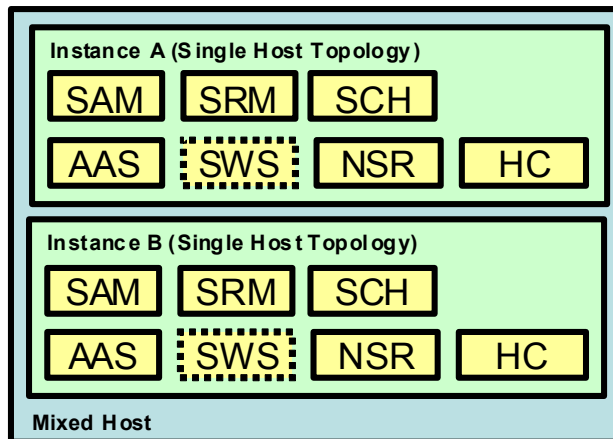


Figure 5-5 Single Host Multiple Instance Topology

Multiple instance topologies are common in development environments. For more complex development environments, multi-host instances can be configured, which share physical resources.

## 5.2 Streams runtime deployment planning

In this section, we describe the most common installation options you should consider when you are planning your InfoSphere Streams deployment. For more information about the installation and configuration options, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888. For more information about installation steps, see Appendix A, “Installing InfoSphere Streams” on page 249.

The following important deployment options must be identified and considered:

- ▶ Streams Software Installation:
  - Operating system version, configuration, and required packages
  - Streams software owner
  - Software installation directory (Streams and Eclipse)
  - Hosts to be configured for InfoSphere Streams usage
- ▶ Streams Instance Configuration:
  - Streams administrator user (for shared instances)
  - Streams administrator home directory (Instance shared space)
  - Instance ID
  - Instance Administrative Group
  - Instance Users Group
  - Hosts assigned to the instance
  - Security Authentication Type
  - Instance Security Key Directory

### 5.2.1 Streams environment

There is a distinction between a Streams environment and Streams instances. A Streams environment is a set of one or more hosts that can be part of an instance. However, at any time, the hosts might or might not be part of an instance. For example, you could install Streams on 10 hosts and ensure that all of the prerequisite and post-installation steps were performed. You now have a 10-host Streams environment.

The hosts in a Streams environment can be dedicated to a single instance. However, in a development environment, they are often divided between a shared instance and hosts that are available to developers for private instances.



## 5.2.2 Sizing the environment

Part of designing a Streams system is sizing the hardware to determine the type and quantity of computers that are needed to support the envisaged applications with the necessary throughput and speed.

Sizing the hardware for an undeveloped application is a difficult task. You must determine the amount of hardware that is required to run the application, and which steps of the application are the most processor, network, and memory-intensive. To complete these tasks, you must consider the following criteria:

- ▶ The volumes of data that are flowing in from the data sources, which includes average data rates and peak data rates.
- ▶ Estimated data reduction at various stages of processing. Wherever possible, you should reduce the data volumes by filtering, with which you can process only the data items that are relevant to the application.
- ▶ Complexity of the Streams operators. It is useful to assess, when possible, the processing requirement of the operators and processing flows within Streams. These requirements can be evaluated by prototyping and measuring the workload and resource usage of the prototype or by comparing it with Streams applications that have similar complexity.
- ▶ Model the total workload. You should scale the measured or estimated processing requirements by the projected workload volumes to give a projection of the processing requirement; that is, how many compute hosts are required to support the envisaged workload.

There are the following trade-offs whether you size a system to cope with average load, peak load, or somewhere in between:

- ▶ Sizing to the peak volumes means that you can achieve your throughput and latency targets more frequently, but the machines are under used at times.
- ▶ Sizing to the average throughput rate results in less hardware than a peak volume sizing, which results in an overall lower-cost system. However, the latency of end-to-end tuple processing is likely to increase during times of above-average traffic, which is undesirable in low latency focused systems.

## 5.2.3 Deployment and installation checklists

Before a Streams deployment is performed, there are two types of checklists that should be created. In Table 5-1 on page 74, we provide an example of such a Streams environment checklist. This checklist provides the information that is necessary when the Streams environment is installed and configured, but does not specify the options for any instances that are defined in the environment. Table 5-2 on page 76 gives an example of a Streams instance configuration checklist. You should create one of these checklists for each shared instance that is created in your Streams environment.

This section describes the options and values that are provided in these checklists.

**Other configuration parameters:** There are many other configuration parameters for Streams environments and instances beyond the options and values that are shown in this chapter. For more information, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888.

## Streams environment checklist

This section describes the options and values that you should organize before a Streams environment installation is performed, as shown in Table 5-1.

Table 5-1 Sample Streams environment checklist

Option	Sample value
Streams version (not configurable)	3.0.0.0
Streams Software Owner	streamsadmin
Streams Software Group	streams
Streams Admin Home Directory (must be shared by all nodes)	/home/streamsadmin
Streams installation location (\$STREAMS_INSTALL)	/opt/streams/InfoSphereStreams
Eclipse installation location	/opt/streams/eclipse
Streams Installation Hosts	redbook1 - redbook14
Streams Available Hosts	redbook8 - redbook14

The options in this table are explained in the following sections.

### **Streams version**

The options that are discussed in this book are based on Streams Version 3.0. Streams versioning is composed of {version}.{release}.{modification}.{fix}. Major interfaces and configuration options as described in here should not change with modification or fix changes.

### **Streams Software Owner**

The Streams Software Owner is the user account that owns the Streams software after installation. You should install the software as the root user before the installation is started. You supply the Streams Software Owner account as part of the installation process (interactive, console, or silent).

### **Streams Software Group**

The Streams Software Group is the group that is assigned to the Streams software after installation. You must supply the group before the installation process.

### **Streams Admin Home Directory**

Your Linux administrator configures the Streams Administrator Home Directory. It is important to know this location because it contains a `~/streams` directory. The `~/streams` directory contains the Streams instance configuration files. If the same user account is used for the Streams administrator and Streams software owner, then the `~/streams` directory also contains the Streams global configuration files. This home directory must be in a shared file system, and it should be backed up regularly.

### **Streams installation location**

When you are installing Streams, you can set the location of the installed Streams files. During the installation, you are prompted to specify the location. If you are performing the installation as the root user (which is the suggested approach), the default location is `/opt/ibm/InfoSphereStreams`. If you do not have root authority, the default location is `<your home directory>/InfoSphereStreams`.

A best practice is to override the default location and select a directory that allows you to co-locate the Eclipse installation for Streams Studio, InfoSphere Streams, and any other toolkits you install (a sample hierarchy is shown in Figure 5-6). A symbolic link is used to hide the version number. This action facilitates installing future versions that can be switched between each other by changing the symbolic link. All references should be made through the link (for example, `/opt/streams/eclipse`).

```
/opt/streams_3.0.0.0/  
InfoSphereStreams/      - Streams Installation Location  
eclipse/                - Eclipse Installation Location  
scripts/                - Your own scripts in this location  
toolkits/               - Additional Toolkits Location  
/opt/streams -> /opt/streams_3.0.0.0 - Symbolic link
```

Figure 5-6 Sample Streams deployment directory structure

### **Eclipse installation location**

InfoSphere Streams 3.0 is now ships with a fully packaged Streams Studio. The user can install and start the studio via the First Steps application. The user also can use an extract application to install the Streams Studio, then start the application by browsing to the `streamsstudio` executable file.

### **Streams Installation Hosts**

The Streams Installation Hosts are all of the hosts that are part of your Streams environment. Each of these hosts features InfoSphere Streams installed and configured to be available as part of shared and private instances.

**Important:** You can install Streams to NFS so that the installation is available to all hosts. In that case, the instance owner's home directory must be in the shared file system (NFS).

### **Streams Available Hosts**

The Streams Available Hosts are the hosts that you make available to Streams users for developing applications and creating their own (that is, private) instances. Hosts that you dedicate exclusively to a shared instance should not be contained in this list because they are specified when you define the shared instance.

## Streams instance configuration checklist

This section describes the options and values that you should organize in preparation for the creation of a shared Streams Instance. Table 5-2 shows a sample instance configuration checklist.

Table 5-2 Sample instance configuration checklist

Option	Sample value
Streams Admin User (Usually the same as the software owner)	streamsadmin
Streams Admin Home Directory (must be shared by all nodes)	home/streamsadmin
Streams Instance ID	streams@streamsadmin
Streams Admin Group	streamsadmin
Streams User Group	streams
Shared Instance Primary (Management) Host/Server	redbook1
Shared Instance Secondary (Application) Hosts/Servers	redbook2 - redbook7
Instance Shared Space	home/streamsadmin/.streams
User Shared Space <sup>1</sup>	shared/data
Streams Name Service URL	DN: - Distributed Name Server (default)
Security Authentication Type	PAM (default)
Pam Enable Key	True (default)
Security Public Key Directory	home/streamsadmin/streams_keys

1. A Linux directory that multiple users can access.

### Streams Admin User

The Streams Admin User (also known as the instance owner) is the user account that creates and owns your Streams instance. This user account should be the same as the Streams Software Owner. This user account can create multiple instances in a multi-instance topology.

You are not prompted to specify this user account during the installation process, but you use this account to create your instance.

### Streams Admin Home Directory

Your Linux administrator most likely configures the Streams Admin Home Directory. It is important to know this location because it contains a `~/streams` directory. The `~/streams` directory contains the Streams instance configuration files. If the same user account is used for the Streams Admin and Streams Software Owner, the `~/streams` directory also contains the Streams global configuration files. This directory should be backed up regularly.

## Streams instance ID

The Streams instance ID is described in “Streams instance IDs” on page 67. This deployment option refers to a shared instance that is created by the Streams Admin for use by multiple users.

Your deployment can have multiple instances. Some instances are single-user developer instances that are somewhat transient, but likely you have one or more shared instances.

Instances are not created as part of the installation. You create them after you install the software.

## Streams Admin Group

This option is the name of the administration group to use during instance creation to initialize the access control list for shared instances. This group must exist in the authentication database before the shared instance is created. After the instance is created, the ACLs can be managed via the Streams Console.

Users in the administration group have permissions to manage all configuration parameters and streams jobs in the instance. As an example, users in the admin group can cancel jobs that are submitted by any user.

## Streams User Group

This option is the name of the users group to use during instance creation to initialize the access control list for shared instances. This group must exist in the authentication database before the shared instance is created.

Users in the user group have permissions to start, stop, and manage their own jobs in the Streams instance.

## Primary (Management) Host/Server

The Primary host represents the server that hosts the management services in a Multi-host Reserved Topology. If you are installing a single host instance, this host is the only host that your instance uses. In the case of a Multi-host Unreserved Topology, identifying a single node from where you run commands is a best practice. For a definition of these terms, see 5.1.3, “Deployment topologies” on page 69.

## Secondary (Application) Hosts/Servers

The Secondary hosts are the other hosts that are configured to be part of your Streams environment. Some of these hosts might be reserved for private developer instances and are not part of your shared instance. In these cases, it is still important to identify these hosts because they do require Streams to be installed. For a definition of these terms, see 5.1.3, “Deployment topologies” on page 69.

## Instance Shared Space

The Instance Shared Space is the location where Streams Instance configuration files are located. In the current release of Streams, this location is within the `~/streams` directory of the Streams instance owner.

## User Shared Space

The User Shared Space is file system storage that is shared across all hosts in a Streams instance. This space is usually in addition to the users home directories that are shared across the hosts.

The primary purpose of this space is for application and toolkit development. Permissions are often opened up to include a development group for collaboration and sharing.

NFS is not the best choice for this space because of latency issues that were observed with NFS in a distributed computing environment. The IBM General Parallel File System (GPFS) is a low-latency, high performance, and highly distributed file system that works well for this purpose. It can use extra local disks on the cluster hosts and SAN-attached storage. GPFS is licensed and distributed separately from InfoSphere Streams.

## Streams Name Service URL

This option is the URL for the name service the instance uses. This option provides a reference to a distributed or file system-based name service. The default is a distributed name service.

Specify DN: to use the distributed name service. This value is the best value to use if you are not configuring your instance for management service recovery. It is the default value if you are not creating an instance with recovery mode.

Specify a value of FS: to use the file system based name service. This value is the default if the instance also has recovery mode that is set on. A default file path is used unless a path is specified following the FS: value. The directory path must not contain white space characters. If you are configuring management service recovery, you must specify the file system-based name service.

## Security Authentication Type

The run time's authentication processing integrates with the customer's existing user authentication infrastructure. Integration with Pluggable Authentication Modules (PAM) for Linux and Lightweight Directory Access Protocol (LDAP) back ends are supported. All user repository management occurs outside of InfoSphere Streams, by using directly the back-end infrastructure's toolset.

The instance's authentication configuration is established when the instance is created. By default, the PAM back end is used. The PAM authentication defaults to a PAM service that is named *login*. For more information about customizing the name of the default PAM service for your installation, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888, or the IBM InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

The integration with PAM also enables use of a public or private key mechanism, which is created by running the **streamtool genkey** command. This mechanism enables the construction of a secure environment where the user is not prompted to provide their password when administrative tasks are performed, such as setting ACLs or submitting jobs. When PAM is used for streamtool commands, to avoid password prompting for shared instances, the public key for users that are authorized to an instance should be copied to the directory that is identified by the SecurityPublicKeyDirectory instance configuration property. To disable the use of keys, specify '--property PamEnableKey=false' with the **streamtool mkinstance** command.

PAM is the suggested authentication mechanism.

A security template that uses LDAP can be used if the installation owner enabled LDAP. The `--template ldap` option can be used to specify LDAP authentication if it was enabled. For more information about configuring the use of LDAP for your installation, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888 or the IBM Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

For more information about Security considerations in InfoSphere Streams 3.0, see Appendix B, “IBM InfoSphere Streams security considerations” on page 277.

## Security Public Key Directory

The Security Public Key Directory property is the PAM-specific property for Streams instances. Enabling RSA authentication for Streams users involves managing the user's public keys and storing them in the directory path that is specified in this property.

**Important:** It is important that access to the RSA public keys directory is restricted and that you ensure the secure management of the public keys. For more information, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888

The RSA public keys use the following default directory locations:

- ▶ Private instance:  
    `~instance_owner/.streams/key`
- ▶ Shared instance:  
    `~instance_owner/.streams/instances/instance_name/config/keys`

In environments that support multiple shared instances with a single instance owner (for example, Instance Admin), it is common to create a single keys directory in the Instance Admin's home directory and set the instance configuration parameter (`SecurityPublicKeyDirectory`) to point to this location when the instance is created. This action allows multiple instances to use the same set of keys and centralize the management of these keys. Access control can still be maintained at the instance level to limit users abilities on each instance.

## 5.3 Streams instance creation and configuration

In this section, we describe the steps that are used to create, start, and verify a shared Streams instance after Streams is installed. For more information about the installation process, see Appendix A, “Installing InfoSphere Streams” on page 249.

In addition, we describe how developers can create their own private development instances by using the set of hosts in their environment that are not reserved for shared instances. The examples in this section are based on the sample deployment and installation checklists that are described in Table 5-1 on page 74 and Table 5-2 on page 76.

### 5.3.1 Streams shared instance configuration

A Streams shared instance can be created by using any of the topologies that are described in 5.1.3, “Deployment topologies” on page 69. In this example, we define a shared instance that uses a multi-host reserved topology.

The commands in this section should be performed as the instance administrator (often the same account as the Streams Software Owner account). In our example, this account is the `streamsadmin` account.

#### The management host, application hosts, and boot logs

In a multi-host reserved topology, one host is identified as the management host and runs all of the management services of the instance. The commands to configure and interact with a Streams instance can be issued from any of the hosts in the Streams environment (including those hosts that are not part of the instance).

You should issue commands and configure the management host because it has the most information about the status of starting the instance. A boot log is created on each host in the instance and is in `/tmp/<instance name>@<instance owner>.boot.log` (for example, `/tmp/streams@streamsadmin.boot.log`). The boot log on each host contains the boot (or startup) log for the services that are running on that host. The boot log on the management host, therefore, has the boot information for each of the management services and it is most accessible to you if you are issuing the instance configuration and start commands on that host. Issuing the `streamtool getlog --services boot -i instancename` command also retrieves the bootlogs from all the nodes that are configured in your instance.

In a multi-host reserved topology, there are one or more application hosts. When you start an instance, the application hosts has boot logs in `/tmp` with the same name as the boot log on the management host. The boot logs on application hosts show only the log for starting the host controller service.

#### Specifying shared instance hosts

The following common ways are used to define the specific hosts that are part of your shared Streams instance that use the `streamtool mkinstance` command:

- ▶ Use the `--hfile` option to specify a file that contains the host names of the hosts to be included in the instance. The first host that is listed is the management host.
- ▶ Use the `--hosts` option to specify a comma-separated list of host names. The first host that is listed is the management host.



**Other options:** There are many other options for host selection that are described in *IBM InfoSphere Streams: Installation and Administration Guide V3.0, SC19-3888*. However, we are presenting the most common for shared instances, which use a reserved topology.

We suggest the use of the first option, which simplifies the `streamtool mkinstance` command while providing a single place to make changes if you must remove and re-create the instance. We created the host names file that is shown in Example 5-1 for our shared instance.

*Example 5-1 Sample mkinstance host names file for use with the -hfile option*

---

```
# /home/streamsadmin/streams@streamsadmin_hosts
# List of hosts for shared instance: streams@streamsadmin
# First host listed will be management host
redbook1
redbook2
redbook3
redbook4
redbook5
redbook6
redbook7
```

---

### **Instance host tags**

In addition to specifying the hosts that are used in the instance, you can apply host tags to any or all of the hosts in the instance. The tags are user-defined labels that can be used to identify capabilities, configurations, or any other differentiation feature of the hosts. These tags can be used during application design and deployment to control how processing elements that contain the operators are distributed and placed on hosts. For more information about user-defined tags and host placement options, see 5.4.2, “Operator host placement” on page 91. Example 5-2 shows a host file with tags applied to some of the hosts.

*Example 5-2 Sample mkinstance host names file (to use with -hfile) with host tags*

---

```
# /home/streamsadmin/streams@streamsadmin_hosts
# List of hosts for shared instance: streams@streamsadmin
# First host listed will be management host
redbook1
redbook2 --tags ingest,fpga
redbook3 --tags ingest,fpgs
redbook4 --tags dbclient
redbook5
redbook6 --tags sanstorage
redbook7 --tags sanstorage
```

---

## Creating and populating the security public key directory

The run time's authentication processing integrates with the customer's existing user authentication infrastructure. Integration with PAM and LDAP back ends are supported. In our example, the product uses PAM authentication because it is easiest to configure and maintain. All user repository management occurs outside of InfoSphere Streams directly by using the back-end infrastructure's toolset.

The instance's authentication configuration is established when the instance is created. By default, the PAM back end is used. The PAM authentication default uses a PAM service named `login`. For more information about customizing the name of the default PAM service for your installation, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888 and the IBM InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

Another PAM integration default is configured to provide for the use of a public or private key mechanism to enable the construction of a secure environment where the user is not prompted to provide their password. To avoid password prompting for shared instances, the public key for users (authorized to an instance) should be copied to the directory that is identified by the `SecurityPublicKeyDirectory` instance configuration property.

The following commands are used to create a public key directory in the stream administrators home directory:

- ▶ `$ mkdir /home/streamsadmin/streams@streamsadmin_keys`
- ▶ `$ chmod 700 /home/streamsadmin/streams@streamsadmin_keys`

Previously in this chapter, we showed you how to create a public key and private key for each Streams user by using the `streamtool genkey` command. You now must copy the public key for each user that is authorized to use the shared instance into the public key directory you created. The streams instance administrator key must also be copied into the key directory. The commands that are required are shown in the following example:

```
$ cp /home/streamsadmin/.streams/key/streamsadmin.pem  
/home/streamsadmin/streams\@streamsadmin_keys
```

**Important:** Copy the key for every user that is authorized to use this shared instance into the directory. Keys can be added at any time, even after the instance was created and started.

## Creating and starting the shared instance

After the instance hosts file is defined and the security public key directory is created, it is time to create the shared instance.

**Important:** Any instance can be shared. The template approach is one way to share an instance, but you can also set permissions for more users to a private instance, thus making it a shared instance.

The `streamtool mkinstance` command is used to create Streams instances. For more information about all of the options, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888 and the output from the `streamtool man mkinstance` command.

The following command creates the shared instance for our example:

```
$ streamtool mkinstance -i streams@streamsadmin --hfile
/home/streamsadmin/streams@streamsadmin_hosts --template shared --property
SecurityPublicKeyDirectory=/home/streamsadmin/streams@streamsadmin_keys --property
AdminGroup=streamsadmin --property UsersGroup=streams
CDISC0040I Creating Streams instance 'streams@streamsadmin'...
CDISC0001I The Streams instance 'streams@streamsadmin' was created.
```

**Important:** Save this command in a script file for reuse and documentation.

After the instance is created, you can start the instance by using the following command:

```
$ streamtool startinstance -i streams@streamsadmin
CDISC0059I Starting up Streams instance 'streams@streamsadmin'...
CDISC0078I Total instance hosts: 7
CDISC0056I Starting a private Distributed Name Server (1 partitions, 1
replications) on host 'redbook1'...
CDISC0057I The Streams instance's name service URL is set to
DN:redbook1.mydomain.net:34875.
CDISC0061I Starting the runtime on 1 management hosts in parallel...
CDISC0060I Starting the runtime on 6 application hosts in parallel...
CDISC0003I The Streams instance 'streams@streamsadmin' was started.
```

## Verifying the instance

After your instance is started, the following **streamtool** commands can be used to manage and configure your instance:

- ▶ **streamtool (get/set/rm)property**
- ▶ **streamtool (ls/add/rm)host**
- ▶ **streamtool (add/rm)service**

For a complete list of **streamtool** commands, run **streamtool man** command.

The **streamtool getresourcestate** command is the best way to get a quick snapshot of the health of the services of an instance. In Example 5-3, we show the results of this command immediately after the instance is started successfully.

### Example 5-3 Command results

---

```
$ streamtool getresourcestate -i streams@streamsadmin1
Instance: streams@streamsadmin Started: yes State: RUNNING Hosts: 7 (1/1
Management, 6/6 Application)
Host          Status  Schedulable  Services                               Tags
redbook1     RUNNING -           RUNNING:aas,nsr,sam,sch,srm,sws
redbook1     RUNNING yes         RUNNING:hc
redbook2     RUNNING yes         RUNNING:hc
redbook3     RUNNING yes         RUNNING:hc
redbook4     RUNNING yes         RUNNING:hc
redbook4     RUNNING yes         RUNNING:hc
redbook4     RUNNING yes         RUNNING:hc
redbook5     RUNNING yes         RUNNING:hc
redbook6     RUNNING yes         RUNNING:hc
```

---

<sup>1</sup> When an instance name (with -i) is specified, the user ID (@ plus streamsadmin) can be omitted.

## 5.3.2 Streams private developer instance configuration

In addition to interacting with a shared instance, your developers want to perform development tasks by using their own private instance. This approach provides isolation from other developers. However, the overlap of development instances on hosts can lead to resource constraints. Therefore, private developer instances should be used for functional development and testing of Streams applications. However, performance testing should be performed on a shared instance with exclusively dedicated hosts.

Development instances are simplified versions of shared instances for use by a single user. They can use one or more hosts from the set of available hosts that are configured in your Streams environment.

The developer instance configuration template simplifies instance creation and provides many default values and properties. The developer template includes the following most important aspects:

- ▶ It creates a private instance by using the creator's private key to authenticate access.
- ▶ The SWS HTTPS port is randomly chosen from the ports that are available when the instance is started. To find the URL, use `streamtool geturl -i <instance_name>` command.

**Manually specify hosts:** It is possible for a user to manually specify hosts that are used by shared instances. Users should be encouraged to use the `--numhosts` option for creating instances.

The commands that are described in this section can be issued by any Streams user account that is configured as described in “Streams User Group” on page 77.

### **STREAMS\_DEFAULT\_IID environment variable**

By using the `streamtool` commands, the user can specify the instance to interact with by using the `-i` or `--instance-id` flags. If a user does not use this option, the `streamtool` command uses the value of the `STREAMS_DEFAULT_IID` environment variable to determine the instance to target with the `streamtool` command. In an environment where a shared instance is set up as the default instance ID, users should override the `STREAMS_DEFAULT_IID` if they are going to work with a private developer's instance rather than the shared instance. This action reduces confusion if the instance option is accidentally omitted when the `streamtool` commands are used.

## Creating a single host private developer instance

The `streamtool mkinstance` default for host selection is `--numhosts 1`. This option selects the first host from the available hosts that are configured for the Streams environment.

The following commands create and start a single host private developer instance on one of the available hosts that are configured in the Streams environment:

```
$ streamtool mkinstance -i streams@devuser --template developer
CDISC0040I Creating Streams instance 'streams@devuser...'
CDISC0001I The Streams instance 'streams@devuser' was created.
$ streamtool startinstance -i streams@devuser
CDISC0059I Starting up Streams instance 'streams@devuser...'
CDISC0078I Total instance hosts: 1
CDISC0056I Starting a private Distributed Name Server (1 partitions, 1
replications) on host 'redbook8.mydomain.net'...
CDISC0057I The Streams instance's name service URL is set to
DN:redbook8.mydomain.net:48643.
CDISC0061I Starting the runtime on 1 management hosts in parallel...
CDISC0003I The Streams instance 'streams@devuser' was started.
```

If you need to specify the exact host on which to create the instance, you can use the `--hosts` parameter to specify the host by name. The following commands create and start a single host private developer's instance on host named `redbook14`:

```
$ streamtool mkinstance -i streams@devuser --template developer --hosts redbook14
CDISC0040I Creating Streams instance 'streams@devuser...'
CDISC0001I The Streams instance 'streams@devuser' was created.
$ streamtool startinstance -i streams@devuser
CDISC0059I Starting up Streams instance 'streams@devuser'...
CDISC0078I Total instance hosts: 1
CDISC0056I Starting a private Distributed Name Server (1 partitions, 1
replications) on host 'redbook14'...
CDISC0057I The Streams instance's name service URL is set to
DN:redbook14.mydomain.net:41656.
CDISC0061I Starting the runtime on 1 management hosts in parallel...
CDISC0003I The Streams instance 'streams@devuser' was started.
```

## Creating a multi-host private developer instance

In addition to single host instances, developers can create multi-host private instances. The `--numhosts` option allows the user to specify the number of hosts to be included in the instance. If this number is greater than 1, the instance defaults to the use of a single reserved node for the management host, and the other hosts are application hosts. To maximize the number of application hosts, you can use the `--unreserved` option, which places the HC service on all nodes.

The following example shows the creation of a multi-host private developer instance on four available hosts and configures them all to be used for running Streams applications. In addition, it shows the `streamtool lshosts` command that displays the host layout of the new instance:

```
$ streamtool mkinstance -i streams@devuser --template developer --numhosts 4
--unreserved
CDISC0040I Creating Streams instance 'streams@devuser...'
CDISC0001I The Streams instance 'streams@devuser' was created.
$ streamtool lshosts -l -i streamsInstance: streams@devuser
Host                Services                Tags
redbook8            hc,aas,nsr,sam,sch,srm,sws
redbook9            hc
redbook10           hc
redbook11           hc
```

**Important:** If you specify a value for the `--numhosts` option that is greater than the number of available hosts, you receive an error from the `streamtool mkinstance` command.

### Creating a private developer instance by using Streams Studio

In addition to the command-line interface, you can create instances from within Streams Studio. It is a best practice that shared instances be created by using the command-line interface and that the scripts to create them are preserved. However, developer instances are good candidates for creation from within Streams Studio.

In Figure 5-7, we select **Make Instance...** from the Instances folder within the Streams Explorer pane.

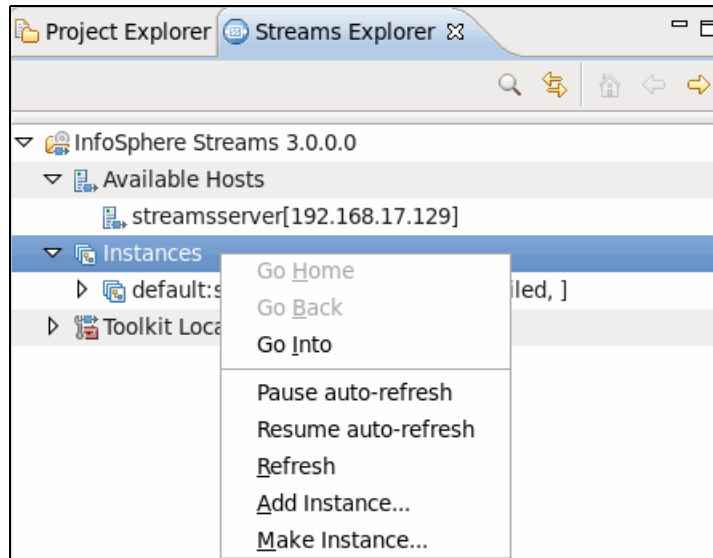


Figure 5-7 Streams Studio Make Instance menu

Figure 5-8 shows the Make Instance window. It is important to select the **Use template:** option and **Browse** to select the developer template (developer.template.properties).

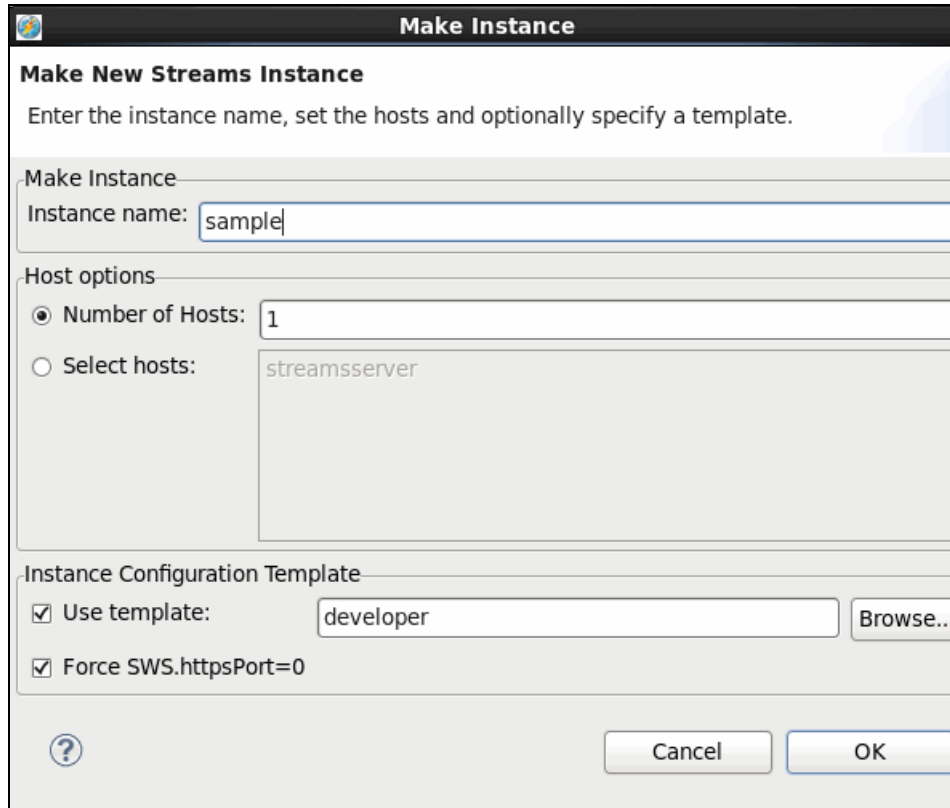


Figure 5-8 Streams Studio Make Instance window

After the instance name is entered, number of hosts, and selecting the template, click **OK**. To start the instance, right-click the new instance within the Streams Explorer pane and select **Start Instance**. You can use the instance from both Streams Studio and the command line.

Streams V3.0 introduced a graphical interface instance management UI available from the CLI: `st launch --config` (or `st launch --firststeps`). For more information about this UI, see 8.1, “InfoSphere Streams Instance Management” on page 200.

## 5.4 Application deployment capabilities

In the first half of this chapter, we discussed Streams runtime deployment. We now focus on Streams application design and deployment. After the topology, size, and layout of the runtime environment that is required for the projected applications (including their required volumes) is determined, you can determine how the processing should be divided across multiple hosts.

Application deployment addresses the issue of how the components of the application are distributed and run on the physical hosts. The application can be segmented and grouped to optimize throughput, latency, and scalability in one of the following ways:

- ▶ At development time
  - An application can be segmented into multiple applications. Then the data that is flowing through different streams within Streams can be exported from one application and imported into another application.
  - You can choose to deploy multiple copies of an operator, each working on a subset of the total stream. This configuration allows parallel processing when the operator is processor-intensive and cannot satisfy the throughput requirements on a single processor or host.
  - You can choose to segment processor-intensive analysis into multiple stages and then process the stream one stage after the other. This action allows pipelined processing and segmenting or distributing the processing load of the operators.
  - You can choose to control operator placement on the hosts of your instance. You can control operators that should be colocated, other operators that should not be colocated, and what set of hosts an operator might be allowed to run.
  - You can choose to manually combine multiple operators into a single Processing Element, called *fusing*. When operators are fused into PEs, they communicate through function calls and tuples are passed from one operator to the next by using memory references, as opposed to the use of the transport (which might involve a transmission of the communication to take place across a network). Fusing can significantly reduce the cost of communication and improve latency and throughput.
- ▶ At compile time

Profile-driven fusion optimization is an approach in which the compiler figures out how to best fuse operators into one or more PEs while the user-specified constraints are respected. This process is called *fusion optimization* and requires a profiling step where the application is first run in a profiling mode to characterize the resource usage concerning CPU consumption and data traffic between each of the operators that make up the application.
- ▶ At run time
  - The SAM accepts requests to run applications. The SAM starts the Scheduler service to determine on which hosts the various PEs should run. The Scheduler then retrieves host information and performance metrics from the Streams Resource Manager, which enables the best placement of PEs to be determined dynamically at run time.
  - The PEs are deployed to the hosts as determined by the scheduler. The HC on the hosts creates and manages a PEC to run each PE.

The rest of this section describes several of the application design and deployment capabilities that are available in InfoSphere Streams.



## 5.4.1 Dynamic application composition

By using the Streams run time, you can run multiple applications in a Streams instance and export a stream from one application and import the stream into other applications. In Figure 5-9, we show where the imagefeeder application exports a stream of .jpeg images to the imagecluster, greyscale, and imagewriter applications. This example also shows Streams data that is flowing from the imagecluster and greyscale applications to the imagewriter application.

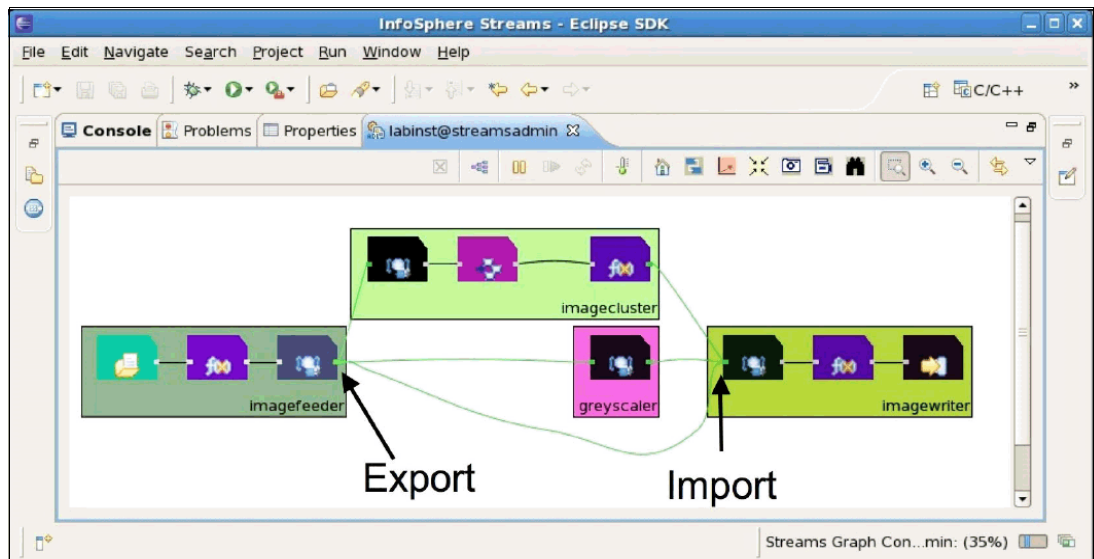


Figure 5-9 Application composition through Export and Import operators

The exporting and importing of Streams between applications allows Streams administrators to use applications together to form larger integrated applications. One advantage of this approach is the ability to start and stop the individual applications independently without affecting the others. For example, you could choose to segment the Streams applications into the following categories:

- ▶ Applications to read, validate, and filter individual source streams.
- ▶ Applications to analyze a number of the source streams.
- ▶ Applications to format and deliver output data to sinks.

This segmentation allows Streams administrators to stop and start the source applications when the availability of Streams hosts varies. For example, you can run multiple analysis applications and develop and deploy new analysis algorithms without impacting the existing algorithms. You can also independently start and stop the sink applications to support new downstream applications to relieve the load on the system when downstream applications are inactive.

When you are exporting a stream from an application, you use the Export operator. When you are importing a stream into an application, you use the Import operator and then define which streams are imported.

Exported streams and imported streams can be matched or connected in the following ways:

► Property-based application composition

This method uses a combination of stream properties that are set on the Export operator along with subscriptions that are specified on the Import operators. The middleware connects the imported and exported streams if the subscription predicate matches the exported stream properties and the stream schemas are compatible. If the Import predicate matches multiple streams that are exported by jobs that are running in the middleware, they are all connected. If there are no matching streams, nothing arrives at the Import operator. Properties and subscriptions can be added, updated, or removed at run time.

Property-based application composition is shown in Example 5-4.

*Example 5-4 Property-based composition*

---

```
// imagefeeder application (partial listing)
composite imagefeeder {
  graph
  ...
  stream<IplImage image,
    rstring filename,
    rstring directory> Img = ImageSource(Files) {...}

  () as ImagesExport = Export(Img) {
    param
      properties : { dataType = "IplImage",
                    writeImage = "true"};
  }
}
-----
// imagewriter application (partial listing)
composite imagewriter {
  graph
  ...
  stream<IplImage image,
    rstring filename,
    rstring directory> ImagesIn = Import() {
    param
      subscription : dataType == "IplImage" &&
                    writeImage == "true";
  }
  ...
}
```

---

► Application name and stream ID composition

This method requires the importing application to specify the application name and a stream ID of the Export stream. The exporting application uses the `streamId` property of the Export operator to identify the stream. The importing application uses the `applicationName` and `streamID` parameters of the Import operator to identify the specific stream that is imported. If the exporting application is defined within a namespace, the importing application must fully qualify the application name by using the namespace and the application name. This approach to job integration is less flexible than property-based composition.

Application name and stream ID composition is shown in Example 5-5.

*Example 5-5 Application name and stream ID composition*

---

```
// SensorIngest application (partial listing)
namespace projectA.ingest;
composite SensorIngest {
  graph
  ...
  stream<uint32 sensorId,
    timestamp sensorTime,
    blob sensorData> SensorData = Functor(DataIn){...}

  () as SensorExport = Export(SensorData) {
    param
    streamId: "SensorIngestData";
  }
}

-----

// SensorProcess Application (partial listing)
namespace projectA.processing;
composite SensorProcess {
  graph
  stream<uint32 sensorId,
    timestamp sensorTime,
    blob sensorData> SensorData = Import(){
  param
  streamId : "SensorIngestData" ;
  applicationName : "projectA.ingest::SensorIngest";
  }
  ...
}
```

---

**Exporting a stream:** If you export a stream from one application and it is not imported elsewhere, the tuples in this streams are lost. They are not automatically persisted or buffered.

When you start an importing application, the contents of the stream are tapped from that time onwards. Likewise, if an importing application is stopped for a time and restarted, then the contents of the stream are lost during the time the application is stopped.

## 5.4.2 Operator host placement

The Streams compiler and Scheduler can optimize your application by spreading the various operators (which are run as processing elements known as PEs) across different hosts. By default, the Scheduler attempts to schedule hosts that are based on their current load.

In addition to automatic host placement, you can manually allocate operators to specific hosts. You can also control which operators are placed on the same host, different hosts, and specify which operators should be isolated and run on a dedicated host.

This section describes the steps and artifacts that are involved in controlling operator host placement. For more information, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

## Host tags

In , “Instance host tags” on page 81, we described host tags, which included the two predefined host tags: build and execution. In addition to the predefined tags, you can create your own tags and assign them to the hosts in your instance.

The reason for defining host tags is to classify groups of hosts as having particular characteristics that a developer operator might need without specifying a specific host and thus restricting portability. Although host tags are not required, they are a good idea if you are going to take manual control over operator placement.

In addition to assigning tags to hosts at instance creation, you can create host tags and assign them to hosts after the instance is created, even while the instance is running. After an instance is created, you use the `streamtool mkhosttag` command to define host tags. After tags are defined, they can be assigned to hosts by using the `streamtool chost` command. Tags can be assigned to multiple hosts and a host can be assigned multiple tags.

An example of defining, assigning, and displaying the host tags on a running instance is shown in Example 5-6.

### Example 5-6 Host tags

---

```
$ export STREAMS_DEFAULT_IID=streams@streamsadmin
$ streamtool mkhosttag --description "Ingest Nodes" ingest
CDISC0102I Host tag 'ingest' has been added to instance 'streams@streamsadmin'.
$ streamtool mkhosttag --description "32GB RAM" himem
CDISC0102I Host tag 'himem' has been added to instance 'streamtool@streamsadmin'.
$ streamtool chost --add --tags ingest,himem redbook2
CDISC0094I Adding host tags 'ingest,himem' to host 'redbook2' in instance
'streams@streamsadmin'.
$ streamtool chost --add --tags himem redbook3
CDISC0094I Adding host tags 'himem' to host 'redbook3' in instance
'streams@streamsadmin'.
$ streamtool lshosts -l
Instance: streams@streamsadmin
Host      Services          Tags
redbook1  aas,nsr,sam,sch,srm,sws  build,execution
redbook2  hc                  build,execution,himem,ingest
redbook3  hc                  build,execution,himem
redbook4  hc                  build,execution
redbook5  hc                  build,execution
redbook6  hc                  build,execution
redbook7  hc                  build,execution
```

---

## Host pools

Host pools are a collection of hosts that are defined in a Streams application that can be used to support operator host placement. A default pool always is available, which by default consists of the complete set of hosts that are available in the instance and were not allocated to exclusive pools. The size of the default pool can be limited by using the `defaultPoolSize` configuration directive or the `-d` Streams compiler option. These options are defined in the configuration clause on the main composite for the application.

Host pools can be defined as *shared* or *exclusive*. A pool is shared if it can contain hosts that also appear in other pools. A pool is exclusive if it must not contain any hosts that appear in other pools.

The hosts that are included in each host pool can be selected at run time in one of the following ways:

- ▶ **Tagged pool:** Hosts that are included in this type of pool are allocated at run time from a set of hosts that were assigned a specific host tag or set of hosts tags.
- ▶ **Implicit Pool:** Hosts are allocated at run time from the set of hosts that are available in the instance the application that is deployed so that they are not allocated to an Exclusive pool.

Although not recommended, a host pool can include the set of hosts that are assigned to it and specified at compile time. In this case, the pool is defined by specifying the set of hosts by name or IP address at compile time. This configuration is not recommended because it hinders deployment activities by tying the application to specific hosts and requiring a recompile if host assignments must be modified in the future.

Host pools can be defined by providing a size, in which case only the specified number of hosts are selected at run time to be part of the pool. For example, if an instance has four hosts with a host tag of `ingest`, but the application defines a tagged pool based on the `ingest` tag with a size of 2, it includes only two of the four tagged hosts in the host pool. If a size is not specified in this example, all hosts with the tag of `ingest` are included in the pool.

Several host pool examples are shown in Example 5-7.

### *Example 5-7 Host pool examples*

---

```
composite Main {
  graph
    // operator invocations here
  config
    hostPool :
      // Tagged pool, shared, no size specified
      ingestPool=createPool({tags=["ingest"]}, Sys.Shared),
      // Tagged pool, exclusive, no size specified
      dbPool=createPool({tags["dbclient"]}, Sys.Exclusive),
      // Implicit pool, shared, sized
      computePool=createPool({size=2u}, Sys.Shared)
      // Explicit compile-time pool - NOT RECOMMENDED
      badPool = ["redbook2","10.5.5.10"];
}
```

---

## Host placement configuration directives

SPL provides a configuration clause that allows developers to control the behavior and deployment of Streams operators. The placement operator configuration provides several subconfiguration options for controlling operator host placement.

The following groups of host placement subconfigurations are available:

- ▶ **Absolute host location:** This host subconfiguration specifies the absolute location (pool, IP address, or name) of the host on which the operator instance should run.
- ▶ **Relative host constraint:** These subconfigurations constrain whether operator instances must run (`hostColocation`) or must not run (`hostExlocation`) on the same host, or whether they run in a partition that has a host of their own (`hostIsolation`).

### ***placement: host***

After one or more host pools are defined, you can specify whether an operator instance should use a host from one of the host pools. If a runtime-selected host pool is specified in the host configuration, the actual host is chosen at run time. This approach ensures maximum application portability. In addition to specifying a host pool, the host configuration can be used to specify an explicit host name or IP address, but this approach severely limits portability.

Examples of host configuration specifications are shown in Example 5-8.

#### *Example 5-8 Host config specifications*

---

```
composite Main {
  graph
    stream<int32 id> S1 = Beacon() {
      // System picks a host from the pool at runtime
      config placement : host (ingestPool);
    }
    stream<int32 id> S2 = Beacon() {
      // System picks host at a fixed offset from a pool
      config placement : host (ingestPool[1]);
    }
    stream<int32 id> S3 = Beacon() {
      // System picks host by name !!Not Recommended!!
      config placement : host ("redbook1.mydomain.com");
    }
  }
  config
    // Tagged pool, shared, no size specified
    hostPool :
      ingestPool=createPool({tags=["ingest"]}, Sys.Shared);
}
```

---

### ***placement: hostColocation***

By using the `hostColocation` placement configuration, you can specify a colocation tag. All other operator invocations that use the same colocation tag are run on the same host.

### ***placement: hostExlocation***

By using the `hostExlocation` placement configuration, you can specify an exlocation tag. Any other operator invocations that use the same exlocation tag must be run on different hosts.

### **placement: hostIsolation**

By using the `hostIsolation` placement configuration, you can specify that an operator must be run in an operator partition that has a host of its own. Other operators can be run in the same partition (see 5.4.3, “Operator partitioning” on page 96), but no other partition can be run on that host.

**Conflicting configuration statements:** There is potential for someone to erroneously specify inconsistent or conflicting configuration statements, which makes it impossible to resolve them completely. For example, `exlocation` can conflict with `colocation`. In most cases, the compiler can detect such conflicts. When the compiler detects a conflict, it issues an error message. If the compiler cannot detect conflicts, the conflicts are identified when the application is submitted to run on an instance, which results in a `streamtool submitjob` error.

In Example 5-9, we show the use of relative operator placement. In this example, all three operators are placed on hosts from the `ingestPool`. However, there are constraints that require at least two hosts to be available in the pool. The operators that produce streams `S1` and `S2` are collocated, and the operator that produces stream `S3` cannot be on the same host as `S1`.

#### *Example 5-9 Relative operator placement*

---

```
composite Main {
  graph
    stream<int32 id> S1 = Beacon() {
      // System picks a host from the pool at runtime
      // and sets a colocation tag
      config placement : host (ingestPool),
                        hostColocation("withS1"),
                        hostExlocation("notWithS3");
    }
    stream<int32 id> S2 = Beacon() {
      // place this operator on the same host as the S1
      config placement : hostColocation("withS1");
    }
    stream<int32 id> S3 = Beacon() {
      // System picks host by name !!Not Recommended!!
      config placement : hostExlocation("notWithS3");
    }
  config
    // Tagged pool, shared, no size specified
    hostPool :
      ingestPool=createPool({tags=["ingest"]}, Sys.Shared);
}
```

---

### 5.4.3 Operator partitioning

You can fuse multiple operator instances in an SPL application into units called *partitions*. The notion of a partition of operator instances in the flow graph is distinct from the notion of a partition in an SPL window clause; they are unrelated concepts. The execution container for a partition is a PE. At run time, there is a one-to-one correspondence between partitions and PEs because each partition runs in exactly one PE. Fusion is important for performance because PE-internal communication is faster than cross-PE communication. Conversely, multiple PEs can use the hardware resources of multiple hosts. By default, the optimizer does not perform fusion and places each operator instance in its own PE. Furthermore, users can control fusion with explicit partition placement constraints in the code, as shown in Example 5-10.

*Example 5-10 Operator partitioning example*

---

```
int32 foo(rstring s, int32 i) { /*do something expensive*/ return i; } //1
composite Main {
graph
  stream<int32 i> In = Beacon(){}

  stream<int32 i> A1 = Functor(In) {
    output A1      : i = foo("A", i);
    config placement : partitionColocation("A");
    //fuse A1 into "A"
  }
  () as A2 = FileSink(A1) {
    param file      : "A.txt";
    config placement : partitionColocation("A");
    //fuse A2 into "A"
  }
  stream<int32 i> B1 = Functor(In) {
    output B1      : i = foo("B", i);
    config placement : partitionColocation("B");
    //fuse B1 into "B"
  }
  () as B2 = FileSink(B1) {
    param file      : "B.txt";
    config placement : partitionColocation("B");
    //fuse B2 into "B"
  }
}
```

---

### 5.4.4 Parallelizing operators

When you deploy your application on the runtime hardware, you might find that you have a single application operator, which is a performance bottleneck. This means that the operator cannot deliver the necessary throughput or perform all the required tasks within the required response time to satisfy the overall system performance requirements.

If you leave this operator unchanged, it throttles the rate of throughput and requires you to spread the workload over multiple hosts.



## Parallelism of operators

One option to spread the workload is to segment the stream into a number of substreams, each of which can be processed in parallel on different host computers, as shown in Figure 5-10.

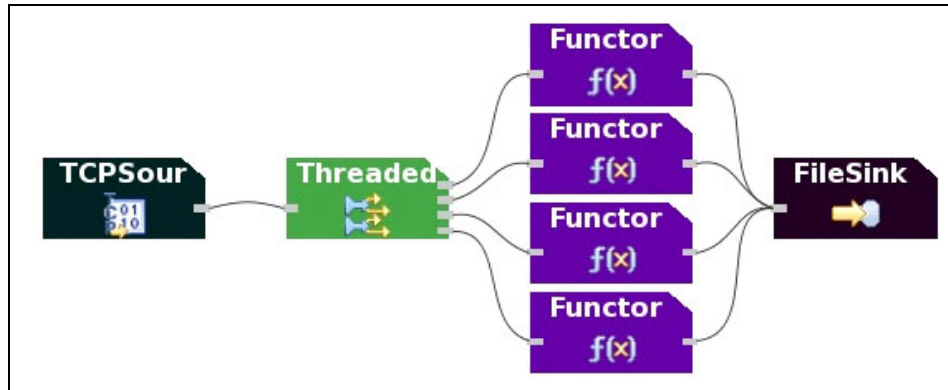


Figure 5-10 Parallel operator example

## The problem of state

Stream computing works on the general principle that applications are stateless and that data that is passing through the system is not stored. However, a number of the Streams operators are not totally stateless, meaning they can be configured to keep state between subsequent tuples. When this configuration is used, splitting a stream for parallel processing can change the results of the operators.

The following list outlines the state management possibilities for the Streams operators:

- ▶ Tuple history: Expressions in parameters or output assignments of operator invocations can refer directly to tuples that were received. History access maintains a list of tuples that can be subscribed to by using the input port name.
- ▶ Operator windows: A window is a logical container for tuples that were recently received by an input port of an operator.
- ▶ Operator custom logic: The logic clause consists of local operator state that persists across operator firings and statements that run when an input port receives a tuple or a punctuation and thus fires.
- ▶ Generic and Non-generic Primitive operators: User-written operators written in C++ and Java can include any number of state mechanisms built into them. These operators are controlled by the operator developer and should be well-documented.

You might want to segment a stream to distribute its workload. However, be aware that this action can change the behavior of the stateful operators as they now receive only a subset of the original stream.

For example, consider a Join operator that joins data from multiple high-volume streams. If these streams are segmented to distribute the processing load, the data items that might match in a single operator are spread out over different join operators and the number of matches might be reduced.

A segmenting a Join operator is shown in Figure 5-11.

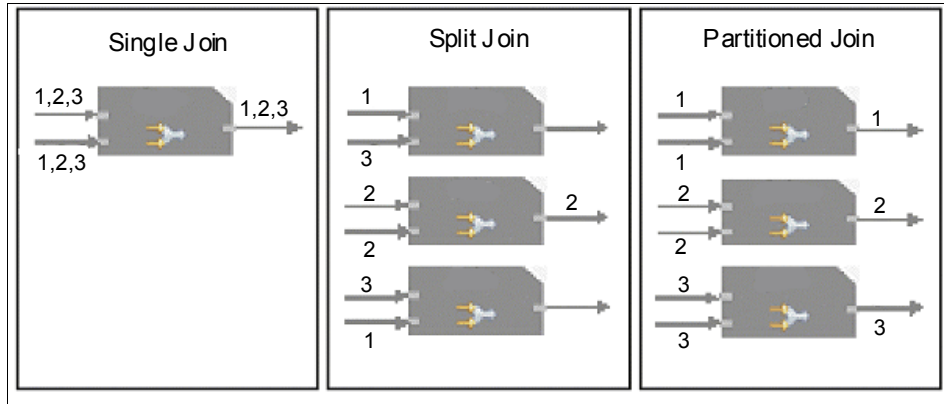


Figure 5-11 Parallel Join operator side effects

- ▶ In the single join case, the operator processes three tuples (1,2,3) from both streams. Three matches are output from the Join operator.
- ▶ In the split join case, the streams are segmented and processed by three separate join operators. Only one of the tuple pairs matches in this case.

In the partitioned join case, we ensured that the streams are segmented to send related tuples to the same Join operator. All three matches are identified and output.

The use of partitioned windows can help when you are parallelizing operators. If you split the data by using the windows partition key attribute, data is kept together, which preserves the integrity of partitioned window functions.

Consider the example of a stock trading application where you calculate the average price of a stock for each company in your portfolio. You can split the stock price stream by the stock symbol, as tuples are kept together and your aggregate is unaffected.

In contrast, if you want to keep an average of all stock quotes together, segmenting the streams by the stock symbol changes the aggregate calculation.

### Pipelining as an alternative

An alternative to splitting a stream and processing in parallel is to use the pipeline pattern. With this option, processing is divided into multiple stages, which can be performed one after another.

Pipelining features larger communication requirements than parallelism. There is a processing impact and a time penalty in communicating between processing elements, especially when the PEs are on different hosts.

In the case of pipelining, every tuple is sent to every operator, so if you have five separate operators pipelined together, and each operator is running on separate hosts, you must send the tuples across all five hosts.

In contrast, if you segment a stream into five parallel operators, each tuple must be sent between hosts only twice (once to the Parallel operator and then back again). However, you incur a processing impact when the streams are segmented.

## Parallel pipelining

The ultimate solution for many applications that are facing a performance bottleneck is to combine parallel processing and pipeline processing, which results in parallel pipelines, as shown in Figure 5-12.

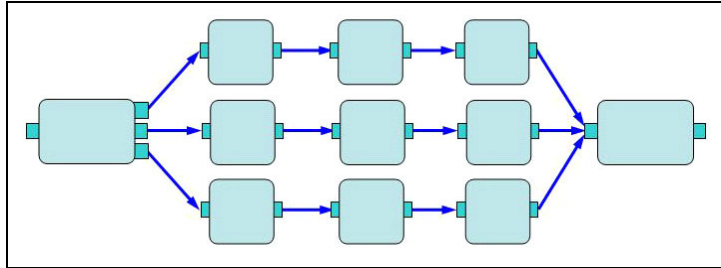


Figure 5-12 Parallel pipelining

Combining these two patterns results in the following benefits:

- ▶ Tuples that require more processing (because of the size or complexity of the data) might slow down one pipeline. However, the other pipelines still process tuples. This configuration allows the faster tuples to pass the slower tuple as opposed to causing other tuples to clog up within the system.
- ▶ Each instantiation of the pipeline can be fused or at a minimum co-located on the same host, which reduces network impact for each tuple. The set of instantiations of the pipeline can be distributed across a set of hardware, thus taking advantage of available resources.

## 5.5 Failover, availability, and recovery

InfoSphere Streams provides the following facilities to allow recovery from failed hardware and software components:

- ▶ Restarting and relocating processing elements
- ▶ Recovery of failed application hosts (hosts that are running only the host controller and PEs)
- ▶ Recovery of management hosts (a host that is running any of the SRM, SAM, SCH, or AAS management services)

Recovery of application hosts and processing elements (PEs) do not require any other instance configuration, but the extent to which processing elements can be recovered does depend on how they are configured within an SPL application.

Recovery of management hosts requires a recovery database (DB2 is required) and two changes to the instance configuration; that is, the recovery mode must be set to true and you must use a file system-based name service (NameServiceUrl=FS:).

These recovery situations are described in the following sections.

## 5.5.1 Restarting and relocating processing elements

Recovery and relocation of stream processing applications includes a few factors that can limit and even prevent recoverability and relocation, especially in a multi-host topology. For example, if operator A adds a counter to tuples that it processes, and operator B, which is down stream, depends on those counters continuing to increase, a restart of operator A that does not continue where it left off could adversely impact the logic and integrity of your application results.

In another case, if operator C reads and writes information to a local (non-shared) file system, and Streams were to restart that operator on a different host, the file system might not exist or might not be in a configuration that is compatible with the operator. Therefore, this situation could cause further failure or invalid behavior.

The Streams runtime engine does not have insight into the internal workings of each operator. It also does not understand the dependencies between operators that were built into your application (beyond the operator-to-stream connectivity). For this reason, the Streams run time takes a conservative view of processing-element recovery and relocation.

By default, Streams assumes that an operator (and thus the processing element that contains it) cannot be restarted or relocated. It is up to you as the application developer to provide configuration directives that tell Streams which operators can be restarted and can be relocated.

To ensure that Streams can automatically start restartable and relocatable PEs on a different host after a host failure, do not specify specific host names in the placement configuration section of your operators.

### Restartable operator configuration

An operator instance can be restarted after a user-requested `streamtool stopPE` command completes or after a partition crashes because of a runtime error if it is marked as restartable in its configuration clause. In Example 5-11, we provide an example of a TCPSource operator that is used in the role of a TCP server. We can mark this operator restartable because it does not contain a state, but we might not be able to make it relocatable because TCP clients might not find it if it is restarted on a different host.

#### *Example 5-11 Restartable example*

---

```
stream<unit32 empid, float64 rate, float64 hours> payroll =
  TCPSource(){
    param
      role : server ;
      port : "55555" ;
      format : csv ;
    config
      restartable : true;
      relocatable : false; // default
  }
```

---

### Relocatable operator configuration

An operator instance can be relocated on a different host after a failure or for better load balancing, which is based on relocation recommendations from the scheduler. The relocatable configuration implies the restartable configuration, but you might want to specify both for clarity.

In Example 5-12, we provide an example of a TCPSource that is used in the role of a TCP client. This operator can be restarted and relocated without causing our application any problems. However, if the network configuration of the environment has limited connectivity (through a firewall) to the TCP server and a limited number of hosts, you might not be able to relocate operators within this environment.

*Example 5-12 Relocatable example*

```
stream<uint32 empid, float64 rate, float64 hours> payroll =
  TCPSource(){
    param
      role : client ;
      address : "HRServer1.mycompany.com" ;
      port : "55555" ;
      format : csv ;
    config
      restartable : true ; // implied by relocatable : true
      relocatable : true;
  }
```

**Important:** If an operator A is restartable and operator B is not, A and B are partition-exlocated (meaning they cannot be in the same partition), unless the compiler option `-O, --setRelaxFusionRelocatabilityRestartability` is used.

## Manual and automatic restart

Automatic restarting of processing elements are governed by the following rules:

- ▶ Processing elements that are marked as restartable (but not relocatable) are automatically restarted when they fail or crash if the application host they were running on is available.
- ▶ Processing elements that are marked as restarted and relocatable are automatically restarted on the same host or a different host when they fail or crash.
- ▶ Processing elements that are not marked as restartable can be restarted only by canceling and resubmitting the application as a new job. If these types of PEs fail, the entire application must be canceled and restarted.

**Important:** When a PE or a Streams Job is restarted, entries in the data streams might be missed.

In addition to automatic restart because of a failure, you can stop and restart processing elements manually. The automatic restart rules are still enforced; for example, you cannot restart a processing element manually that is not configured to be restartable. Manual processing element control can be performed by using the `streamtool` command and the Streams administration console. The following subcommands are used through the `streamtool` command:

- ▶ `streamtool stoppe` is used to stop a specified processing element.
- ▶ `streamtool restartpe` is used to start a specified processing element on the same host or a different host.
- ▶ `streamtool lspec` is used to list the processing elements in the instance and to view their health and host assignments.

## 5.5.2 Recovering application hosts

Application hosts are configured to run the Streams HC service. The information in this section also applies to Mixed hosts because they run an HC. As described previously, the HC service controls the running of processing elements on a Streams host. If the host controller fails or stops responding, the SRM flags that the host cannot be scheduled. After a short period, the SAM changes the state of all PEs that are running on the failed host to Unknown. For example, if the state is *Running* when the HC fails, the state becomes *Unknown*.

The status of hosts in a running instance can be viewed by using the **streamtool getresourcestate** command. The Schedulable column can contain the following values:

- ▶ A value of yes indicates that the host is available to run PEs.
- ▶ A value of no (and the reason text that is shown with it) indicates that the HC service on the host is not functioning and that any PEs on that host are in the Unknown state. The reason text indicates why the host is not available to run PEs.
- ▶ A dash (-) indicates that there is no HC service running or planned on the host.

Figure 5-13 shows the results of the **streamtool getresourcestate** command after a host controller failed on the specific host named redbook6.

```
$ streamtool getresourcestate
Instance: streams@streamsadmin Started: yes State: PARTIALLY_FAILED
Hosts: 7 (1/1 Management, 5/6 Application)
Host      Status  Schedulable  Services
redbook1  RUNNING -            RUNNING:aas,nsr,sam,sch,srm,sws
redbook2  RUNNING yes         RUNNING:hc
redbook3  RUNNING yes         RUNNING:hc
redbook4  RUNNING yes         RUNNING:hc
redbook5  RUNNING yes         RUNNING:hc
redbook6  FAILED  no (failed)  FAILED:hc
redbook7  RUNNING yes         RUNNING:hc
```

Figure 5-13 Failed application host

PEs in the Unknown state cannot be restarted automatically or with the **streamtool restartpe** command. However, they can be stopped with the **streamtool stoppe** command, but do not move to the Stopped state until the problem with the host controller is resolved.

To repair a host that has a value of no in the Schedulable column, the administrator can perform the following actions:

- ▶ Restart the HC (**streamtool restartservice**): This command attempts to restart the host controller service on the specified host. If the host controller does start, any PEs that are running on that host move from the Unknown state to the Running state if they did not also fail.
- ▶ Quiesce the host (**streamtool quiecehost**): This command can be performed only if the host is not running a Streams management service. Quiesce performs a controlled shutdown of the PEs and the HC on a host. PEs are relocated, if possible. The host remains in the instance's configuration.
- ▶ Remove the host (**streamtool rmhost**): This command can be performed only if the host is not running a Streams management service. The command here performs a controlled shutdown of the PEs and the HC on a host. PEs are relocated, if possible. The host is removed from the instance's configuration.
- ▶ Remove the HC from the host (**streamtool rmservice**): This command removes the HC from a host, which prevents the scheduler from running PEs on that host. The PEs on that host are relocated, if possible. The Schedulable column that is shown as a result of running the **streamtool getresourcestate** command contains a dash (-).

In the last three cases, SAM attempts to move restartable and relocatable PEs to other hosts. If SAM cannot move the PEs to other hosts, the PEs remain in the Stopped state.

### 5.5.3 Recovering management hosts

The default behavior of a Streams instance is to not allow the recovery of management services. In this default mode, if one of the instance's management services or hosts fails, you must restart your instance to restore it to a fully operational state.

One exception to this default behavior is the SWS. This service can be stopped or restarted without affecting the Streams instance or requiring special instance configurations.

To enable the recovery of any failed management services, Streams can work with an IBM DB2 database server to store the runtime data in a recovery database. For example, if a host is running the SAM server and the host crashes, the Streams instance administrator can restart the SAM server on another cluster host and the Streams run time restores its state from the values that are stored within the recovery database.

#### Streams recovery database

The Streams recovery database is a DB2 database that records the state of an instance's services. If instance services fail, Streams restarts them and restores their state by retrieving the appropriate state data from the recovery database.

In addition, the Streams recovery database records information that enables the Streams instance administrator to manually restart the following management services, move these management services to another host, or both:

- ▶ Streams Resource Manager
- ▶ Streams Application Manager
- ▶ Authentication and Authorization Service

The information that is stored in the recovery database includes the following states for an instance and its applications:

- ▶ Job state
- ▶ PE placement
- ▶ Stream connection state information
- ▶ Instance resource state (instance state, instance daemon placement, and state)
- ▶ Authentication and authorization state information

A steady state is when all jobs are submitted and running and there are no changes to the state of the instance. The recovery database is not active when an instance is in the steady state. In addition, the recovery database is unaffected by the activity of Streams applications. This action prevents the configuration of failover and recovery from impacting the performance of Streams applications.

For more information about configuring the Streams Recovery Database, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888.

#### RecoveryMode instance property

To configure your instance to use the Streams Recovery Database, you must set the Streams instance property `RecoveryMode` to `On`, as shown in the following example:

```
$ streamtool setproperty RecoveryMode=on
```

This attribute can be set also by using Streams Console.

**Restart the instance:** If the instance is running when you set the `RecoveryMode` to on, you must restart the instance for the change to take effect.

## File system based name service

By default, the `NameServiceUrl` property for a Streams instance is set to `DN:`, meaning a distributed name service is used. A distributed name service does not support a Streams recovery database. In contrast, it is a good idea to use a file system-based name service when the recovery of management services is critical. To use a file system-based name service, set the `NameServiceUrl` property to `FS:`.

## Recovery process

If your Streams instance is configured with the `RecoveryMode` property set to on and you configured a recovery database on one of your cluster's hosts, individual Streams management components can be recovered if they fail.

If more than one of these management services fail, restart them in the following order to ensure that any prerequisite service is already restarted:

1. SRM
2. AAS
3. SCH
4. SAM
5. SWS

You can view the status of all services by using the `streamtool getresourcestate` command or by using the Streams Console.

To restart a service on the configured node, use the `streamtool restartservice` command. For example, to restart the SAM service, enter the following command:

```
$ streamtool restartservice sam
```

To move a service from one node to another node within the instance, use the `streamtool addservice` command. For example, to move the SAM service to the host named `hostname`, enter the following command:

```
$ streamtool addservice --host hostname sam
```





## Application development with Streams Studio

In the previous chapters of this book, we introduced IBM Big Data strategy and how InfoSphere Streams manipulates the data in motion to provide low-latency analytics. We also covered the InfoSphere Streams architecture, and the new features in version 3.0.

InfoSphere Streams applications use Streams Processing Language (SPL) as the primary development language, although developers found that the analytic toolkits and programming tools provide a high level of agility when applications are built. However, one of the problems that is faced by the decision makers was the difficulties non-programmers had in developing applications. One of the best new features of InfoSphere Streams V3.0 is the ability to develop the applications faster and easier than in previous versions. IBM InfoSphere Streams 3.0 provides a drag-and-drop application composition as an innovative development programming experience and new data visualization tools, which make the application development and monitoring process become more agile and reactive to changes.

This chapter includes the following topics:

- ▶ InfoSphere Streams Studio overview
- ▶ Developing applications with Streams Studio
- ▶ Streams Processing Language
- ▶ InfoSphere Streams operators
- ▶ InfoSphere Streams toolkits

## 6.1 InfoSphere Streams Studio overview

IBM InfoSphere Streams Studio is an Eclipse-based integrated development environment (IDE) that is designed to enable application developers to create, edit, visualize, debug, and test SPL or SPL mixed-mode applications. For more information about installing Studio, see Appendix A, “Installing InfoSphere Streams” on page 249.

The default perspective for the Streams Studio is InfoSphere Streams. Studio also includes other perspective for C/C++, Java, and Debug to simplify the development of native operators. Team Synchronizing perspective is ready with the views that are required for the use of Studio in a team environment.

InfoSphere Streams Studio also includes the following major features:

- ▶ InfoSphere BigData Task Launcher that provides quick access to tasks and information for developing Big Data applications.
- ▶ Streams Explorer view that helps you set up and manage your Streams development environment.
- ▶ SPL Project and SPL Application Set Project support for organizing and building SPL applications and toolkits.
- ▶ Project Explorer view to visualize your SPL resources in a logical manner.
- ▶ Support for the use of Streams Studio in a team environment.
- ▶ Data visualization for viewing tuples from running SPL applications.
- ▶ Custom color scheme support for rapid understanding of Streams instance performance and state.
- ▶ Graph view that displays topology of application.
- ▶ Metrics view with which you view and analyze metrics from running applications.
- ▶ Log viewing support with which you can gather and examine logs from a running instance.
- ▶ Launchers for running stand-alone and distributed applications.
- ▶ Debugger for testing and debugging Streams applications.
- ▶ Integrated Help system that provides information that is needed during the development of SPL applications and toolkits.
- ▶ SPL Graphical and text editors for creating and modifying SPL applications, primitive operators, and native functions.

InfoSphere Studio uses package toolkits to include all the SPL artifacts, such as SPL and native functions or primitive and composite operators to make those toolkits reusable across all the applications that access those toolkits. A toolkit provides one or more namespaces that contain the functions and operators that are packaged as part of the toolkit.

To create a toolkit, you create an SPL project from Streams Studio that holds the following toolkit artifacts:

- ▶ Namespace
- ▶ SPL Composite Operator
- ▶ SPL Function
- ▶ SPL Type
- ▶ Primitive Operator Model
- ▶ Native Function Model

Another type of project that is called SPL application set project is used to reference a set of other SPL projects and build them as a single unit. You can use the Application graph feature to see how the applications are related to each other. This type of application is more useful with applications that import and export streams.

## 6.2 Developing applications with Streams Studio

Ease of use is a major feature of Streams 3.0.Streams Studio that offers ready toolkits that can be used to develop other applications by importing those toolkits to the developed applications. First, you must import those toolkits to the current workspace of the Studio.

### 6.2.1 Adding toolkits to Streams Explorer

Complete the following steps to add the toolkits to the Streams Explorer:

1. Start the IBM InfoSphere Streams First Steps window by using the command that is shown in Example 6-1.

*Example 6-1 Launch first steps*

```
streamtool launch --firststeps
```

2. From the *IBM InfoSphere Streams First Steps* window (as shown in Figure 6-1 ), click **Develop Application with InfoSphere Streams Studio**.

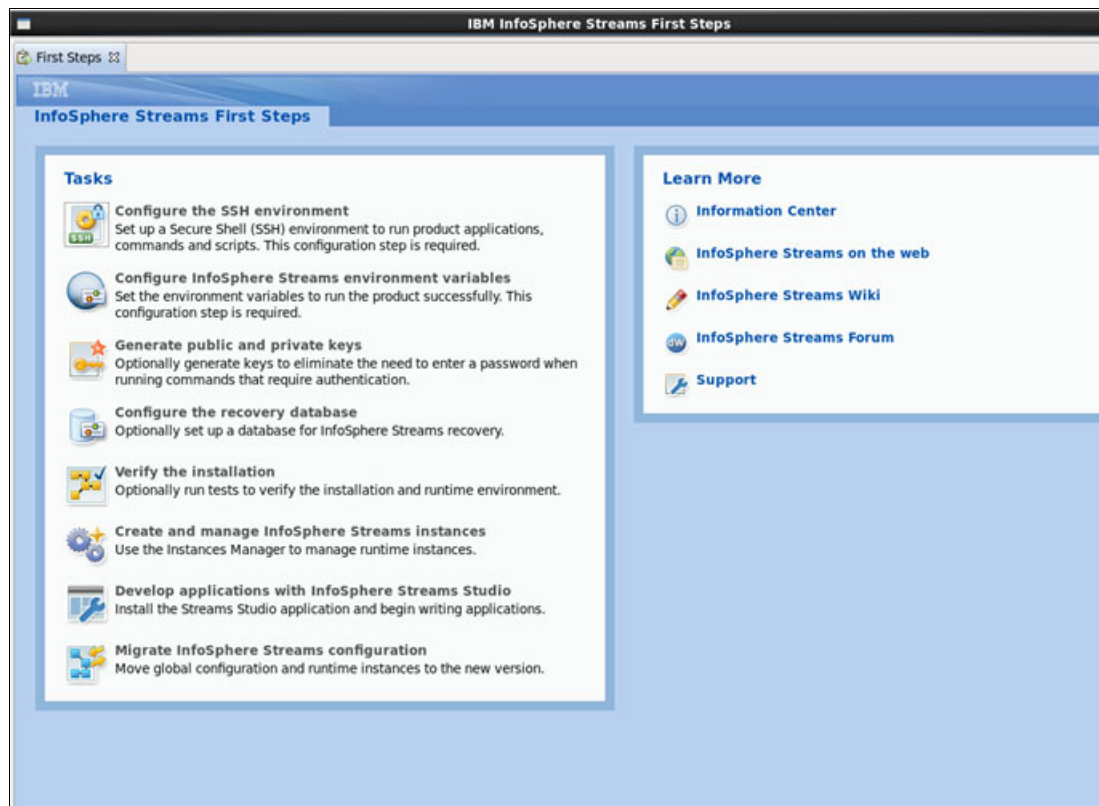


Figure 6-1 IBM InfoSphere Streams First Steps window

3. In the Develop applications with InfoSphere Streams Studio window (as shown in Figure 6-2 ), choose **Launch only**. Click **OK**.

**Important:** If Streams Studio is not already installed, the installation directory is blank and you do not see the existing installation options.

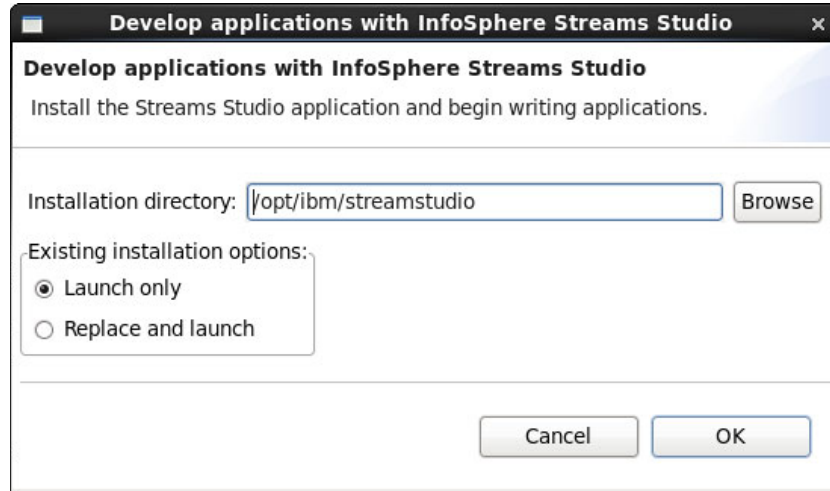


Figure 6-2 Launch InfoSphere Streams Studio

4. Select the path of the Streams Studio workspace in the Workspace Launcher window (as shown in Figure 6-3 ), and then click **OK**.

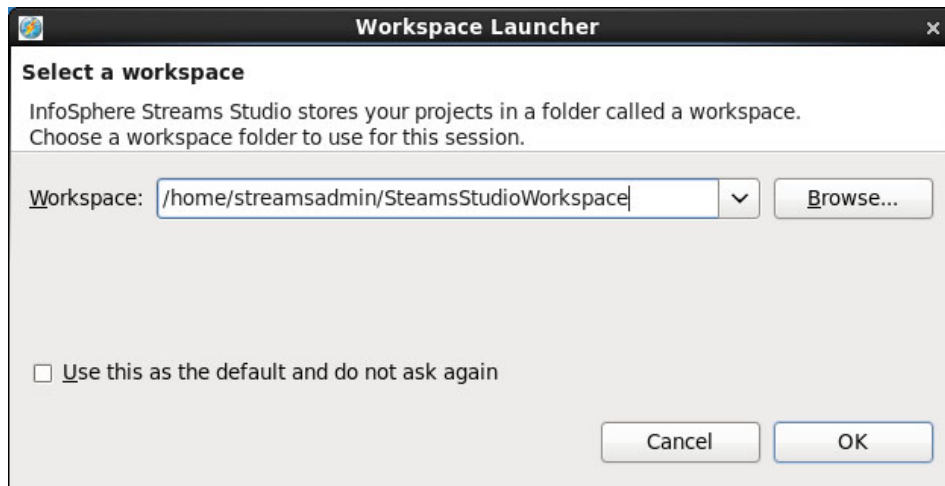


Figure 6-3 Workspace Launcher window

- Streams Studio opens in the default InfoSphere Streams perspective. Task Launcher for Big Data appears as a welcome page. It provides quick access to tasks and information for developing big data applications, as shown in Figure 6-4.

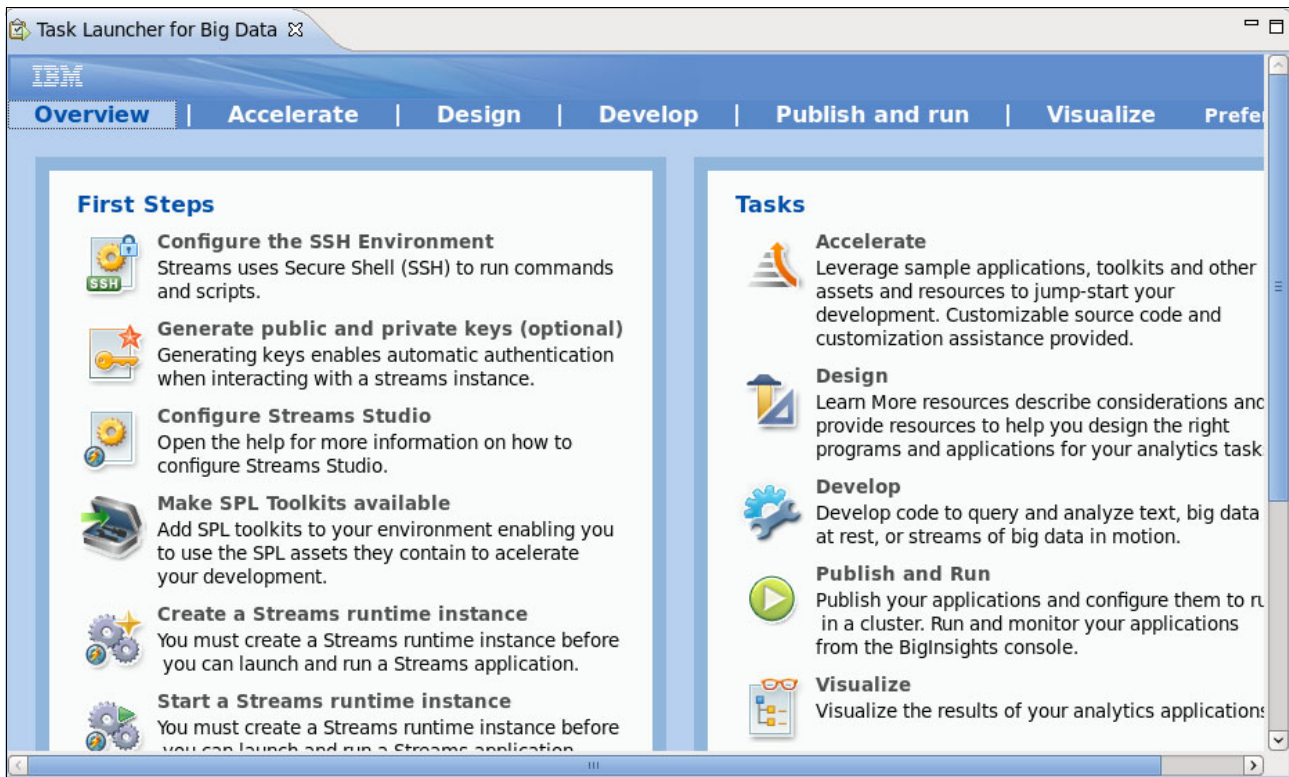


Figure 6-4 Task Launcher for Big Data

- In the *Task Launcher for Big Data* window, you quick access to tasks and information for developing big data applications. You can also find topics to help you begin your first project and create your SPL applications.

7. To make SPL toolkits available to be used to accelerate you development, click **Make SPL Toolkits available** from the First Steps panel. Streams Studio adds the toolkits under Toolkit Locations in the Streams Explorer view, as shown in Figure 6-5.

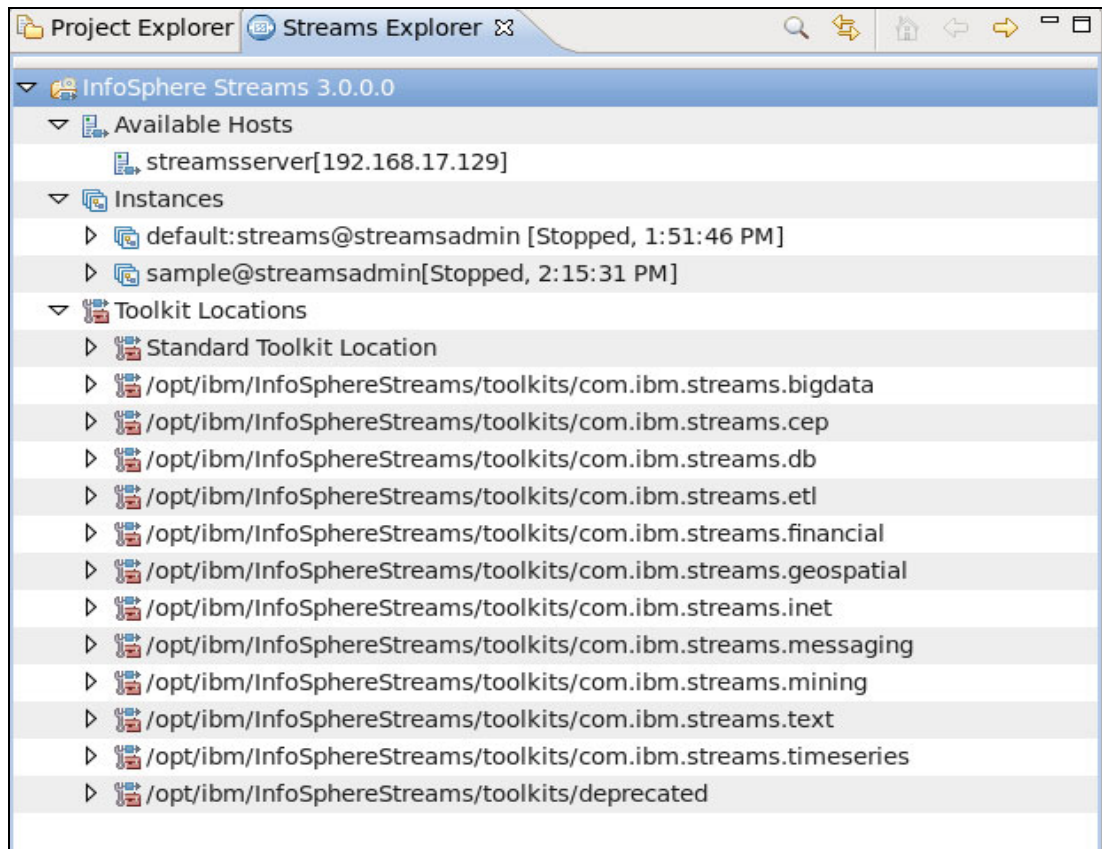


Figure 6-5 Streams Explorer view

**Important:** You can import the sample projects that are packaged with InfoSphere Streams that are found in the installation path by clicking the **Accelerate** tab from the Task Launcher of Big Data window. Click **Import sample SPL application source**. The sample projects are useful when you first explore how projects are constructed and built-in Streams Studio.

## 6.2.2 Using Streams Explorer

The Streams Explorer view that is shown in Figure 6-6 is enabled by default in the InfoSphere Streams perspective of Streams Studio. By using this view, you can manage your Streams environment. In particular, by using the Streams Explorer view, you can perform the following tasks:

- ▶ View available hosts
- ▶ Manage Streams instances and jobs
- ▶ Manage the toolkits that are available for your projects

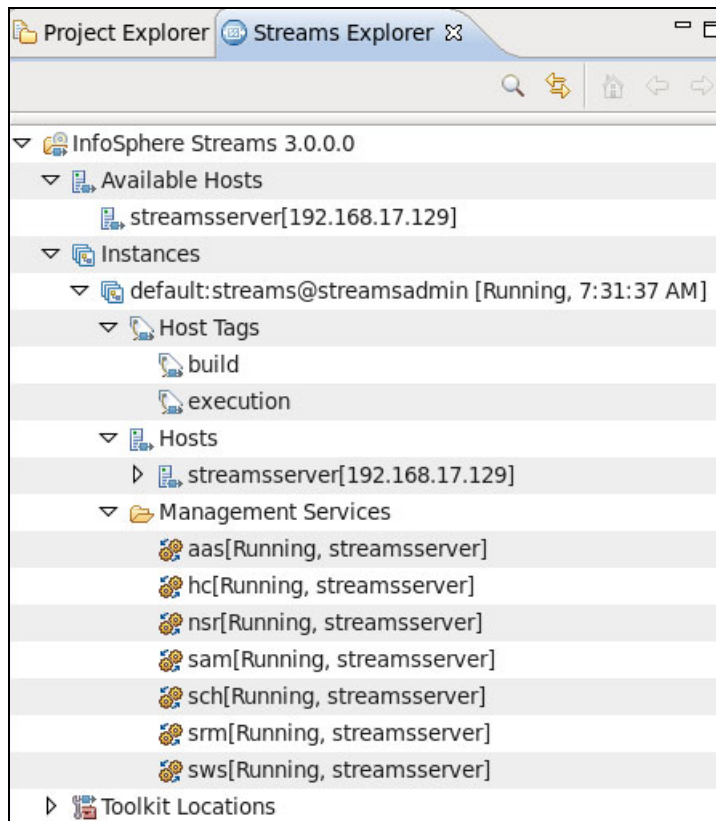


Figure 6-6 Streams Explorer

**Important:** The environment variable `STREAMS_INSTALL` is read during the start of Streams Studio to determine the location of your Streams installation. You can edit the Streams installation location for the current Streams Studio session after Studio is started to point to a new location.

### Available Hosts folder

The Available Hosts folder in the Streams Explorer view provides a read-only list of the hosts that are available for creating instances. This list is a read-only list and is the same as running the `streamstool lsavailablehost` command.

### Instances folder

The Instances folder contains a list of Streams instances. During start, Streams run time is queried for a list of instances that you own. These instances are automatically added to the Instances folder. Instances that you do not own but were added to the folder in a previous session also are listed.

For each instance, Streams Explorer provides the following folders:

- ▶ Instance Hosts: Contains the list of hosts for the instance.
- ▶ Management Services: Lists the management services that are running for the instance.
- ▶ Jobs: Each job is listed as its own folder.

From the Properties view of the Streams Studio, you can see more information about the hosts and instances.

From Streams Explorer, you can run commands such as create, start, stop, and remove a Streams instance in InfoSphere Streams Studio.

To create an instance, right-click the **Instances** folder and click **Make Instance**.

After the instance is created, you can make it the default instance, stop it, refresh it, get logs about it, and submit jobs to it.

You can add an instance through Streams Studio and remove it later. By using the Add Instance command, you can add an instance that you do not own (but can access) to the Instances folder.

Every Instance includes its own administration console on a unique port. From Streams Studio, you can open the Streams Console for an Instance. The Streams Console provides more administrative functions for instances. For more information about Streams Administration Console, see Chapter 8, “IBM InfoSphere Streams administration” on page 199.



Jobs represent running SPL applications. You can work with jobs in the Streams Explorer view in Streams Studio. You can see the jobs that are running on an instance in the Streams Explorer view. Each job contains a list of processing elements (PEs), and each PE contains its ports. You can also manage your jobs from a Streams instance in the Streams Explorer view; for example, you can submit and cancel jobs. Figure 6-7 shows how Streams Explorer shows PEs status and runtime information in the Properties view.

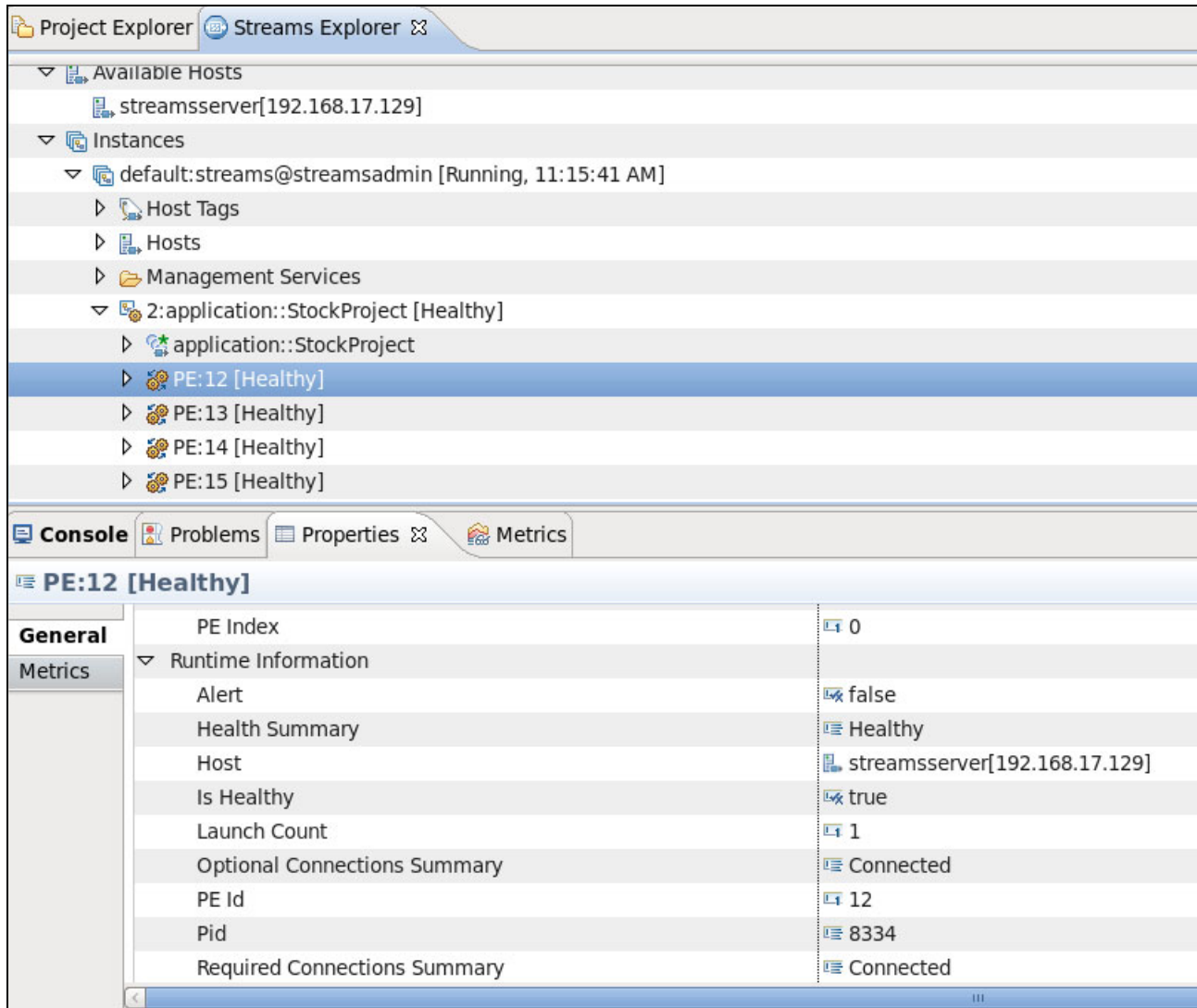


Figure 6-7 Streams Explorer jobs and PEs

Show Metrics and Show Instance Graph monitor a running SPL application in the Streams Explorer view. We describe how to use the Show Instance graph tool in 6.2.3, “Creating a simple application with Streams Studio” on page 114.

## Toolkit Locations

We explained how to add all SPL toolkits that are available in InfoSphere Streams into Streams Studio in the previous section.

A new toolkit is added by using one of the following methods:

- ▶ A directory that contains a `toolkit.xml` file. The `toolkit.xml` file defines a toolkit and is in the root of the toolkit directory.
- ▶ A directory that contains toolkit directories. By using this type of location, you can create a directory that is called `toolkits` and, within that directory, place each toolkit directory.
- ▶ An XML file that defines a set of toolkit locations. For more information about the format of this file, see the InfoSphere Streams 3.0 Information Center and search for the *Working with toolkits paths* topic, at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

To add a toolkit, right-click **Toolkit Locations** and then click **Add**. You must add the toolkit by using one of the previously mentioned methods. From Streams Explorer, you also can remove a toolkit location and refresh or search for specific toolkit elements.

### 6.2.3 Creating a simple application with Streams Studio

This section describes how to create a simple SPL project application by using Streams Studio. You see how creating applications is easier than with previous releases, so it gives you the ability to concentrate on the business value more than technical details. Streams Studio is enhanced by a drag-and-drop feature to build the applications faster.

The input stream for this application is a file that carries stock market transaction information for different companies over a specific interval of time. The purpose of the application is to transform and analyze the input stream to another stream that holds only one tuple for each ticker that contains the important attributes from the input stream, such as the ticker name, last transaction time, and last trading price. Some new information also is calculated by the application, such as the minimum trading price, maximum trading price, and the number of trading transactions that occurred on this ticker. The output stream is printed in a new file in a view of a report.

Complete the following steps to create the StockProject application:

1. From the Streams Studio, click **File** → **New** → **Project**. As shown in Figure 6-8, choose the **SPL Application Project** type and then choose a project name and namespace. Click **Finish**.

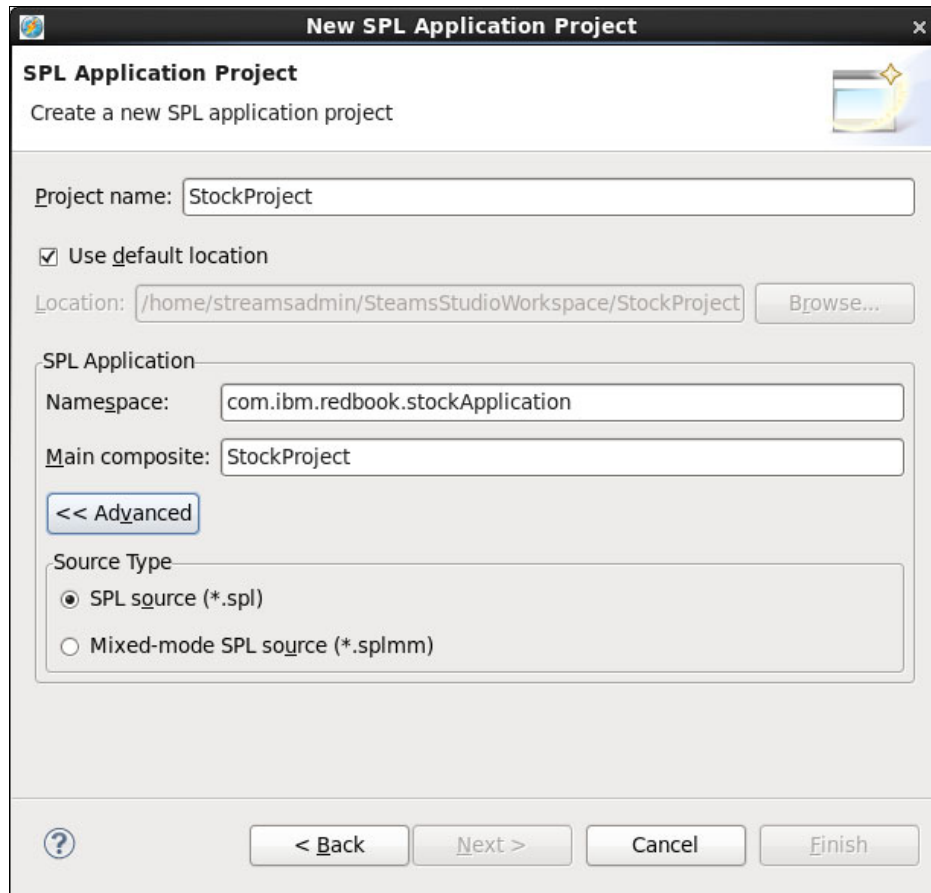


Figure 6-8 StockProject creation

- After the project is created and built, a new composite `StockProject.spl` file is created and opened in a graphical editor. A palette is included on the left side of the GUI, as shown in Figure 6-9 .

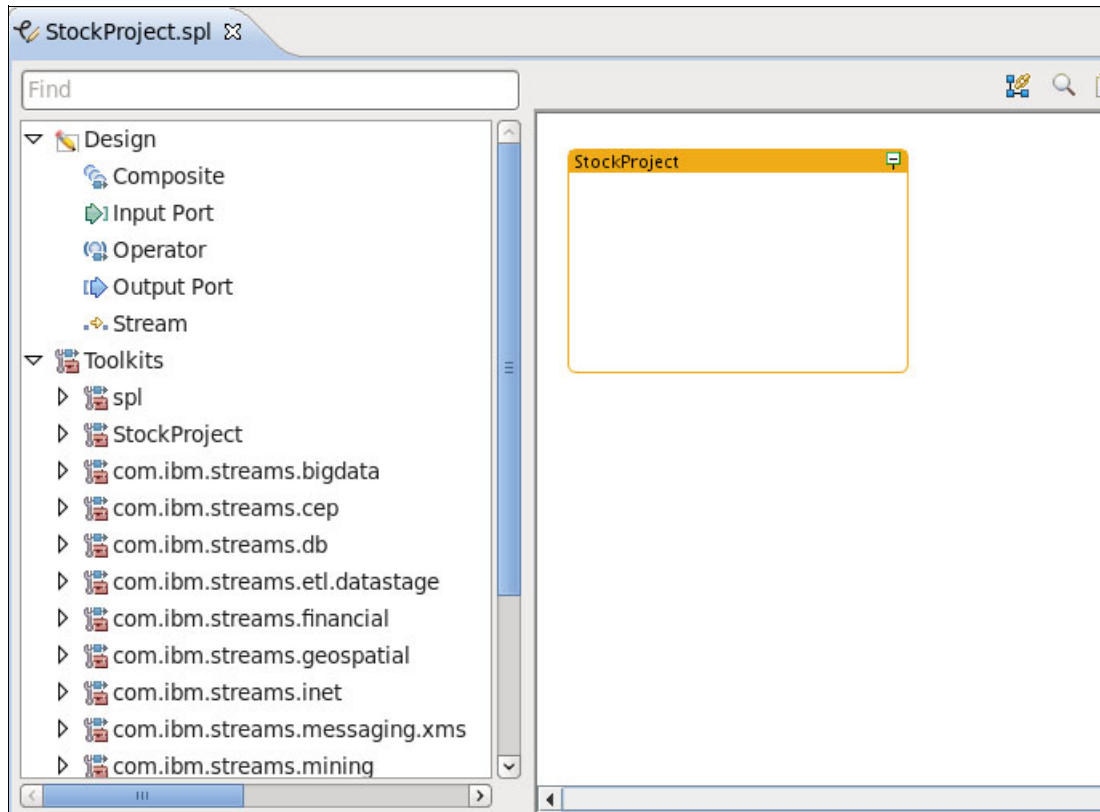


Figure 6-9 *StockProject composite*

Each tuple in the input stream consists of the attributes that are defined in Example 6-2. We define the stream that is read from the input file as `bigDataTuple`.

*Example 6-2 bigDataTuple attributes*

```

type
    bigDataTuple = rstring ticker, rstring date, rstring time, int32
gmtOffset, rstring ttype, rstring exCntrbID, decimal64 price, decimal64 volume,
decimal64 vwap, rstring buyerID, decimal64 bidprice, decimal64 bidsizes,
int32 numbuyers, rstring sellerID, decimal64 askprice, decimal64 asksizes,
int32 numsellors, rstring qualifiers, int32 seqno, rstring exchtime,
decimal64 blockTrd, decimal64 floorTrd, decimal64 PERatio, decimal64 yield,
decimal64 newprice, decimal64 newvol, int32 newseqno, decimal64 bidimpvol,
decimal64 askimpcol, decimal64 impvol ;

```

- Define the `bigDataTuple` file in the `StockProject` composite by right-clicking the composite on the canvas. Click **Edit**.

4. Select the **Types** tab then click **Add New Type** and define the `bigDataTuple` file as it is defined in Example 6-2 and shown in Figure 6-10.

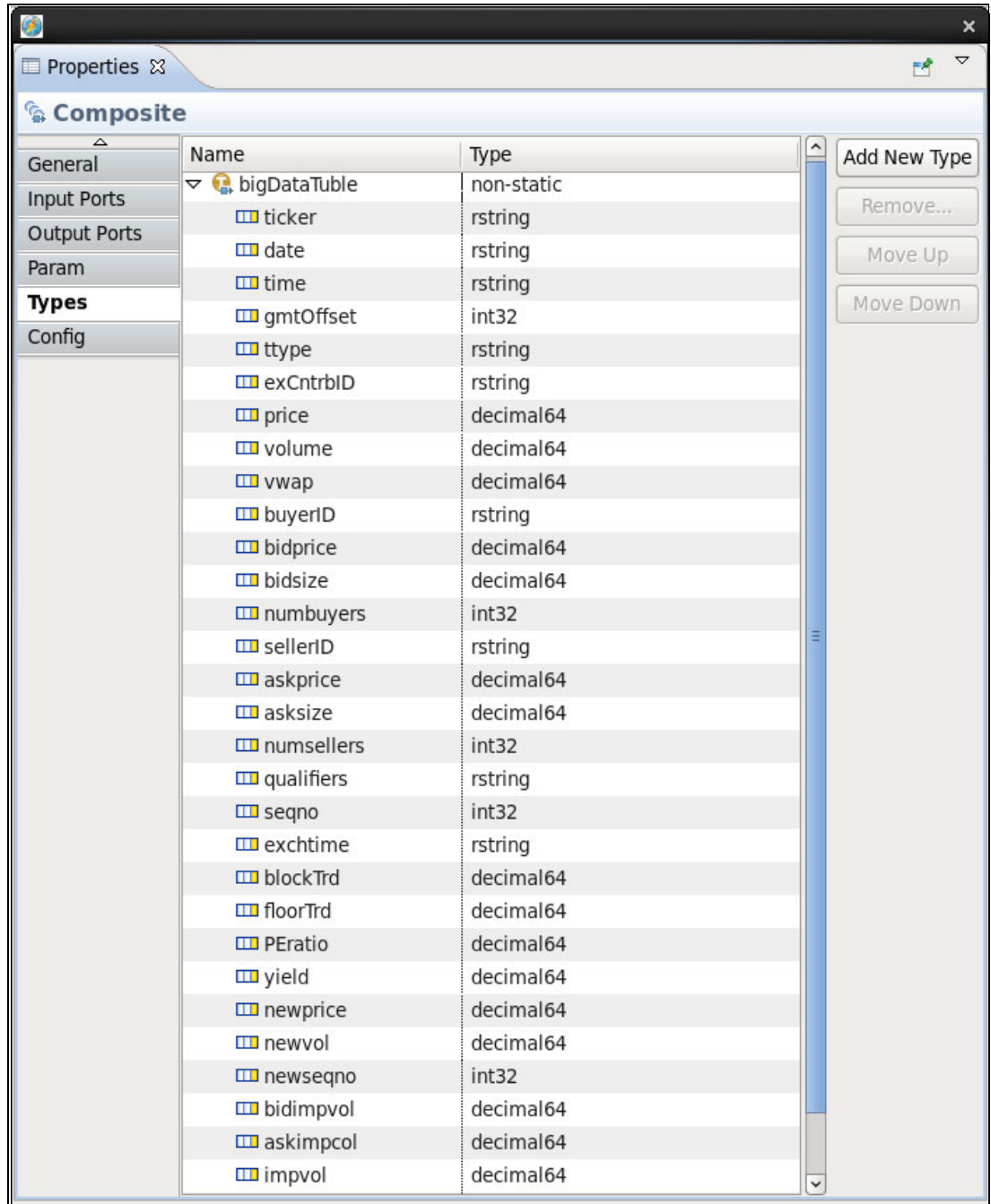


Figure 6-10 `bigDataTuple` attributes

- As with the input stream tuple definition, we define the output stream tuple attributes as shown in the Figure 6-11 . The attributes of the output stream include the count, minimum, and maximum attributes, which we calculate in our application. Close the Properties window.

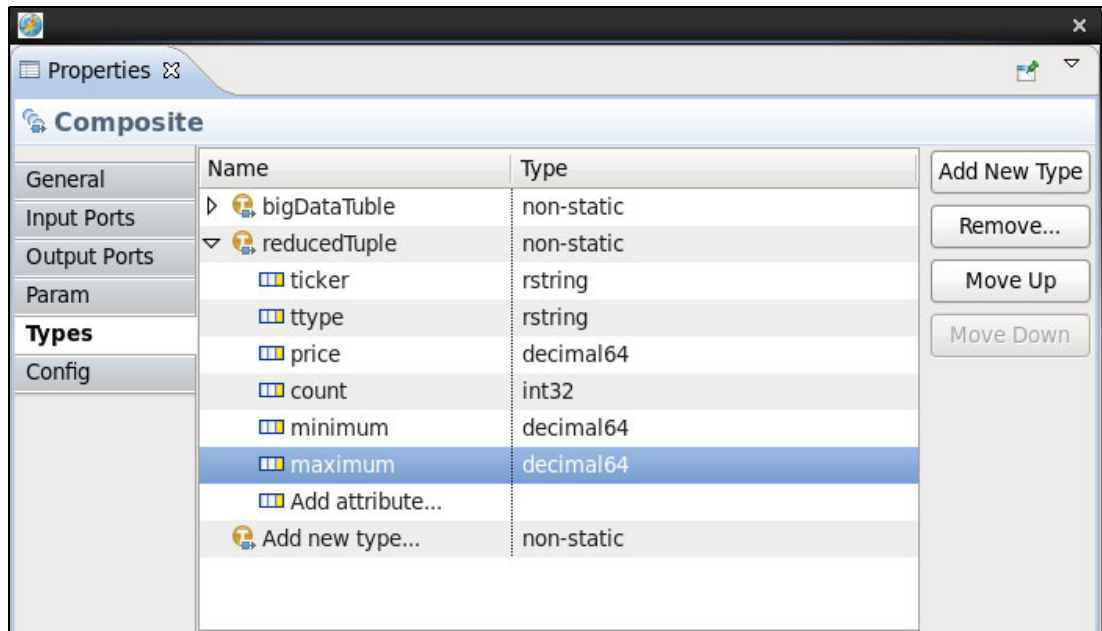


Figure 6-11 reducedTuple attributes

- Copy the radesAndQuotes.csv.gz file from the sample project STREAMS\_INSTALL\_PATH/samples/spl/application/vwap/data to the default data folder of the StockProject project: WORKSPACE\_PATH/PROJECT\_NAME/data.
- Enter File in the Find field at the top of the palette.
- Drag the FileSource operator from the palette to the StockProject composite into the canvas, as shown in the Figure 6-12 .

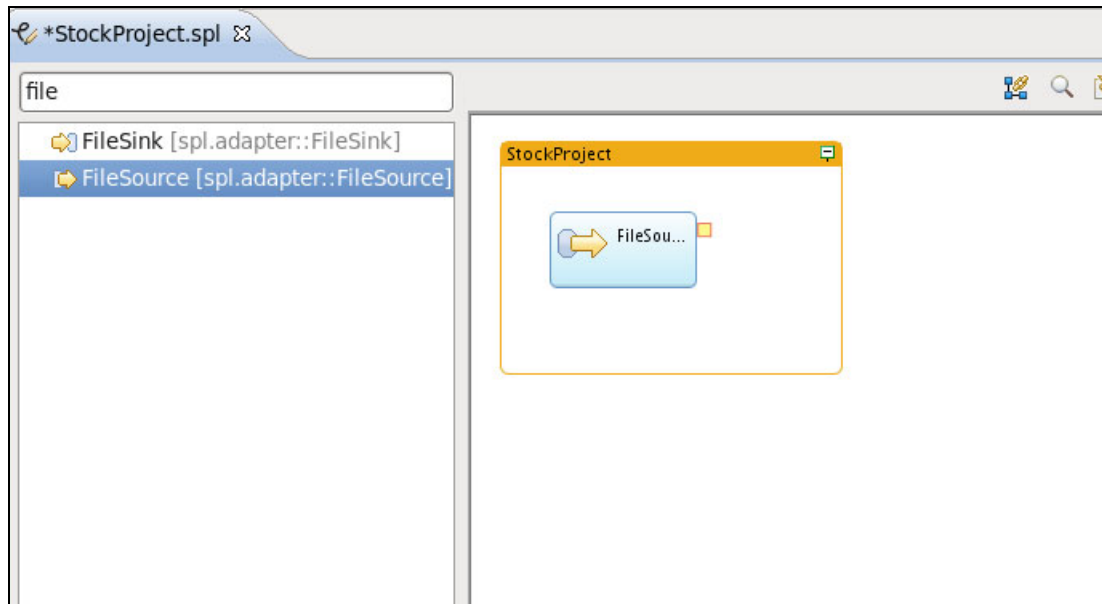


Figure 6-12 Insert FileSource operator

9. Right-click the **FileSource** operator and click **Edit** to open the properties window of the FileSource operator, as shown in the Figure 6-13.
10. From the Properties window, select the **Param** tab and then click **Add**.

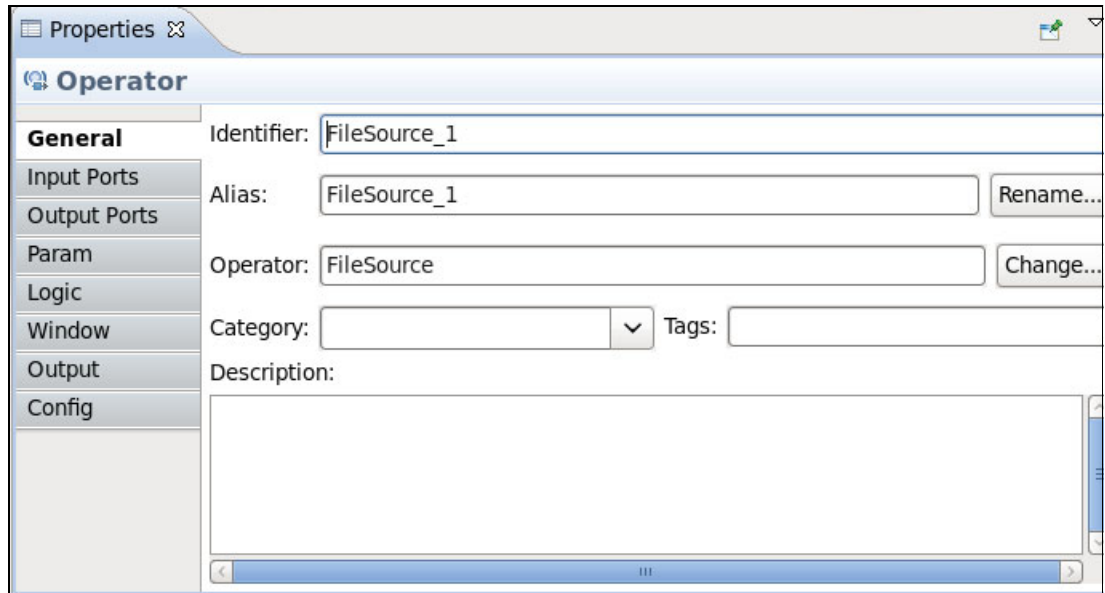


Figure 6-13 FileSource properties

11. Select **compression, file, and format** and then click **OK**.
12. Adjust the following values as shown in Figure 6-14 :
  - File parameter: "TradesAndQuotes.csv.gz"
  - Format parameter: csv
  - Compression parameter: gzip

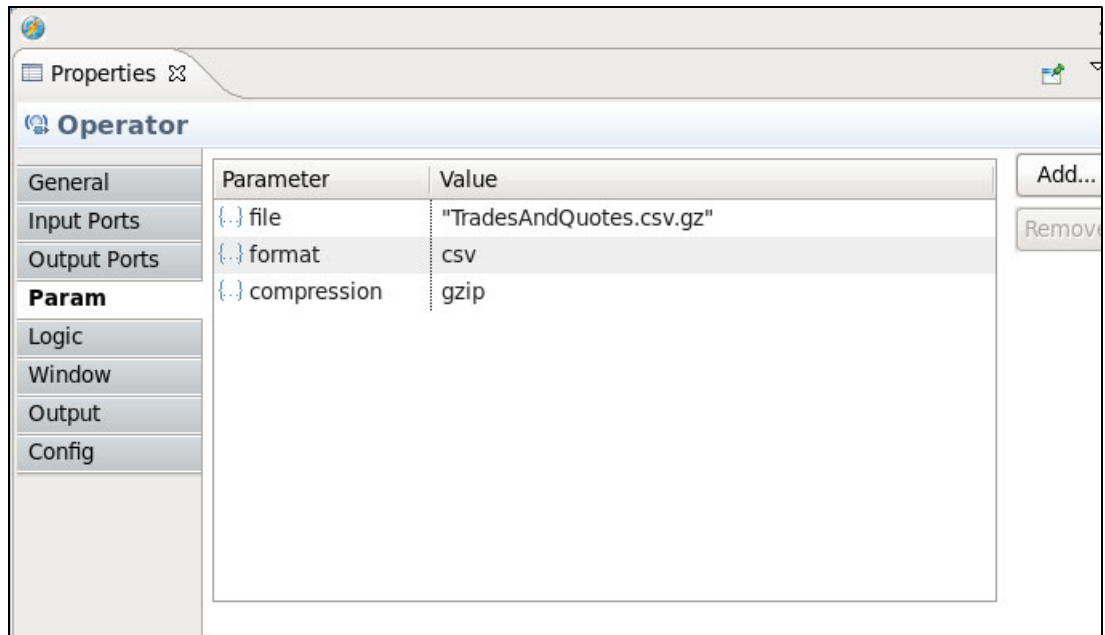


Figure 6-14 FileSource parameters

**Default workspace configuration:** The workspace by default is configured to build the projects automatically so you can see some compilation errors in the problems view. However, they are corrected after the development finishes.

13. With the Properties window still open, click the **Output ports** tab and remove the varName from the Output stream schema and add <extends> as a new attribute name with a bigDataTuple file type, as shown in Figure 6-15 ). Close the Properties window.

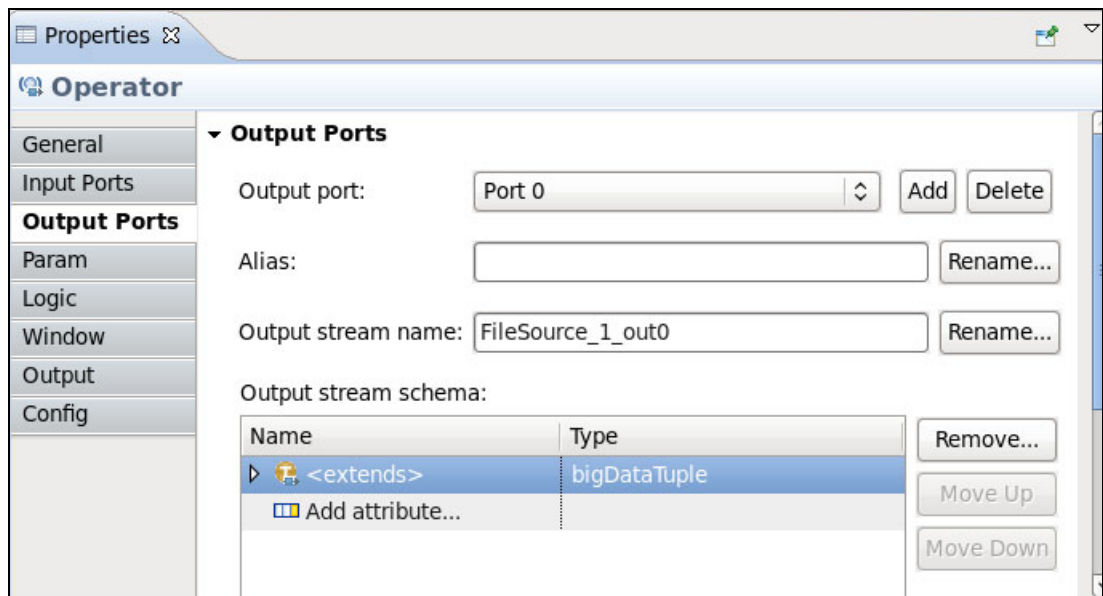


Figure 6-15 Define Output Ports

**Available options:** Press Ctrl+spacebar in the type field to see the list of the available options.

14. Enter functor in the Find field at the top of the palette.
15. Drag the **Functor** operator that is on the right side of the FileSource operator that is already in the composite file.
16. Connect the output port of the FileSource operator to the input port of the Functor operator by left-clicking and holding the output port of the FileSource operator and dragging the pointer to the input port of the Filter operator.
17. The main function of the Functor operator is to perform functional transformations, including filtering, dropping attributes from the input stream, and adding new attributes to the output stream. To do complete these tasks, right-click the **Functor** operator and then click **Edit**.
18. From the Output ports tab, remove the existing varname attribute and create an attribute as a name that is equal to <extends> and its type is reducedTuple.
19. We need to assign initial values for the new attributes that are defined in the output stream. From the Output tab, expand **Functor\_1\_out0**, expand **reducedTuple**, and then assign zero values for the **count**, **minimum**, and **maximum** attributes (as shown in Figure 6-16). Close the Properties window.



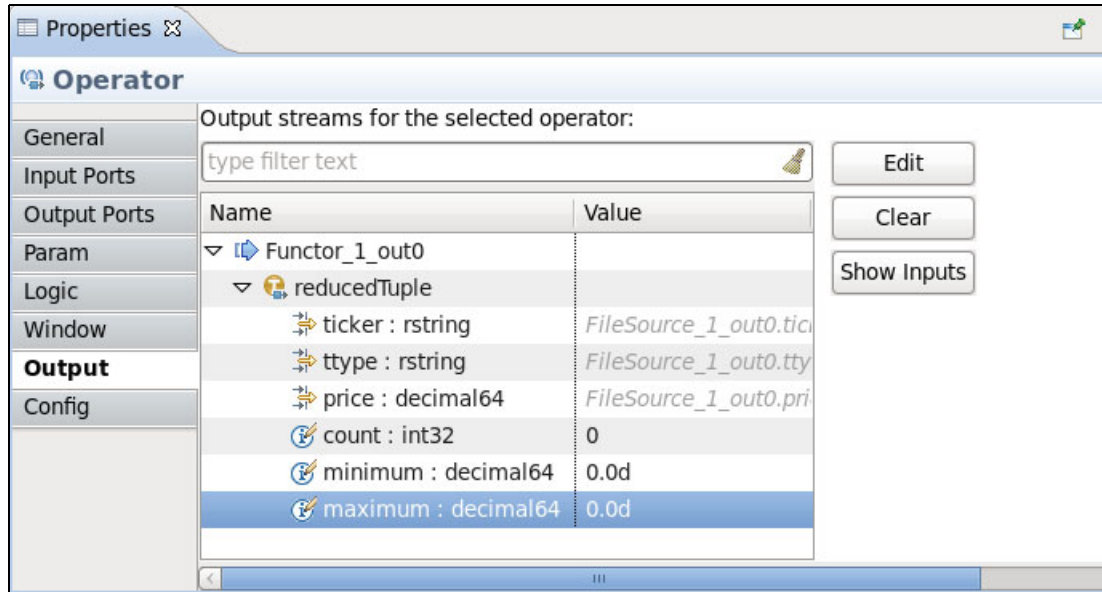


Figure 6-16 Output tab

Our report is focused only on the transactions of type Trade, so another Filter operator is required to filter on transaction types that equal to Trade. The transaction type is represented by attribute ttype.

20. Drag a Filter operator to the right side of the Functor on the canvas so that the output of the Functor is the input to the Filter.

21. Connect the output of the Functor to the input of the Filter.

22. Right-click the Filter operator and click **Edit**.

If you check the Input Ports tab, you find it is automatically configured to stream of type reducedTuple.

23. Click the **Output Ports** tab and remove the existing varname attribute. Create an attribute that uses a name that is equal to <extends> and its type is reducedTuple.

24. Click the **Param** tab, then click **Add**. Select the **filter** parameter and click **OK**.

25. Enter `ttype == "Trade"` in the value of the filter parameter (as shown in Figure 6-17) and then close the Properties window.

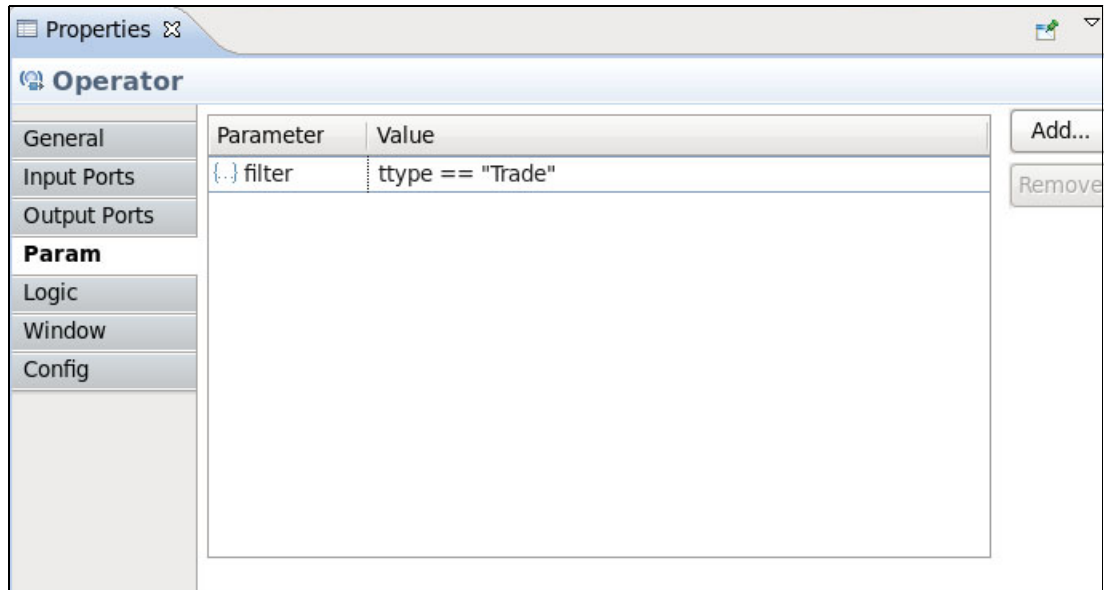


Figure 6-17 Filter operator parameter

You can click **Layout** and **Fit to Content** at any time from the toolbar of the editor to adjust the graph that is drawn.

26. Next, we want to calculate the maximum, minimum, and count for transactions that occurred per ticker name. The aggregate operator can summarize a group of data and any processing can be divided into groups where one or more attributes feature the same values. The operator also supports sliding and tumbling windows. For more information about the SPL operators, see 6.4, “InfoSphere Streams operators” on page 144.
27. Add the Aggregate operator next to the Filter operator and connect the output from the filter to the input of the Aggregate operator.
28. Click the Output Ports tab and then remove the existing varname attribute. Create an attribute that uses the name equal to `<extends>` and its type is `reducedTuple`.
29. In the Param tab, click **Add** and select the `groupBy` parameter. Click **OK**.
30. In the Value field, enter `ticker` as a value for the `groupBy` parameter.
31. In the Window tab, select the **inPort0Alias** row, then click **Edit**.

32. As we are reading data from a file, we need to produce our report after the file reading is finished. Select **Tumbling Window** and set the Eviction policy as **Punctuation - punct()**, as shown in Figure 6-18 . For more information about SPL windows, see 6.3, “Streams Processing Language” on page 136.

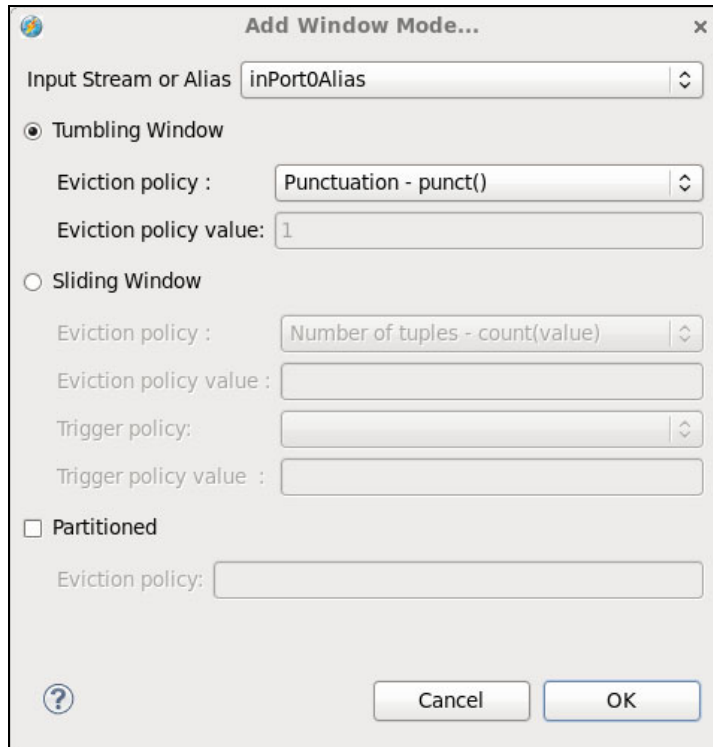


Figure 6-18 Edit window tab of Aggregate operator

33. In the Output tab, define the count, minimum, and maximum functions, as shown in Figure 6-19. Close the Properties window.

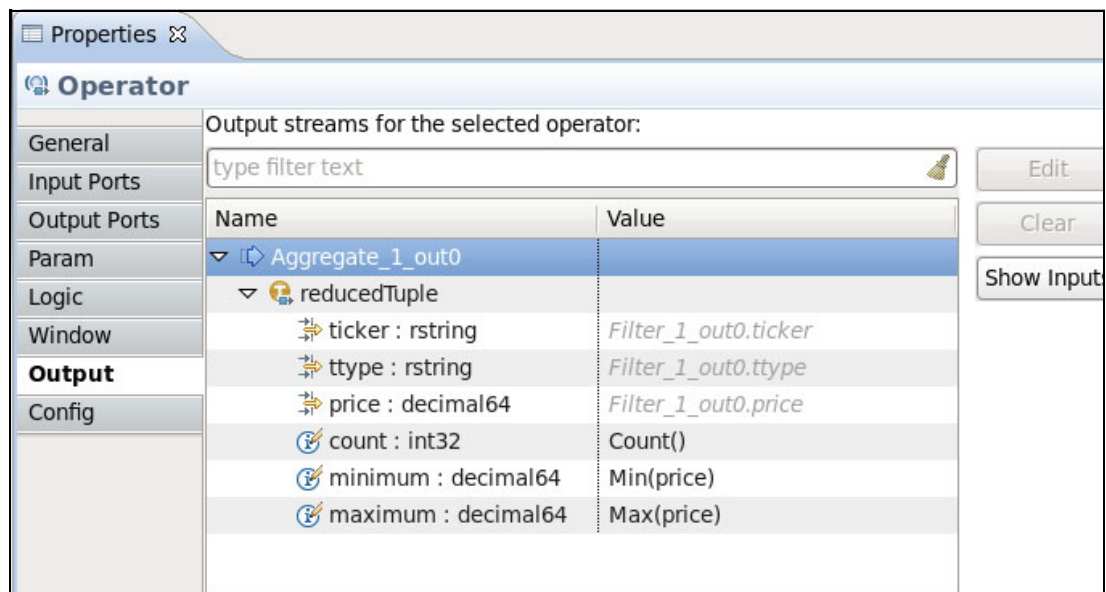


Figure 6-19 Output tab of Aggregate operator

34. The final step is to write the report to a file by using a FileSink operator. Place a FileSink operator next to the Aggregate operator. You might be familiar by now with how to connect them.
35. Right-click **Edit**, move to the Param tab, and edit the File and Format parameters to the name of the output file and the extension as they were made for the FileSource operator in Figure 6-14. Ensure that you created the file in advance.
36. The final figure for the application is similar to the application that is shown in Figure 6-20 .

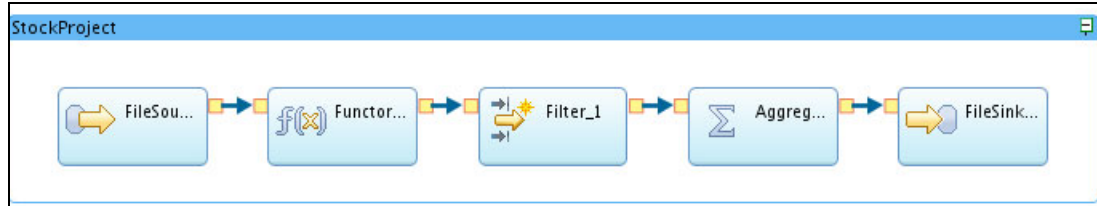


Figure 6-20 Final graph for StockProject application

37. To see the code that is behind the StockProject composite (as shown in Example 6-3), select the composite and right-click **Open with SPL Editor**.

*Example 6-3 StockProject composite*

```
namespace com.ibm.redbook.stockApplication;
composite StockProject
{
  type
    bigDataTuple = rstring ticker, rstring date, rstring time,
                  int32 gmtOffset, rstring ttype, rstring exCntrbID, decimal64 price,
                  decimal64 volume, decimal64 vwap, rstring buyerID, decimal64 bidprice,
                  decimal64 bidsize, int32 numbuyers, rstring sellerID,
                  decimal64 askprice, decimal64 asksize, int32 numsellers,
                  rstring qualifiers, int32 seqno, rstring exctime, decimal64 blockTrd,
                  decimal64 floorTrd, decimal64 PEratio, decimal64 yield,
                  decimal64 newprice, decimal64 newvol, int32 newseqno,
                  decimal64 bidimpvol, decimal64 askimpcol, decimal64 impvol ;

    reducedTuple = rstring ticker, rstring ttype, decimal64 price,
                  int32 count, decimal64 minimum, decimal64 maximum ;

  graph
    (stream<bigDataTuple> FileSource_1_out0) as FileSource_1 = FileSource()
    {
      param
        compression : gzip ;
        file : "TradesAndQuotes.csv.gz" ;
        format : csv ;
    }

    (stream<reducedTuple> Functor_1_out0) as Functor_1 =
      Functor(FileSource_1_out0)
    {
      output
        Functor_1_out0 : count = 0, minimum = 0.0d, maximum = 0.0d ;
    }
  }
}
```

```

(stream<reducedTuple> Filter_1_out0) as Filter_1 = Filter(Functor_1_out0)
{
    param
        filter : ttype == "Trade" ;
}

(stream<reducedTuple> Aggregate_1_out0) as Aggregate_1 =
    Aggregate(Filter_1_out0 as inPort0Alias)
{
    window
        inPort0Alias : tumbling, punct() ;
    param
        groupBy : ticker ;
    output
        Aggregate_1_out0 : count = Count(), minimum = Min(price), maximum =
            Max(price) ;
}

() as FileSink_1 = FileSink(Aggregate_1_out0)
{
    param
        file : "Report.csv" ;
        format : csv ;
}
}

```

---

## Conclusion

IBM InfoSphere Streams V3.0 features an enhancement for developing applications with Streams Studio. In our example, the application is developed without writing a single line of code, which is a great progress. Later, we see how testing this application becomes an interesting and time-saving process by using Streams Studio.

### 6.2.4 Build and launch the application

After the application is created, you must test it to see whether it is following your business requirements. The first step in the testing process is to build the application, then run the application.

There are two types of builds supported. One build produces a stand-alone executable file that can be run as a single process on your machine, which is called the stand-alone build. The other type of build is the distributed build that produces an SPL application that can be submitted to a Streams instance to run.

A stand-alone build configuration is used to run applications locally in the workspace. This build configuration produces a stand-alone executable file that can be run as a single process on your local machine. It does not run on the Streams runtime environment. In Stand-alone mode, all operators are fused into a single partition and a single processing element (PE) is generated for this partition. You can instruct the compiler to generate a stand-alone application by providing the `-T` option at compile time. A stand-alone executable file also is generated with the name `stand-alone` in the `output/bin` directory. When it is run, this executable file loads the PE.

A distributed build configuration is used to run an application on the InfoSphere Streams runtime environment. This build configuration creates an executable file that can be submitted to an InfoSphere Streams runtime environment for running. Unlike a stand-alone application, the operators in a distributed application can be fused to more than one PE, and the PE can be distributed into multiple hosts. To run this type of application on a Streams instance, you must provide the compiler-generated ADL to the streamtool commands.

Each main composite operator in your SPL project can include one or more defined build configurations. Only one of the build configurations for a main composite operator can be the active build configuration for that composite. The active build configuration is noted in the Project Explorer with the [Active] tag that follows its name.

When you create a main composite operator, a new build configuration that is called Distributed is created and set to be the active build configuration by default, as shown in Figure 6-21.

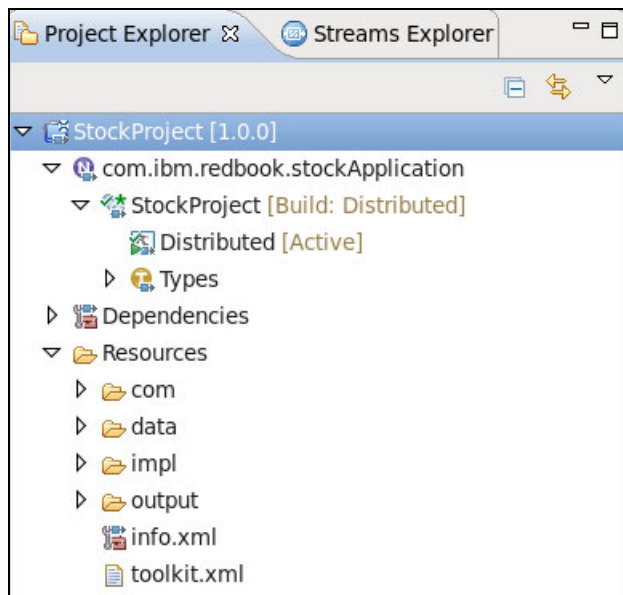


Figure 6-21 Distributed build configuration that is created by default and set as active

A main composite can be associated with one or more build configurations. An active build configuration is used to build the main composite when its project is built. Non-active build configurations are built only if the Build action is manually started. A main composite can have only one active build configuration at a time.

This information applies to projects that are configured to use the Streams Studio internal builder. For projects that use an external builder, there is no need to define build configurations.

To edit the distributed build configuration, right-click the **Distributed** configuration, then click **Edit**. The Edit Build Configuration window is displayed, as shown in Figure 6-22.

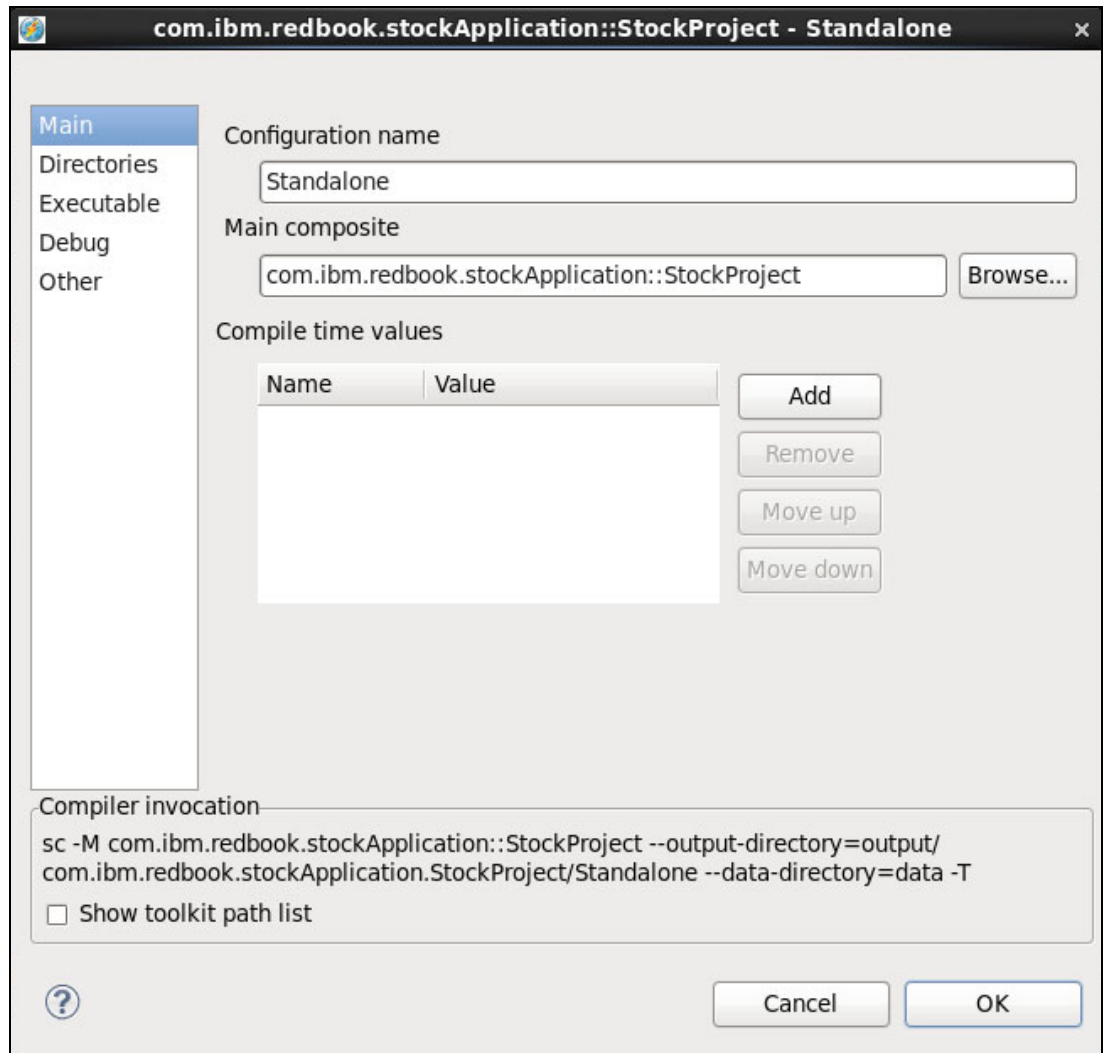


Figure 6-22 Edit Build Configuration window

Within the Main tab, you can configure the following properties:

- ▶ Enter a name for the configuration, which is Distributed or Standalone by default.
- ▶ In the Main composite field, the current SPL main composite name is displayed. You can browse to another SPL main composite if you want to change this name.
- ▶ Add compile-time arguments.

Within the Directories tab, you can configure the following properties:

- ▶ Enter the name of the output directory for your build. By default, the output directory name is the same name as the build configuration name.
- ▶ Select **Use project default data** directory if you want to use the data directory from the project options. This value is the default value, so all build configurations in the project access the same application data directory.

Within the Executable tab, you can configure the following properties:

- ▶ Use the Executable type list to select the executable file type. The choice that is selected is Distributed application; another valid option is Standalone application.
- ▶ Use the Operator fusion list to select an operator fusion. When the SPL application is compiled, each operator implementation is placed inside a processing element. Operator fusion is the process of placing multiple operators into a single processing element, which makes the connections between the operators more efficient.
- ▶ Use the Number of operator profiling samples list to select the number of operator profiling samples to include in your configuration. This option controls the `-P` option on the `sc` command.
- ▶ Use the Profiling sampling rate option to set the profiling sample rate. The profiling sampling rate controls the `-S` option.
- ▶ Use the Processing element (PE) linkage list to select the Create statically linked PEs or Create dynamically linked PEs option.
- ▶ Use the Code optimization list to select whether to generate optimized code.

**Important:** If the Executable type option is set to Standalone application, the Operator fusion, Number of operator profiling samples, Profiling sampling rate, Default pool size, and Processing element (PE) linkage options are unavailable for editing.

Within the Debug tab, use the Streams Debugger (SDB) list to select whether to debug the application with the SDB.

Within the Other tab, configure the following properties:

- ▶ In the C++ compiler options field, enter any C++ compiler options to add to your application.
- ▶ In the C++ linker options field, enter any C++ linker options to add to your application.
- ▶ In the Additional SPL compiler options field, enter any other SPL compiler options to add to your application. There are some restrictions on the values that can be entered in this field because you cannot enter the following compiler options:
  - `v -h` or `--help`
  - `v -C` or `--clean-and-suppress`
  - `v -J` or `--suppress-all-but-the-application-model`
  - `v -M` or `--main-composite`
  - `v -f` or `--spade-dps-file`
  - `v --output-directory`
  - `v --data-directory`

In the Compiler invocation section, you can select **Show toolkit path list** to include the project's toolkit path list in the compiler invocation string that is displayed.

We keep the default values for the current build configuration.



Complete the following steps to create a Standalone configuration build:

1. In the Project Explorer view, click **StockProject** → **application**, select **StockProject [Build: Distributed]**, right-click and click **New** → **StandaloneBuild**.
2. In the Build configuration window, keep the default settings and click **OK**.
3. Right-click the **Standalone** build and click **Build**. The Launch option is unavailable.
4. When you click **Build**, the console view lists the compile command, as shown in Example 6-4.

*Example 6-4 Standalone build execution*

---

```
---- SPL Build for project StockProject started ---- November 13, 2012 2:31:27 PM PST
```

```
Building main composite: com.ibm.redbook.stockApplication::StockProject using build configuration: Standalone
```

```
/opt/ibm/InfoSphereStreams/bin/sc -M
com.ibm.redbook.stockApplication::StockProject
--output-directory=output/com.ibm.redbook.stockApplication.StockProject/Standalone --data-directory=data -T -t
/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.db:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.etl:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.mining:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.messaging:/opt/ibm/InfoSphereStreams/toolkits/deprecated/com.ibm.streams.text_deprecated:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.geospatial:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.timeseries:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.inet:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.financial:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.bigdata:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.cep:/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.text --no-toolkit-indexing --no-mixed-mode-preprocessing
```

```
Creating types...
Creating functions...
Creating operators...
Creating PEs...
Creating standalone app...
Creating application model...
Building binaries...
[CXX-type] tuple<rstring ticker,rstring ttype,decimal64 price,int32...m,decimal64 maximum>
[CXX-type] enum{zlib,gzip,bzip2}
[CXX-type] enum{csv,txt,bin,block,line}
[CXX-type] tuple<rstring ticker,rstring date,rstring time,int32gmt0...ol,decimal64 impvol>
[CXX-operator] FileSource_1
[CXX-operator] Functor_1
[CXX-operator] Filter_1
[CXX-operator] Aggregate_1
[CXX-operator] FileSink_1
[CXX-operator] Throttle_1
[CXX-pe] pe com.ibm.redbook.stockApplication.StockProject-a
[LD-pe] pe com.ibm.redbook.stockApplication.StockProject-a
[CXX-standalone] standalone
[LD-standalone] standalone
```

[LN-standalone] standalone

---- SPL Build for project StockProject completed in 33.673 seconds ----

---

5. After the build is finished, the Launch option is available.

Streams Studio provides launch support for Standalone and Distributed applications. The launch mode is determined by how the application is built. If the application is built as a Standalone application, the application is launched as a Standalone application. If the application is built as a Distributed application, you can submit the application to a Streams instance by using the various Launch menu options and launch configurations.

You now have the choice to run the application as a Standalone application or as a Distributed application. Complete the following steps to run the application as a Distributed application:

1. In the Project Explorer view, click **StockProject** → **application**.
2. Select **StockProject [Build: Distributed]**, right-click and then click **Launch**.
3. The Edit Configuration window opens. Click **Continue** to submit the application to InfoSphere Streams instance.

If the default instance is not started, the Streams Studio prompts you to start the instance.

If there is no instance that is defined, you cannot launch your application in the Distributed mode. For more information about how to make or add one or more instances from the Streams Explorer view and set default instance, see 6.2.2, “Using Streams Explorer” on page 111.

4. After your application is launched, the output report of the FileSink operator is printed on the `Report.csv` file under the data folder path that is defined on the build configuration. The file contains all the company’s ticker information. Because we are interested in data only from IBM, so we add a filter to the Functor operator that limits the results that are sent to IBM tickers.
5. The result of the report file is shown in Example 6-5. The report columns represent the ticker name, transaction type, last price that is quoted, number of transaction operations, minimum stock price, and maximum stock price, which was represented by the `reducedTuple` attributes within the application.

*Example 6-5 Report.csv output*

---

"IBM", "Trade", 83.64, 32, 83.44, 83.68

---

## 6.2.5 Monitor your application

From the Streams Explorer view, you can see the jobs and PEs status, check the health of the server services, and monitor your application while it is running, as shown in Figure 6-23 .

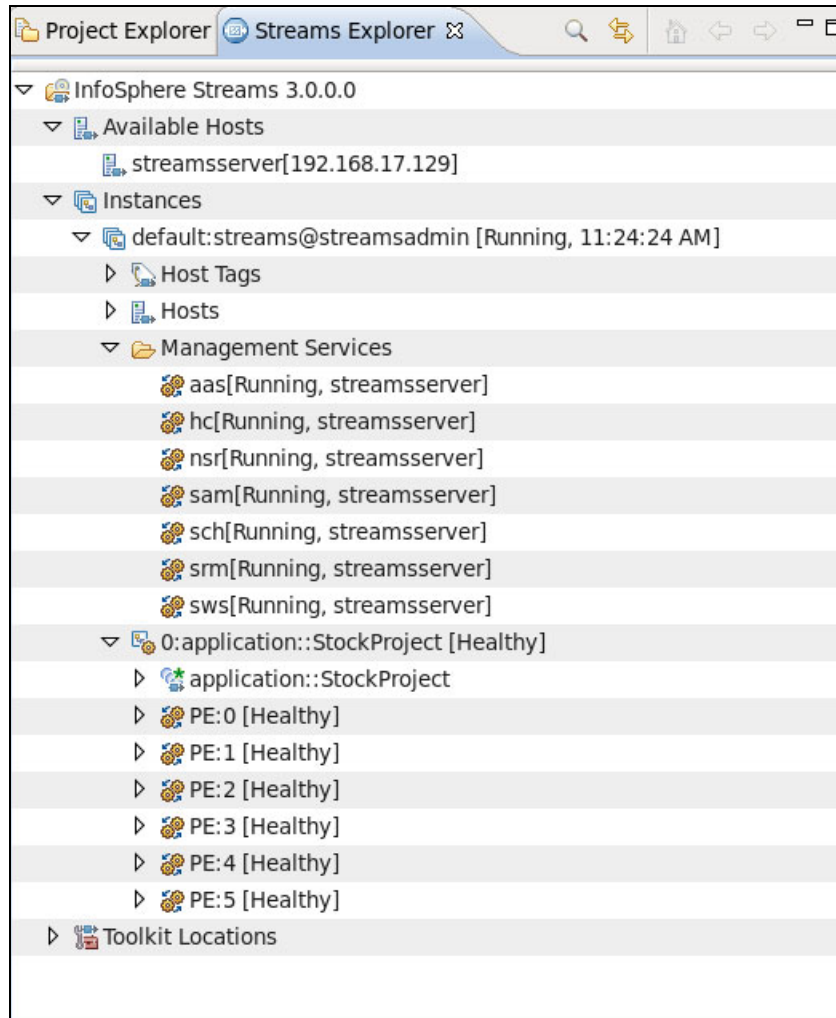


Figure 6-23 Streams Explorer

You use an instance graph to verify that your job is deployed correctly and to determine the health of your job. If your job is healthy, you can monitor the data flow in your application and verify that there is no congestion in your job. If the job is unhealthy, you can view data that is flowing through your application or retrieve log and trace data to troubleshoot issues in your application.

The instance graph provides a graphical view of the jobs that are running on a Streams instance and is the best starting point for monitoring your applications.

Complete the following steps to open the instance graph for the streams instance:

1. From the Streams Explorer, right-click the default instance (streams).
2. Click **Show instance Graph**.

By default, the Instance Graph view displays the topology of your application by using the Composite layout option, which encloses operators in their composite operators (and jobs), as shown in the Figure 6-24.

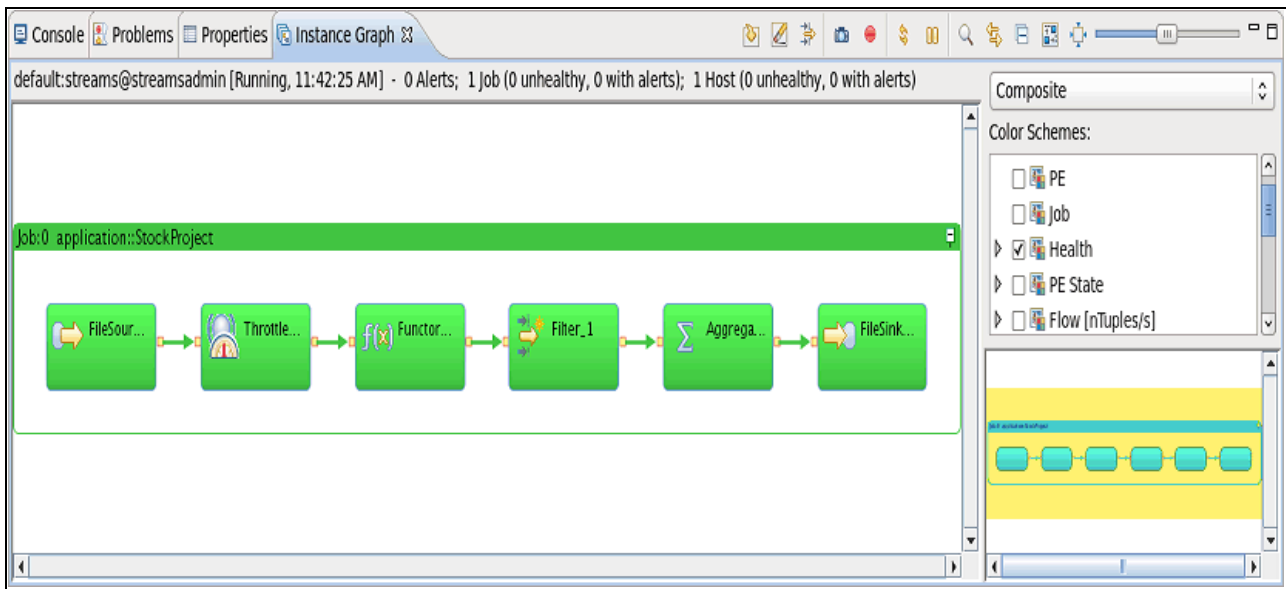


Figure 6-24 Instance Graph

In the graph that is shown in Figure 6-24, a Throttle operator was added between the FileSource operator and the Functor operator of the StockProject SPL application. The function of the Throttle operator is to reduce the rate of the tuples that are submitted to the Functor so we can be able to better monitor the application health graphs. For more information about the SPL operators, see 6.4, “InfoSphere Streams operators” on page 144.

From the Instance Graph, hover over the job to view details, such as the number of PEs in a job, the number of hosts that are assigned to the job, and the PE in which the operator is running, as shown in Figure 6-25.

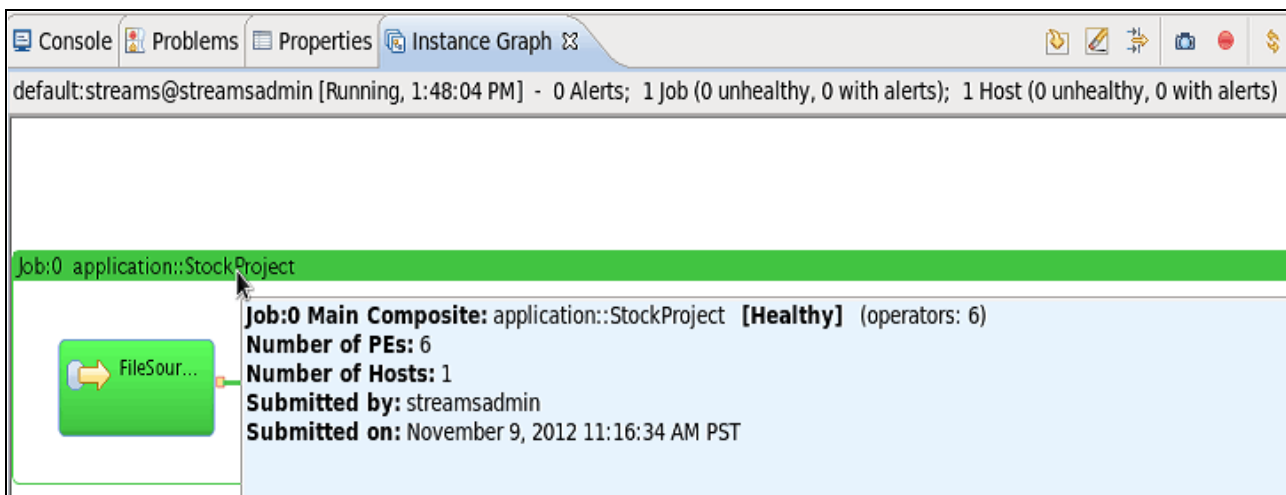


Figure 6-25 Instance Graph job details

To view deployment information about the PE and host placement, such as the PE on which the operator is running, the status of the PE, the host on which the PE is running on, and the PE consumption in terms of memory and CPU cycles, hover over the operator, as shown in Figure 6-26. You also can view Tuples Flow information, such as the number of tuples that are flowing in and out of the operator.

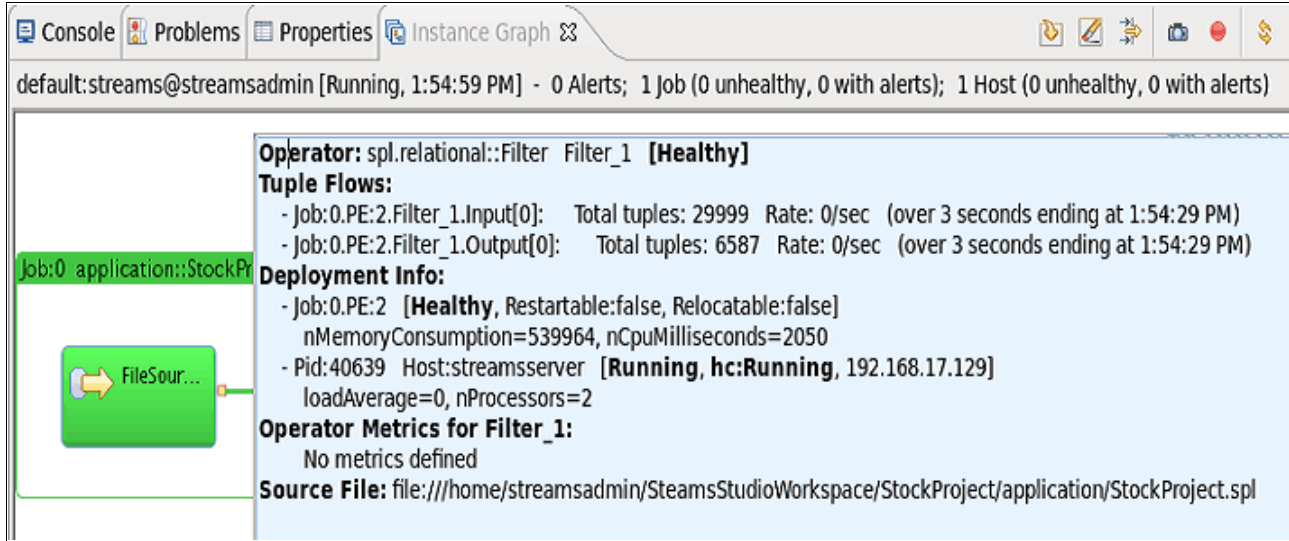


Figure 6-26 Instance Graph Operator details

By default, the Health scheme is the default schema that is selected when the Instance Graph is opened, as shown in Figure 6-27.

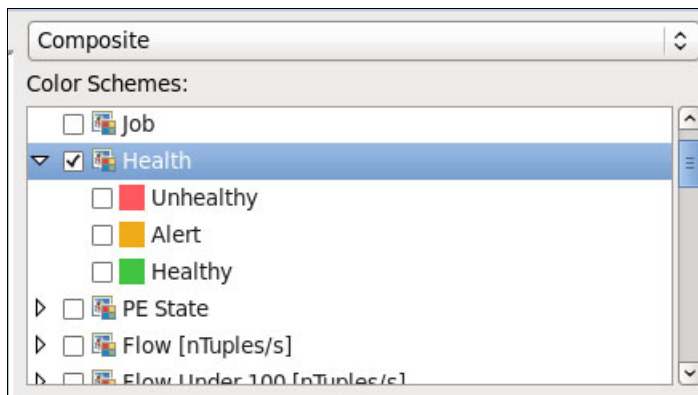


Figure 6-27 Color schema

You can expand the Health scheme to see the color assignments. The Instance Graph view colors each operator in your job that is based on the health of the operator. The Health scheme defines the following health indicators:

- ▶ Green: The operator is healthy.
- ▶ Yellow: One or more operator metrics generated an alert.
- ▶ Red: The operator contains errors and is unhealthy.

When you submit a job, the initial state of the operators is red because the PEs can take a few seconds to start and establish connections. Figure 6-28 shows that the application was running for only a few seconds. All of the operators are marked in red because the job was just submitted.

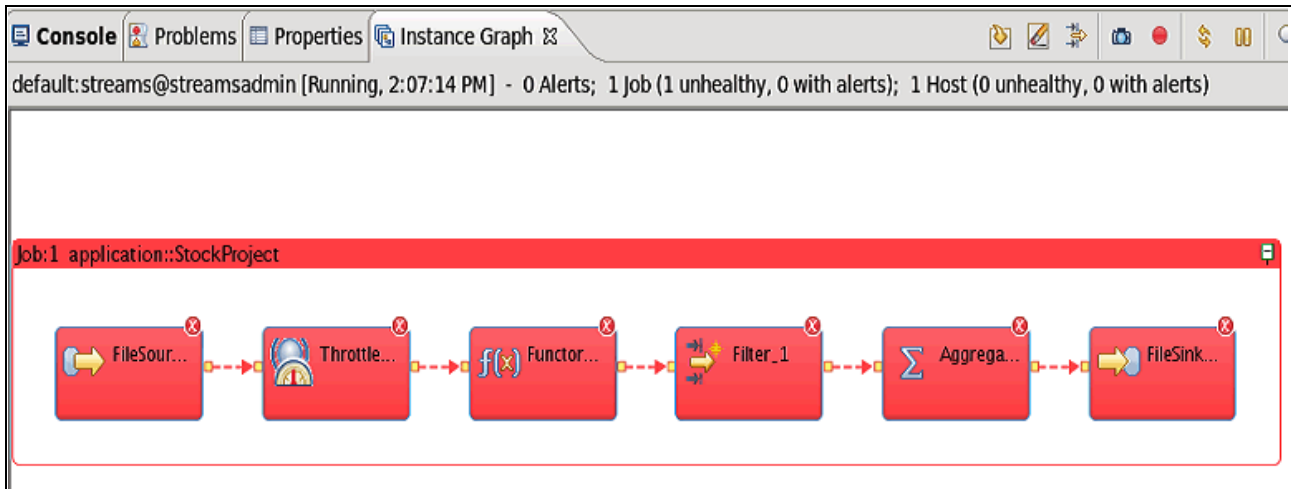


Figure 6-28 Instance Graph after job was submitted

After few more seconds, the PEs start to work and the operators start to take other colors that are based on their health, as shown in Figure 6-29.

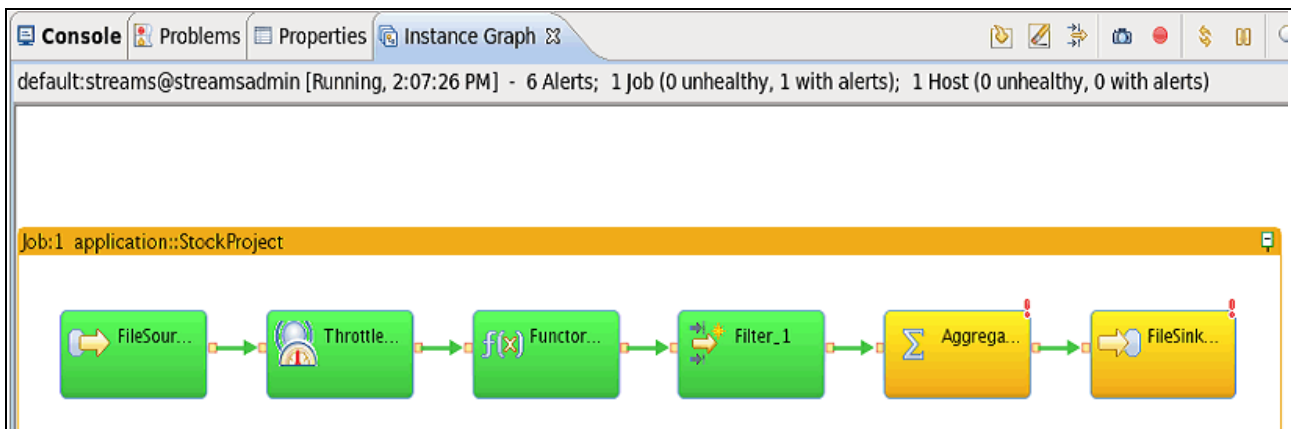


Figure 6-29 Instance Graph of the StockProject

Metrics are counters that are maintained by InfoSphere Streams run time to monitor interesting statistics. Alerts indicate that a metric value is outside an expected range. This condition might or might not be a problem, depending on what your application is doing. Stream Studio includes a predefined range for system-level, operator-specific, and PE-specific metric values. You can view or change these settings on the Preferences page of the InfoSphere Streams. For more information, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/>

An alert indicates that one of the metric values of the element is outside the expected range. You can view the details or investigate the alerts by using the hover help or the highlighting options.

In Figure 6-29, the Aggregate and FileSink operators indicate an alert. When you hover over the Aggregate operator, it indicates that no tuples are processed, as shown in Figure 6-30. However, because our application is designed to include the window parameter of the Aggregate operator as tumbling and the Eviction policy as on punctuation, it is logical that the operator works only after the upcoming stream finishes collecting all tuples and gets the punctuation. Therefore, all of the operators after the Filter operator feature this alert, which explains why the FileSink also includes the same alert. After investigating, it appears that this alert is normal and there are no errors in the application.

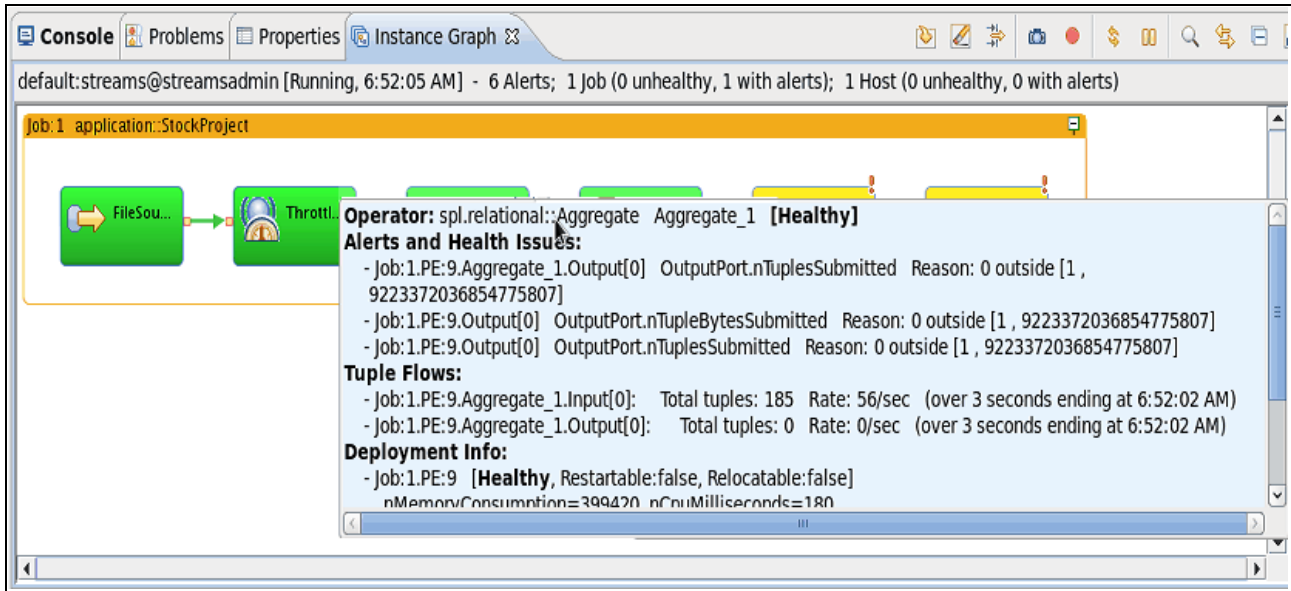


Figure 6-30 Alert in Instance Graph

A significant feature that was added in Streams 3.0 for developers is the ability to see the data flow at any time while the application is running. This feature can be seen by right-clicking a connection or operator output port then clicking **Show Data**. Figure 6-31 shows the data flow from the output port of the filter operator.

The screenshot shows the Properties pane with the 'Output[0]::Filter\_1\_out0' selected. The data is presented in a table with the following columns: Time, ticker, ttype, price, count, minimum, and maximum.

General	Time	ticker	ttype	price	count	minimum	maximum
<b>Data</b>	12/17/12 4:13:15 AM PST	IBM	Trade	83.6	0	0	0
<b>Metrics</b>	12/17/12 4:13:11 AM PST	IBM	Trade	83.6	0	0	0
	12/17/12 4:13:07 AM PST	IBM	Trade	83.6	0	0	0
	12/17/12 4:13:06 AM PST	IBM	Trade	83.59	0	0	0
	12/17/12 4:13:05 AM PST	IBM	Trade	83.57	0	0	0
	12/17/12 4:13:02 AM PST	IBM	Trade	83.57	0	0	0
	12/17/12 4:13:00 AM PST	IBM	Trade	83.57	0	0	0
	12/17/12 4:12:47 AM PST	IBM	Trade	83.59	0	0	0
	12/17/12 4:12:45 AM PST	IBM	Trade	83.58	0	0	0
	12/17/12 4:12:44 AM PST	IBM	Trade	83.59	0	0	0

Figure 6-31 Data flow at the output port of the filter operator

To assign a unique color to each PE, from the Color Schemes pane, select **PE**. All operators that are running in the same PE are now displayed in the same color as the PE.

If you select **Flow** in the Color Schemes pane, you can see that each operator in your job is colored that is based on the flow rate and calculated that is based on the tuples that are flowing through the operator per second. The line thickness of the streams also changes with the flow rates, where a thicker line represents heavier flow.

If the operators are unhealthy, log and trace files are usually the best starting place to help you determine why your job is unhealthy or why you are not getting the results that you expect. Complete the following steps to retrieve and view log and trace data in the Instance Graph view to monitor and diagnose problems with Streams applications:

1. In the Streams Explorer view, right-click an instance, and then click **Set Log and Trace levels**.
2. In the Application and Runtime Logging Level area, set the log level for all runtime services and applications, and then click **OK**.
3. In the Instance Graph view, right-click an operator and select any of the **Show PE** or **Show Operator** options to view log or trace data in the Console view.
4. To retrieve logs for a job, right-click the job (Main composite operator) and click **Get Job Logs**. Streams Studio downloads and saves the logs in the StreamsStudioLogs Eclipse project.

## 6.3 Streams Processing Language

InfoSphere Streams V3.0 features a remarkable enhancement for developing applications. As we see in 6.2.3, “Creating a simple application with Streams Studio” on page 114, we can develop end-to-end applications by using the new Streams Graphical Editor in InfoSphere Studio by using only drag application composition features. However, the underlying layer of the application is still the Streams Processing Language (SPL) source file. It is preferable to have a certain level of knowledge about SPL language. In this section, we give a high-level overview of SPL.

SPL is a powerful declarative language that provides a high level of abstraction to build Streams applications. SPL files (source files) are ASCII text files that represent graph-like what was described in the example in 6.2.3, “Creating a simple application with Streams Studio” on page 114.

As text files, the SPL files can be written and edited by using any text editor. However, they are most often developed and maintained by using the InfoSphere Streams Studio editor. The source code files often feature a `.spl` extension. The text describes the flow from inputs (sources) to outputs (sinks) through operators and describes the inner connections between them. The direction of the flow is always from left to right.

SPL files are compiled by using the Streams compiler to binary executable code (bin), which runs in the InfoSphere Streams environment to run tasks on the various hosts of the Streams instance. SPL application consists of one or more composite operators. A composite operator consists of a reusable and configurable Streams subgraph. Technically, all Streams applications contain at least one composite (the main composite for the application), but they can include more than one composite (composites also can be nested). A composite defines zero or more input streams and zero or more output streams. Streams can be passed to the inputs of the composite and are connected to inputs in the internal subgraph. Outputs from the internal subgraph are similarly connected to the composite outputs. A composite can show the parameters that are used to customize its behavior. Example 6-3, “StockProject composite” on page 124 is an example of a composite.



## 6.3.1 Structure of an SPL program file

Each source file consists of up to six sections, of which two are mandatory. The following sections are headed by using square braces:

- ▶ Namespace
- ▶ Use
- ▶ Composite
- ▶ Type
- ▶ Graph
- ▶ Config

The composite and graph sections are mandatory. As such, they are described together. The other sections are optional and are described separately.

### Composite and Graph

A simple .spl file contains the mandatory sections composite and graph. Following the composite keyword is the name of the application, which is followed by a keyword that indicates the beginning of the application subgraph. Following the keyword of graph is the body of the code.

### Namespace

At the top of the .spl program file, the developer can specify an optional namespace keyword, which is followed by a namespace for the composite, as shown in Example 6-6. The presence of a namespace allows the application developers to avoid collisions between two different SPL composites that carry the same name.

*Example 6-6 Namespace*

---

```
namespace com.ibm.redbook.stockApplication ;
```

---

### Use

Following the optional namespace keyword, developers can optionally declare other composites and types that are used within this .spl file. The Use section allows the developers to improve the code readability by not expanding the fully qualified name for the external types and composites that are used in the .spl file. As an example, a developer could define a Use statement to indicate that an external composite is used later in the code section of this .spl file, as shown in Example 6-7.

*Example 6-7 Use*

---

```
use  
com.ibm.streams.myUtils::FileWriter;
```

---

### Type

Inside a composite, developers can optionally declare all the different data types that are used in the SPL application. In this section, user-defined types, constants, and full-fledged tuple types can all be defined, as shown in Example 6-8.

*Example 6-8 Type*

---

```
type reducedTuple = rstring ticker, rstring ttype, decimal64 price, int32 count,  
                    decimal64 minimum, decimal64 maximum ;
```

---

## Config

The SPL language allows an extensive set of configuration directives that can be applied at the individual operator level or at the composite level. These configuration directives can feature the following functionality:

- ▶ Assigning a default node pool size
- ▶ Providing a list of host pools
- ▶ Setting the level of logging
- ▶ Determining whether a PE should be restarted
- ▶ Establishing a threaded port for an input port with an optional queue size
- ▶ Fusing multiple operators,
- ▶ Co-locating multiple operators on a specific host
- ▶ Ensuring two operators are not placed on a same host

Some of these SPL configuration directives are shown in Example 6-9.

*Example 6-9 Using SPL configuration directives*

---

```
config
  logLevel: info; // Log level verbosity increases in this order':
error, info, debug, trace
  placement: host("myhost.acme.com");
  restartable: true;
  relocatable: true;
  threadedPort: queue(Port1, Sys.Wait, 1500);
```

---

## 6.3.2 Streams data types

In this section, we describe the data types that are supported by Streams.

A stream has an associated schema, which is the particular set of data types that it contains. A schema consists of one or more attributes. The type of each attribute in the schema can be selected from one of the Primitive types or the Composite types categories.

### Primitive types

The following primitive types are supported:

- ▶ `boolean`: True or false
- ▶ `enum`: User-defined enumeration of identifiers
- ▶ `intb`: Signed b-bit integer (int8, int16, int32, and int64)
- ▶ `uintb`: Unsigned b-bite integer (uint8, uint16, uint32, and uint64)
- ▶ `floatb`: b-bit floating point number (float32 and float64)
- ▶ `decimalb`: Decimal b-bit floating point number (decimal32, decimal64, and decimal128)
- ▶ `complexb`: b-bit complex number (complex32 and complex64)
- ▶ `timestamp`: Point in time, with nanosecond precision
- ▶ `rstring`: String of raw 8-bit characters
- ▶ `ustring`: String of UTF-16 Unicode characters
- ▶ `blob`: Sequence of raw bytes
- ▶ `string[n]`: Bounded length string of at most n raw 8-bit characters

### Composite types

InfoSphere Streams provides two types of composite types: built-in collection types and tuple types.

The built-in collections can be bounded or unbounded, which makes the following actual types:

- ▶ `list<T>`: A list of random access zero-indexed sequence of SPL types
- ▶ `set<T>`: Unordered collection without duplicates
- ▶ `map<K,V>`: Unordered mapping from key type K to value type V
- ▶ `list<T>[n]`: `list<T>` with a bounded length
- ▶ `set<T>[n]`: `set<T>` with a bounded length
- ▶ `map<K,V>[n]`: `map<K,V>` with a bounded length

Another type of composite is a tuple, which is a sequence of attributes that are also known as named value pairs. For example, the tuple `{name="aname", age=21}` has two attributes, `name="aname"` and `age=21`. The type for this tuple is `tuple<rstring name, uint32 age>`. Tuples are similar to database rows in that they include a sequence of named and typed attributes. Tuples differ from objects in Java or C++ in that they do not have methods. You can access the attributes of a tuple with dot notation. For example, given a tuple `t` of type `tuple<rstring name, uint32 age>`, the expression `t.name` yields the name attribute's value. Attributes within a tuple type are ordered.

### 6.3.3 Stream schemas

The data types can be grouped into a schema. The schema of a stream is the name that is given to the definition of the structure of data that is flowing through that stream.

Within the Streams application code, the schema of a stream can be referred to in several ways.

#### Schema can be explicitly listed

The schema can be fully defined within the operator output stream parameters, as in schema that is fully defined in operator output stream, which shows the `GlobalCalls` schema, as shown in Example 6-10.

*Example 6-10 Schema that is fully defined in the operator output stream*

---

```
stream <int32 callerNumber, int16 callerCountry, rstring startTimestampGMT,
rstring endTimestampGMT, list<uint32> mastIDS> GlobalCalls = TCPSource() {...}
```

---

#### Schema can reference a previously defined type

The schema definition can be created ahead of the operator definitions in the type section of the SPL program file. In the type section, Streams developers can associate a name with that schema and refer to that name within the code rather than the entire definition in full each time, as shown in Example 6-11.

*Example 6-11 Defining the tuple in the type section*

---

```
type callDataTuples = tuple<int32 callerNumber, int16 callerCountry, rstring
startTimestampGMT, rstring endTimestampGMT, list<uint32> mastIDS>;
..
stream <callDataTuples> GlobalCalls = TCPSource(...){...}
```

---

## Schema is defined by using a combination of the two methods

The last variant is that a schema for a stream can be defined by using a combination of the two methods, as shown in Example 6-12. In this example, another attribute is added to the list of attributes that are present.

*Example 6-12 Defining a stream schema by using a combination of methods*

---

```
type callDataTuples = tuple<int32 callerNumber, int16 callerCountry, rstring
startTimeStampGMT, rstring endTimeStampGMT, list<uint32> mastIDS>;
graph
stream <callDataTuples, tuple<int32 anotherAttribute>> = ...
```

---

### 6.3.4 Streams punctuation markers

InfoSphere Streams supports the concept of punctuation marks, which are markers that are inserted into a stream. These markers are designed so that they cannot be misinterpreted by streams as tuple data. Some Streams operators can be triggered to process groups of tuples that are based on punctuation boundaries, or they can insert new punctuation marks into their output stream.

### 6.3.5 Streams windows

Conceptually, we are working on streams of data that have no beginning or end. However, the streams operator's processing might need to group sections of the DataStream within similar attributes or time intervals. The SPL enables this type of grouping by using a feature that is called windows.

Streams can be used by operators either on a tuple-by-tuple basis or through windows that create logical groupings of tuples.

Windows are associated and defined on the input streams flowing into particular operators. If an operator supports windows, each stream input to that operator can have window characteristics that are defined. Not all Streams operators support windows; this support is dependent on the operator's functionality.

Windows include the following characteristics:

- ▶ Type of window
- ▶ Window eviction policy
- ▶ Window trigger policy

## Type of window

Two window types are supported: tumbling windows and sliding windows.

### ***Tumbling windows***

In tumbling windows, after the trigger policy criteria is met, the tuples in the window are collectively processed by the operator and then the entire window contents are discarded, as shown in Figure 6-32 .

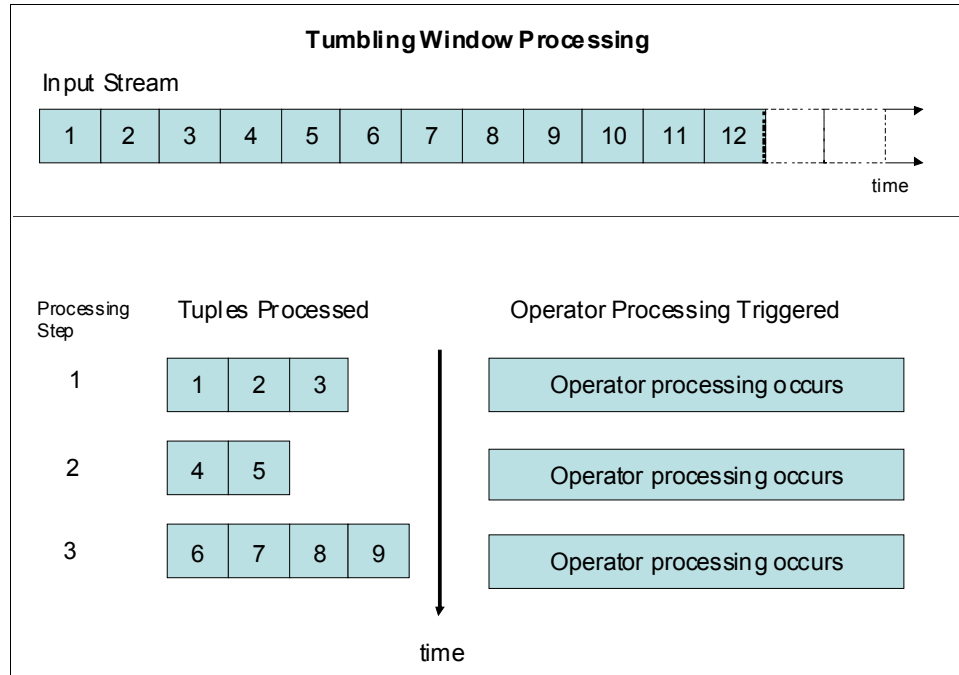


Figure 6-32 Tumbling window

## Sliding windows

In sliding windows, newly arrived tuples cause older tuples to be evicted if the window is full. In this case, the same tuple can be present for multiple processing steps. In contrast to tumbling windows, with sliding windows any tuple may be processed by an operator more than once, as shown in Figure 6-33.

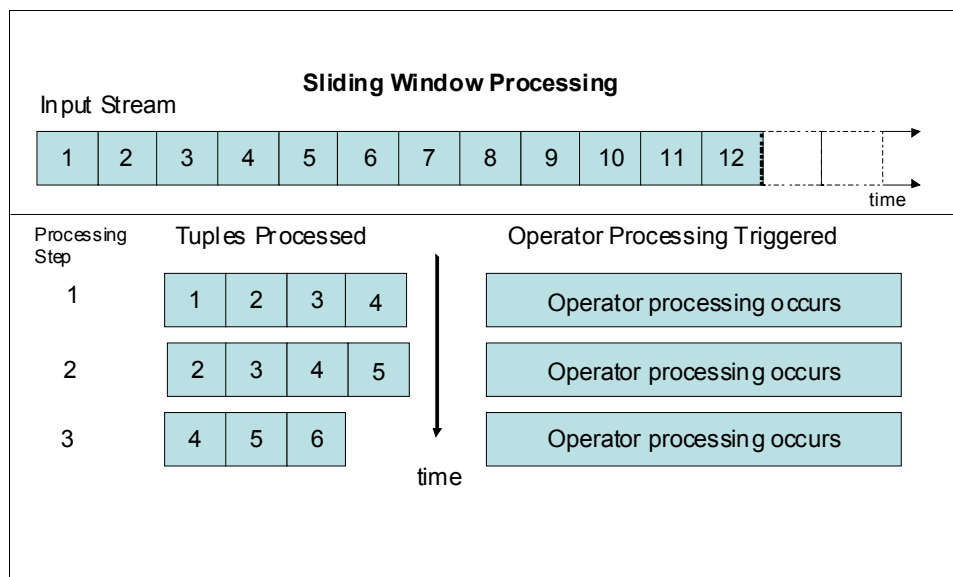


Figure 6-33 Sliding window

## Window eviction policy

The eviction policy defines when tuples are removed from the window. The eviction policy can be specified after a selected numeric attribute increases by an amount beyond a threshold that use the following criteria:

- ▶ Quantity of tuple
- ▶ Age of tuples
- ▶ Punctuation marks (special tuple separators that are understood by Streams)

## Window trigger policy

The window trigger policy defines when the operator that is associated with the window performs its function on the incoming data. For example, a trigger policy can be based on a period or a number of tuples.

## Effect of adding the partitioned keyword on the window

For supported operators, the use of the optional partitioned keyword causes multiple window instances to be created for the stream. The number of instances depends on the number of distinct values for the selected attributes, meaning the runtime values of the attributes that are used to define the group. The number of instances is controlled by the partitioned keyword.

As the application runs and new combinations of attribute key values are received, more window instances are created dynamically. Each window instance on the stream operates independently of each other within the common set of characteristics, as shown in Figure 6-34.

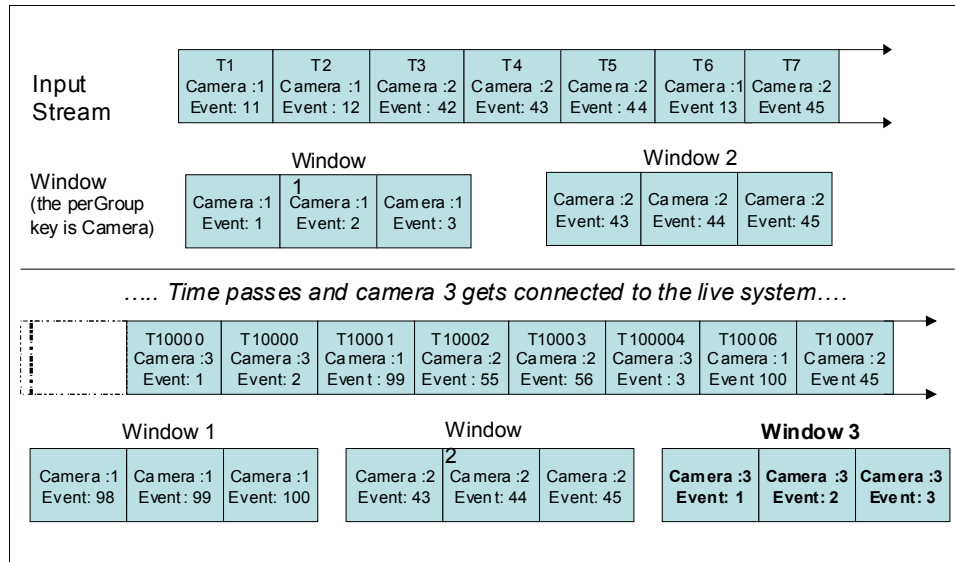


Figure 6-34 Partitioned keyword

For tumbling windows, the eviction policy is explicitly specified (with or without the partitioned keyword) and includes the following options:

- ▶ <punct (>): Punctuation-based tumbling windows
- ▶ <count (>): Number of tuples-based tumbling windows
- ▶ <time (seconds)>: Time-based tumbling windows
- ▶ <delta (attribute-delta)>: Change in a non-decreasing numeric amount

The eviction policy must be explicitly defined for tumbling windows. There is no explicit trigger policy definition in the tumbling windows because the eviction policy always matches the trigger policy. When the tumbling window becomes full, it triggers or performs the action that is intended by that SPL operator and then evicts all stored tuples in the window.

For sliding windows, the eviction and trigger policy must be specified and they must be specified in that order (again, with or without the partitioned keyword). The following choices are available for each policy:

- ▶ <count (>): The number of tuples-based sliding windows
- ▶ <time (seconds)>: The time-based sliding windows
- ▶ <delta (attribute-delta)>: The change in a non-decreasing numeric amount

Excluding partitioned, these choices result in the following possibilities of sliding window definition, which are specified in <evictionPolicy,triggerPolicy> order:

- <count(n),count(n)>
- <count(n),time(seconds)>
- <count(n), delta(attribute,value)>
- <time(seconds),count(n)>
- <time(seconds),time(seconds)>
- <time(seconds,delta(attribute,value)>
- <delta(attribute,value),count(n)>
- <delta(attribute,value),time(seconds)>
- <delta(attribute,value),delta(attribute,value)>

Sliding windows do not support a trigger or eviction policy that is based on punctuation.

## 6.4 InfoSphere Streams operators

An operator is a component of processing functionality that takes one or more streams as input, processes the tuples and attributes, and produces one or more streams as output.

As shown in Figure 6-35, each input port can receive tuples from multiple streams when this process is complete and each of those streams has the same schema. However, each separate input and output port can have different schemas, as required.

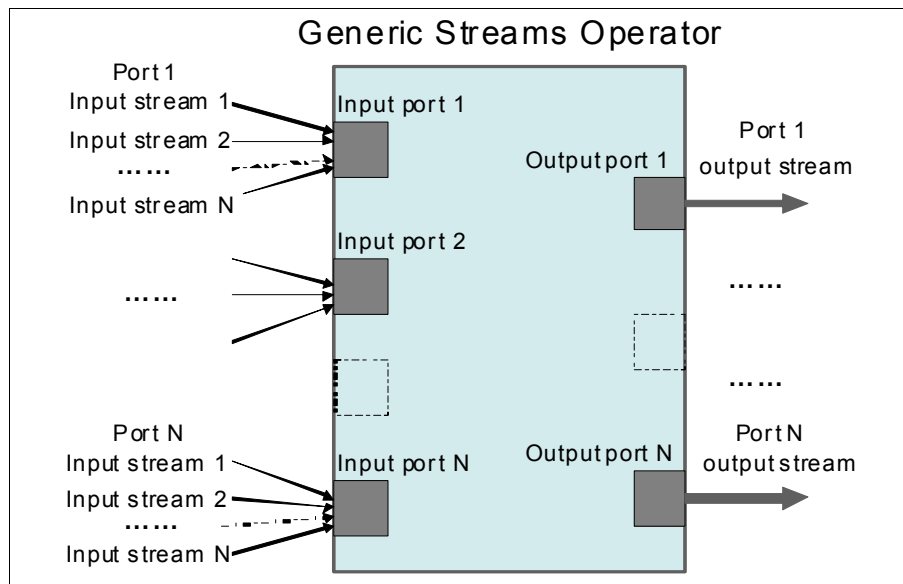


Figure 6-35 Operator inputs/outputs

SPL is supported by many built-in operators in the standard toolkit and other special toolkits, such as data-mining, geo-spatial, and database toolkits. The standard toolkit operators are grouped into the following categories:

- ▶ Adapter operators
- ▶ Relational operators
- ▶ Utility operators
- ▶ XML operators
- ▶ Compact operators



The basic functions of some of these operators are described next. For more information, see the product documentation.

## 6.4.1 Adapter operators

The adapter toolkit includes 10 operators, as shown in Table 6-1. For more information about the adapter operators, see the InfoSphere Streams 3.0 Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/>

Table 6-1 Adapter operators

Operator name	Function	Notes
FileSource	The FileSource operator reads data from a file and produces tuples.	The FileSource operator has one optional input port and two output ports (the second output port is optional). The FileSource operator does not accept any window configurations. The file parameter is mandatory if the operator does not have input port.
FileSink	The FileSink operator writes tuples to a file.	The FileSink operator is configurable with a single input port and an optional output port. The FileSink operator does not accept any window configurations. The file parameter is mandatory.
DirectoryScan	The DirectoryScan operator watches a directory and generates file names on the output, one for each file that is found in the directory.	The DirectoryScan operator does not have any input ports and has a single output port. The directory parameter is mandatory.
TCPSource	The TCPSource operator reads data from a TCP socket and creates tuples out of it. It can be configured as a TCP server (listens for a client connection) or as a TCP client (initiates a connection to a server). In both modes, it handles a single connection at a time.	The TCPSource operator does not have any input ports and is configurable with a single output port. The role parameter is mandatory. The TCPSource works with IPv4 and IPv6 addresses.
TCPSink	The TCPSink operator writes data to a TCP socket in the form of tuples. It can be configured as a TCP server (listens for a client connection) or as a TCP client (initiates a connection to a server). In both modes, it handles a single connection at a time.	The TCPSink operator is configurable with a single input port and does not have any output ports.  The TCPSink operator does not accept any window configurations.
UDPSource	The UDPSource operator reads data from a UDP socket and creates tuples (and punctuations) out of it. Each tuple must fit into a single UDP packet and a single UDP packet must contain only a single tuple or punctuation.	The UDPSource operator does not have any input ports and is configurable with only single output.  The UDPSource operator does not accept any window configurations.

Operator name	Function	Notes
UDPSink	The UDPSink operator writes data to a UDP socket in the form of tuples (and punctuations). Each tuple must fit into a single UDP packet and a single UDP packet contains only a single tuple or punctuation.	<p>The UDPSink operator is configurable with a single input port and does not have any output ports.</p> <p>The UDPSink operator does not accept any window configurations.</p>
Export	The Export operator sends a stream from the current application, making it available to Import operators of applications that are running in the same streaming middleware instance. The Export operator does not send tuples to the exported stream inside standalone SPL applications. The tuples are ignored.	The Export operator does not allow a config clause to be specified. The Export operator does not accept windows on the input port. The Export operator has a single punctuation-oblivious, non-mutating input port. and does not have any output ports.
Import	The Import operator receives tuples from streams that are made available by Export operators of applications that are running in the same streaming middleware instance. Import streams are matched to Export streams. The match can be done by subscription or properties or by streamId name. The Import operator does not generate tuples inside standalone SPL applications.	The Import operator does not allow a config clause to be specified. The Import operator does not have any input ports. The Import operator has a single, non-mutating, punctuation-preserving output port. The Import operator does not accept any window configurations.
MetricsSink	The MetricsSink operator creates Custom Operator Metrics and updates them with values when a tuple is received. These metrics can be viewed by using Streams Studio and the streamtool capturestate command.	The MetricsSink operator is configurable with a single input port. The MetricsSink operator does not have any output ports. Metrics is a mandatory parameter that specifies a list of int64 expressions that are used to set the values of Custom Operator Metrics.

## 6.4.2 Relational operators

The relational toolkit includes the six operators that are described in Table 6-2. For more information about the relational operators, see InfoSphere Streams 3.0 Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/>

Table 6-2 Relational operators

Operator name	Function	Notes
Filter	The Filter operator removes tuples from a stream by passing along only those tuples that satisfy a user-specified condition. Non-matching tuples might be sent to a second optional output.	The Filter operator is configurable with a single input port. The Filter operator is configurable with one or two output ports. The first output port is mandatory, but the second port is optional and has only the filter parameter. The Filter operator does not accept any window configurations.
Functor	The Functor operator is used to transform input tuples into output tuples, and optionally filter them as in a Filter operator.  If you do not filter an input tuple, any incoming tuple results in a tuple on each output port.	The Functor operator is configurable with one or more output ports. The Functor operator has only the filter parameter. The Functor operator does not accept any window configurations.
Punctor	The Punctor operator is used to transform input tuples into output tuples and add window punctuations to the output.	The Punctor operator is configurable with a single input port.
Sort	The Sort operator is used to order tuples that are based on user-specified ordering expressions and window configurations.	The Sort operator is configurable with a single input port and a single output. The sortBy parameter is mandatory. The operator supports tumbling and sliding windows.
Join	The Join operator is used to correlate tuples from two streams that are based on user-specified match predicates and window configurations.	The Join operator is configurable with two input ports and a single output port. The operator supports the sliding window.  When a tuple is received on an input port, it is inserted into the window that corresponds to the input port, which causes the window to trigger. As part of the trigger processing, the tuple is compared against all tuples inside the window of the opposing input port. If the tuples match, an output tuple is produced for each match. If at least one output was generated, a window punctuation is generated after all the outputs.

Operator name	Function	Notes
Aggregate	The Aggregate operator is used to compute user-specified aggregations over tuples that are gathered in a window.	The Aggregate operator is configurable with a single input port and a single output port. The Aggregate operator supports tumbling and sliding windows.

### 6.4.3 Utility operators

The utility toolkit includes the operators that are described in Table 6-3. For more information about the utility operators, see the InfoSphere Streams 3.0 Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/>

Table 6-3 Utility operators

Operator name	Function	Notes
Custom	The Custom operator is a special logic-related operator that can receive and send any number of streams and does not do anything by itself. Thus, it can be customized.	The Custom operator can have zero or more input ports and zero or more output ports. The Custom operator can submit tuples from within its onTuple, onPunct, and onProcess clauses. It requires special support from the compiler. The submit intrinsic function is only available in the Custom operator. The Custom operator does not accept any window configurations and does not require any parameters.
Beacon	The Beacon operator is a utility source that generates tuples on-the-fly.	The Beacon operator does not have any input ports and has only one output. The Beacon operator outputs a window marker punctuation when it finishes generating tuples. The Beacon operator does not accept any window configurations.
Throttle	The Throttle operator is used to pace a stream to make it flow at a specified rate.	The Throttle operator is configurable with a single input port and a single output port. The rate parameter is mandatory and it specifies the wanted rate in tuples per seconds. It is of type float64. The Throttle operator does not accept any window configurations.
Delay	The Delay operator is used to delay a stream by a fixed amount while keeping the inter-arrival times of tuples and punctuations intact.	The Delay operator is configurable with a single input port and a single output port. This delay parameter is mandatory of type float64. It specifies that the delay is to be introduced in seconds. The Delay operator does not accept any window configurations.
Barrier	The Barrier operator is used to synchronize tuples from two or more streams.	Corresponding tuples from each input port are used to create an output tuple. The Barrier operator creates and sends this output tuple only when all the tuples from the input ports are received. The Barrier operator is configurable with two or more input ports and one single port. The Barrier operator does not accept any window configurations.

Operator name	Function	Notes
Pair	The Pair operator is used to pair tuples from two or more streams.	The Pair operator is configurable with two or more input ports and one output. It operates exactly like the Barrier operator, with one major difference: Rather than combining the input tuples into one output tuple, it outputs them individually by using the order of the input ports, followed by a window marker. The Pair operator does not accept any window configurations.
Split	The Split operator is used to split a stream into one or more output streams, which are based on a user-specified split condition.	The Split operator is configurable with a single input port and one or more output ports. The Split operator does not accept any window configurations.
DeDuplicate	The DeDuplicate operator suppresses duplicate tuples that are seen within a given timeOut, count, or delta.	The DeDuplicate operator has one input port and one or two output ports. The DeDuplicate operator does not accept any window configurations.
Union	The Union operator combines tuples from streams that are connected to different input ports.	The Union operator has two or more input ports and one output port. The Union operator does not have any parameters and does not accept any window configurations.
ThreadedSplit	The ThreadedSplit operator splits tuples across multiple output ports to improve concurrency.	The ThreadedSplit operator is not a direct replacement for the Split operator. The ThreadedSplit operator does not allow choosing to which stream a tuple is sent. The ThreadedSplit operator has one input port and one or more output ports. The bufferSize is a mandatory field. The ThreadedSplit operator does not accept any window configurations.
DynamicFilter	The DynamicFilter is a version of the Filter operator that can decide at run time which input tuples are passed through, based on control input it receives.	The DynamicFilter operator has three input ports (the third port is optional) and one or two output ports. The key and addKey are mandatory parameters. The tuples on the first port are passed through the DynamicFilter if their key matches a valid key within the DynamicFilter operator. The second port is a stream of tuples that contain expressions to be added to the valid keys in the DynamicFilter operator. The third port is a stream of tuples that contain expressions to be removed from the valid keys in the DynamicFilter operator. The DynamicFilter operator does not accept any window configurations.
Gate	The Gate operator is used to control the rate at which tuples are passed through. Each tuple that passes through Gate must be acknowledged by a downstream operator to allow more tuples through.	The Gate operator has two input ports and a single output port. The first port is the data port and holds the tuples to be gated. The second input port is the control port, and is used to acknowledge data tuples. The maxUnackedTupleCount parameter is mandatory.
JavaOp	The JavaOp operator is used to call out to operators that are implemented in Java by using the Java Operator API.	The JavaOp operator is configurable with zero or more input ports and zero or more output ports. The JavaOp operator has both className and classLibrary as mandatory parameters. The JavaOp operator accepts any window configuration, including not windowed.

Operator name	Function	Notes
Switch	The Switch operator is used to temporarily stop tuples from flowing. The Switch is open (tuples are blocked) or closed (tuples are flowing).	The Switch operator has two input ports and single output port. The status parameter is mandatory. The Switch operator can be used to hold tuples until a downstream operator is ready to process them. This operator is available in InfoSphere Streams Version 3.0 and later.
Parse	The Parse operator parses the input data from a file or network connection and generates SPL tuples from that data. The Parse operator accepts data in many formats (such as line or bin); therefore, the data is passed in by using a blob attribute. The Parse operator generates the SPL tuples that correspond to the input format.	The Parse operator is configurable with a single input port and a single output port. The Parse operator does not accept any window configurations. This operator is available in InfoSphere Streams Version 3.0 and later.
Format	The Format operator converts SPL tuples into formatted data. The Format operator writes data in many formats (such as line or bin); therefore, the output of the Format operator is generated as a blob attribute.	The Format operator is configurable with a single input port and a single output port. The Format operator does not accept any window configurations. This operator is available in InfoSphere Streams Version 3.0 and later.
Compress	The Compress operator is used to compress data in a blob and generate blob output.	The Compress operator is configurable with a single input port and a single output port. The compression parameter is mandatory. The Compress operator does not accept any window configurations. This operator is available in InfoSphere Streams Version 3.0 and later.
Decompress	The Decompress operator is used to decompress data in blob and generate blob output.	The Decompress operator is configurable with a single input port and a single output port. The compression parameter is mandatory. The Decompress operator does not accept any window configurations. This operator is available in InfoSphere Streams Version 3.0 and later.
CharacterTransform	The CharacterTransform operator is used to convert from one encoding in blob to another encoding in blob.	The CharacterTransform operator is configurable with a single input port and a single output port. The CharacterTransform operator does not have any parameters. The CharacterTransform operator does not accept any window configurations. This operator is available in InfoSphere Streams Version 3.0 and later.

## 6.4.4 XML operators

The XML operators include the XMLParse operator only, which is described in Table 6-4. For more information about the XML operators, see InfoSphere Streams 3.0 Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/>

Table 6-4 XML parse

Operator name	Function	Notes
XMLParse	The XMLParse operator accepts a single input stream and generates tuples.	The XMLParse operator has one input port and one or more output ports. The trigger parameter is mandatory. The XMLParse operator does not accept any window configurations. This operator is available in InfoSphere Streams Version 3.0 and later.

## 6.4.5 Compact operators

### SPADE operators

The Stream Processing Application Declarative Engine (SPADE) language was introduced with the original release of IBM InfoSphere Streams. InfoSphere Streams V2.0 introduced a new SPL and a new implementation of operators. Users of InfoSphere Streams V1.2 systems might want to perform new development in SPL on a V2 Streams system, while leaving older code on a V1.2 Streams system. In some cases, users might want to pass tuples from V1.2 to V2 or V2 to V1.2. This pass can be done for a subset of SPADE data types that are supported by SPL by connecting the two systems by using TCP/IP connection. This connection uses two SPL operators (V1TCPSink and V1TCPSource) that know how to access V1.2 SPADE tuples. The SPL operators are connected to SPADE TCP Source/Sink operators to pass the tuples between the systems.

Not all SPADE types are supported by SPL. Therefore, they cannot be passed between SPL and SPADE.

### Compact sample

The KeyedAggregate, KeyedDelay, KeyedJoin, Normalize, and PunctMerger operators were ported from the SPADE sample operators for developers who used these operators. However, these operators are deprecated and are not available in the next release of InfoSphere Streams. Often we use these operators only as a temporary solution for porting applications that relied on those samples from SPADE.

## 6.5 InfoSphere Streams toolkits

In this section, we introduce one of the strengths of the IBM InfoSphere Streams platform, that is, various toolkits that accelerate development of Streams applications. We present here major toolkits that are available with InfoSphere Streams Version 3.0, which is the current release of InfoSphere Streams as of this writing. We also describe how you can find information for these toolkits.

In the context of the InfoSphere Streams platform, toolkits are a collection of assets that facilitate the development of a solution for a particular industry or functionality. The primary goal of InfoSphere Streams is not to get you to think about new ways of doing things that can make a difference. Instead, it is about enabling you to do those things. This component of the Streams platform is focused on enabling you to achieve that goal. These toolkits are provided to give the new InfoSphere Streams developer working examples of what the language can do and how to program a working application to accomplish it. The toolkits provide accelerated development in a particular application or industry.

### 6.5.1 Mining toolkit

Data mining was a valuable analytical technique for decades. The process involves extracting relevant information or intelligence from large data sets that are based on such things as patterns of behavior or relationships between data or events to predict, react, or prevent future actions or behavior. Accumulating the amount of data that is needed to perform meaningful data mining meant that most data mining traditionally was performed on stored historical data.

This toolkit enables scoring of real-time data in a Streams application. The goal is to use the data mining models that you are using against data at rest to do the same analysis against real-time data. The scoring in the Streams Mining Toolkit assumes and uses a predefined model. Various model types and scoring algorithms are supported. Models are represented by using the Predictive Model Markup Language (PMML) (a standard XML representation) for statistical and data mining models.

Supported mining algorithms and PMML version as of this writing are listed in Table 6-5.

*Table 6-5 Operators that are supported in Mining toolkits*

Operator Name	Algorithm	Supported PMML versions
Classification	Decision Tree	2.0-3.0
	Logistic Regression	2.0-3.0
	Naive Bayes	2.0-3.0
Regression	Linear Regression	2.0-3.0
	Polynomial Regression	2.0-3.0
	Transform Regression	2.0-3.0
Clustering	Demographic Clustering	2.0-3.0
	Kohonen Clustering	2.0-3.0
Associations	Association Rules	2.0-3.0



A user starts by building, testing, and training a model of a stored data set by using modeling software (such as SPSS, InfoSphere Warehouse, and SAS). After one or more models are created, the user can export models as PMML statements. The user can then incorporate the scoring of real-time data streams against this model into any Streams application through the scoring operators in the Mining Toolkit.

At run time, real-time data is scored against the PMML model. The data streams flow into the scoring operators and the results of the scoring flow out of the operator as a stream.

## 6.5.2 Financial toolkit

Financial Markets long struggled with vast volumes of data and the need to make key decisions quickly. To address this situation, automated trading solutions were discussed and designed in several forms. The challenge is to be able to use information from widely different sources (from a speed and format perspective) that might hold the definitive keys to making better and profitable trades.

InfoSphere Streams offers a platform that can accommodate the various sources of information and deliver decisions at the low latency that automated trading requires.

The combination of the ability to use the various sources and deliver results in real time is not only attractive in financial markets, but also might provide insight into how to design an application for other industries that have similar needs.

The toolkit includes the following operators to support adapters for Financial Information Exchange (FIX):

- ▶ FIXInitiator operator
- ▶ FIXAcceptor operator
- ▶ FIXMessageToStream operator
- ▶ StreamToFixMessage operator

The toolkit also supports IBM WebSphere Front Office for Financial Markets (WFO) adapters with the following operators:

- ▶ WFOSource operator
- ▶ WFOSink operator

For messaging, the toolkit includes an operator to support the IBM WebSphere MQ Low Latency Messaging adapter: MQRMMSink operator.

The toolkit also provides adapters for Simulated Market Feeds (Equity Trades and Quotes, Option Trades) with the following operators:

- ▶ EquityMarketTradeFeed operator
- ▶ EquityMarketQuoteFeed operator
- ▶ OrderExecutor operator

In the Function layer, this toolkit includes the following analytic functions and operators:

- ▶ Analytical Functions:
  - Coefficient of Correlation
  - The Greeks (Delta, Theta, Rho, Charm, DualDelta, and so on):
    - For Put and Call values
    - For general values
- ▶ Operators:
  - Wrapping QuantLib financial analytics open source package.
  - Provides operators for equity pricing and rating:
    - TrailingPriceStatsCalculator operator, which computes the volume-weighted average price of equities, over a range of the equity's three most-recent trading prices.
    - OpportunityRater operator (Identifies opportunities).
    - VWAPDeltaAggressive operator. (This operator and the following operator examine opportunities and determines whether to generate an order for different criteria.).
    - VWAPDeltaConservative operator.
  - Provides operators to compute theoretical value of an option:
    - EuropeanOptionValue operator, which provides access to 11 different analytic pricing engines, such as Black Scholes, Integral, Finite Differences, Binomial, and Monte Carlo.
    - AmericanOptionValue operator, which provides access to 11 different analytic pricing engines, such as Barone Adesi Whaley, Bjerksund Stensland, and Additive Equiprobabilities.

The Solution Framework layer provides the following extensive example applications:

- ▶ Equities Trading
- ▶ Options Trading

These examples often are called *white box applications* because customers can (and often do) modify and extend them. These example applications are modular in design with pluggable and replaceable components that you can extend or replace. This modular design allows these applications to demonstrate how trading strategies can be swapped out at run time without stopping the rest of the application.

The Equities Trading starter application includes the following modules:

- ▶ TradingStrategy module: Looks for opportunities that have specific quality values and trends.
- ▶ OpportunityFinder module: Looks for opportunities and computes quality metrics.
- ▶ SimpleVWAPCalculator module: Computes a running volume-weighted average price metric.

The Options Trading starter application includes the following modules:

- ▶ Data sources module: Uses incoming data and formats and maps for later use.
- ▶ Pricing module: Computes theoretical put and call values.
- ▶ Decision module: Matches theoretical values against incoming market values to identify buying opportunities.

These starter applications give the application developer good examples and a great foundation to start building their own applications.

### 6.5.3 Database toolkit

Streams applications often process streams of data that are flowing from external data sources and can convert processed streams to external formats to be used by components that are not part of Streams. Much of the data is stored in high-level data-at-rest stores (traditional databases), with higher-level interfaces than the interfaces that are provided by the standard toolkit (which cover file and network interfaces).

The Database Toolkit is provided to facilitate the development of applications that need to interact with traditional databases.

Streams application might need to interact with such traditional data sources for various reasons. The most common use case is to write the streaming data (after processing) into databases for future use, but there might be other reasons, such as check-pointing the state of the application. In this scenario, a Streams application might also need to read data from such databases. In other cases, Streams applications can merge data from external repositories with internal streams, which enriches their content.

The Streams integration approach is devoted to integration with data stores. The focus of that section of the toolkit is user-centric. It walks a user through the steps that are needed to set up a database so that a Streams application can interact with it.

### 6.5.4 Internet toolkit

Streams applications might need to process streams of data that are flowing from external data sources that reside on the Internet and are made available by using the standard HTTP/FTP mechanism. For RSS data feeds, the generated stream is the RSS XML data. The Internet toolkit facilitates the interaction of Streams applications with Internet sources.

Furthermore, the application developer might be interested in not only reading the source once, but continually obtaining updates from the source. In other cases, the source might be an RSS feed to which the application subscribes. This mechanism could be used to read from files, especially if you are interested in updates to the file.

The Internet toolkit provides the `InetSource` operator that receives text-based data from remote sources that use HTTP and generates an input stream from this content. `InetSource` generalizes the functionality of the `File/TCP Source` operators that are provided in the SPL Standard toolkit in the following main ways:

- ▶ **Multiple input sources:** `InetSource` operator can take, as input, a URI list, which allows it to access all the sources that are listed in sequential order (as opposed to `File/TCP Source` that can read only from one source).
- ▶ **Incremental updates:** `InetSource` operator can be configured to periodically check for, and fetch updates from, the sources in its URI list, and stream the whole content or only the updates. It also can be configured to not stream the initial contents or to stream the whole content periodically even if there are no updates.

Furthermore, the `InetSource` operator is significantly more configurable than the `TCP/File Source` operators. For example, the operator can be configured to treat `k` lines of input as a given record, where `k` is a parameter. It can be configured to emit tuples per record, per URI, or per fetch. Not only does this configuration make it a natural interface to connect to remote sources, but in fact can be used to read from local files if the other capabilities are required.

## 6.5.5 Geospatial toolkit

This toolkit provides high-performance analysis and processing of geospatial data. It enables location-based services, geospatial data types (such as point and polygon), and geospatial functions (such as distance between one location to another).

The operators and functions in the Geospatial toolkit facilitate efficient processing and indexing of location data. For example, with map data, you can search for entities in or around an area of interest, map Global Positioning System (GPS) location data to road networks, or calculate spatial relationships between different features on the Earth.

The shape of the Earth closely resembles an oblate ellipsoid, which provides a mathematically tractable way to specify a point on the surface of the Earth. An oblate ellipsoid is obtained by rotating an ellipse around its shorter (minor) axis. For the Earth, the minor axis of the oblate ellipsoid aligns with the North and South poles on the Earth and the major axis lies in the Equatorial plane. Since the Earth is not exactly an ellipsoid, a particular ellipsoid might provide a better fit for one region of the Earth than another ellipsoid. Different countries might use different reference ellipsoids to model the Earth, depending on which model provides the best approximation for a region. With the advent of GPS, however, there is a convergence to the Standard known as WGS84. In IBM InfoSphere Streams Version 3.0, the Geospatial Toolkit supports the WGS84 model of the Earth.

The toolkit supports the following data types:

- ▶ Points in the Geospatial toolkit: Geospatial toolkit contains data types for points and functions to manipulate them.
- ▶ Lines in the Geospatial toolkit: Geospatial toolkit contains data types for line segments and line strings, and functions to manipulate them.
- ▶ Polygons in the Geospatial toolkit: Geospatial toolkit supports simple polygons, which are geometric objects that can be defined by the interior of a closed, non-intersecting path.

The following reference information can be used for the Geospatial Toolkit:

- ▶ Functions in the Geospatial toolkit: Geospatial toolkit provides functions in the `com.ibm.streams.geospatial.conversions` and `com.ibm.streams.geospatial.twodimension.geometry` namespaces.
- ▶ Enumerations and data types in the Geospatial toolkit: Geospatial toolkit defines several types and enumerations.

For more information, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/>

## 6.5.6 TimeSeries toolkit

TimeSeries Toolkit helps you find and determine patterns and anomalies about data streaming. A *time series* is a sequence of numerical data that represents the value of an object or multiple objects over time. For example, you can have time series that contain the monthly electricity usage that is collected from smart meters, daily stock prices and trading volumes, Electrocardiogram (ECG) recordings, seismographs, or network performance records. Time series have a temporal order that is the foundation of all time series analysis algorithms.

You can perform the following main types of processing on time series data in InfoSphere Streams:

- ▶ Data preprocessing involves reading, repairing, and conditioning time series.
- ▶ Data analysis involves investigating the underlying information for the time series. For example, analysis operators calculate statistics and correlations, and decompose and transform time series.
- ▶ Data modeling involves the creation of a model of the time series and the use of that model for prediction or regression.

The toolkit also provides a set of functions that you can use to generate time series for test and validation purposes. A time series can be regular, irregular, univariate, vector, or expanding, depending on the characteristics of the data.

TimeSeries toolkit is organized into the following packages:

- ▶ Data generator functions in the TimeSeries toolkit: Data generator functions generate time series for test and validation.
- ▶ Data preprocessing operators in the TimeSeries toolkit: Data preprocessing operators perform data conditioning, which often occurs before data analysis or modeling.
- ▶ Analysis functions and operators in the TimeSeries toolkit: The analysis functions and operators help you compute various information about time series data, such as statistics, correlations, and components.

For more information, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/>

## 6.5.7 Complex Event Processing toolkit

Complex event processing (CEP) uses patterns to detect composite events in streams of tuples. For example, CEP can be used to detect stock price patterns, routing patterns in transportation applications, or user behavior patterns in web commerce settings. You can perform complex event processing in IBM InfoSphere Streams by using the Complex Event Processing toolkit.

The Complex Event Processing Toolkit provides the MatchRegex operator for SPL. The MatchRegex operator is parameterized with a regular-expression pattern. In addition to event detection, the operator supports aggregation over the matched input tuples to compute the output tuple, which is also known as the *composite event*. Individual input tuples are often referred to as *simple events*.

The Complex Event Processing Toolkit provides a single operator that is called MatchRegex, which matches a regular-expression pattern over the sequence of input tuples to detect composite events.

For more information, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/>

## 6.5.8 Accelerators toolkit

IBM offers analytic components that accelerate the development and implementation of big data solutions. These components, or accelerators, address specific data types or operations, such as analytics for social media data and telecommunications event data.

Accelerators provide business logic, data processing capabilities, and visualization for specific use cases. By using accelerators, you can apply advanced analytics to help integrate and manage the variety, velocity, and volume<sup>1</sup> of data that constantly enters your organization. Accelerators also provide a development environment for building new analytic applications that are tailored to the specific needs of your organization.

### Social media accelerator toolkit

Data from social media forums contains valuable information about user preferences. However, accessing and working with that information requires large-scale import, configuration, and analysis. IBM Accelerator for Social Data Analytics, a set of end-to-end applications, extracts relevant information from tweets, message boards, and blogs and then builds social profiles of users that are based on specific use cases and industries.

With IBM Accelerator for Social Data Analytics, you can perform the following tasks:

- ▶ Import and analyze social media data, identifying user characteristics such as gender, locations, names, and hobbies.
- ▶ Develop comprehensive user profiles across messages and sources.
- ▶ Associate profiles with expressions of sentiment, buzz, intent, and ownership around brands, products, and companies.

After IBM Accelerator for Social Data Analytics is installed and configured, you can customize IBM Accelerator for Social Data Analytics for feedback analysis that is specific to a use case (Brand Management, Lead Generation, or a generic use case) and industry (Retail, Financial, or Media and Entertainment).

### Telco accelerator toolkit

Telecommunications data contains insight into outages and the events that precipitate those outages. Customers in the telecommunications industry face the challenge of performing real-time mediation and analytics on large volumes of Call Detail Records (CDR). By using IBM Accelerator for Telecommunications Event Data Analytics, customers can import and analyze raw telecommunications data in real time, and then transform that data into meaningful and actionable insight.

In addition to a master script that starts, stops, and controls IBM Accelerator for Telecommunications Event Data Analytics, a typical IBM Accelerator for Telecommunications Event Data Analytics workflow consists of the following steps:

1. Import data files (the CDRs).
2. Scan and parse the input files.
3. Extract, enrich, and transform the files.
4. Remove duplicate CDRs.
5. Aggregate the data for statistics or write the CDRs to a repository.

---

<sup>1</sup> For more information, see this website:

<http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>

IBM Accelerator for Telecommunications Event Data Analytics provides a simple rule language that includes several domain-specific features, such as automated type inference, optimized lookup query access, and the generation of optimized SPL code from rules.

IBM Accelerator for Telecommunications Event Data Analytics rules have fairly generic syntax, which is designed for domains where all applicable rules in a rule set are run for each incoming data tuple, and each tuple has exactly the same schema. IBM Accelerator for Telecommunications Event Data Analytics rules cannot maintain any state from one data tuple to the next.

IBM Accelerator for Telecommunications Event Data Analytics features the following capabilities:

- ▶ Reads input files
- ▶ Parses files and extracts CDRs
- ▶ Applies business rules to enrich and transform CDRs
- ▶ Removes duplicate CDRs by using a highly efficient Bloom Filter
- ▶ Writes CDRs to a repository, either an ODBC database or files







## Streams integration considerations

In the previous chapters of this book, we described the architecture and components of the IBM InfoSphere Streams products and applications. In this chapter, we look at how to integrate a Streams application with other current technologies. This information provides the developer with guidance about how to approach integrating Streams with other technologies, such as SPSS, DataStage, XML, WebSphere MQ, and various databases.

## 7.1 Integrating with IBM InfoSphere BigInsights

With the growing use of digital technologies, the volume of data that is generated by our society is exploding into the exabytes (1 terabytes= $10^{12}$ , 1 exabytes= $10^{18}$ ). With the pervasive deployment of sensors to monitor everything from environmental processes to human interactions, the variety of digital data is rapidly encompassing structured, semi-structured, and unstructured data. Finally, with better pipes to carry the data, from wireless to fiber optic networks, the velocity of data is also exploding (from a few kilobits per second to many gigabits per second). We call data with any or all of these characteristics *big data*<sup>1</sup>. Examples include sources such as the internet, web logs, chat, sensor networks, social media, telecommunications call detail records, biological sensor signals (such as EKG and EEG), astronomy, images, audio, medical records, military surveillance, and e-commerce.

With the ability to generate all this valuable data from their systems, businesses and governments are grappling with the problem of analyzing the data for two important purposes: to sense and respond to current events in a timely fashion, and to use predictions from historical learning to guide the response. This situation requires the seamless functioning of data-in-motion (current data) and data-at-rest (historical data) analysis, operating on massive volumes, varieties, and velocities of data. How to bring the seamless processing of current and historical data into operation is a technology challenge that is faced by many businesses that have access to big data.

In this section, we focus on the IBM leading big data products, namely IBM InfoSphere Streams and IBM InfoSphere BigInsights, which are designed to address big data challenges. InfoSphere BigInsights delivers an enterprise-ready big data solution by combining Apache Hadoop, including the MapReduce framework and the Hadoop Distributed File Systems (HDFS), with unique technologies and capabilities from across IBM. Both products are built to run on large-scale distributed systems, which are designed to scale from small to large data volumes, handling both structured and unstructured data analysis. In this first section, we describe various scenarios where data analysis can be performed across the two platforms to address the big data challenges. Streams and real-time scoring by using PMML models

### 7.1.1 Streams and BigInsights application scenario

The integration of data-in-motion (InfoSphere Streams) and data-at-rest (InfoSphere BigInsights) platforms addresses three main application scenarios:

- ▶ Scalable data ingest: Continuous ingest of data through Streams into BigInsights.
- ▶ Bootstrap and enrichment: Historical context that is generated from BigInsights to bootstrap analytics and enrich incoming data on Streams.
- ▶ Adaptive analytics: Models that are generated by analytics such as data-mining, machine-learning, or statistical-modeling in BigInsights used as basis for analytics on incoming data in Streams and updated based on real-time observations.

---

<sup>1</sup> For more information, see this website:

<http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>

These application scenarios are as shown in Figure 7-1.

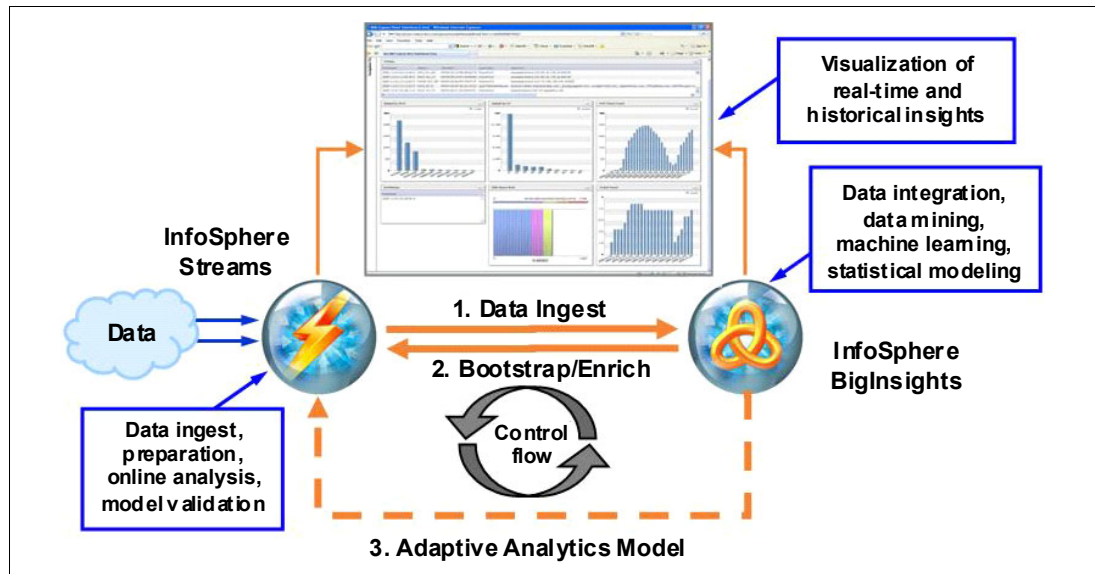


Figure 7-1 Application scenarios with InfoSphere BigInsights and InfoSphere Streams

## 7.1.2 Scalable data ingest

Data from various systems could arrive, as a continuous stream, as a periodic batch of files, or other means. This volume of data must be processed for consumption by downstream applications for further analysis. Data-preparation steps include operations such as data-cleansing, filtering, feature extraction, deduplication, and normalization. These functions are performed on InfoSphere Streams. Data is then stored in BigInsights for deep analysis and also forwarded to downstream analytics on Streams. The parallel pipeline architecture of Streams is used to batch and buffer data and, in parallel, load it into BigInsights for best performance.

In this scenario, we focus on continuous ingest of data through Streams into BigInsights. One of the cases for this scenario is from the telecommunications industry. The telecommunications companies are faced with ever exploding Call Detail Records (CDRs). CDRs come in from the telecommunications network switches periodically as batches of files. Each of these files contains records that pertain to operations such as call initiation and call termination for telephone calls. It is most efficient to remove the duplicate records in this data as it is ingested because duplicate records can be a significant portion of the data that needlessly uses resources if not filtered up front. Additionally, telephone numbers in the CDRs must be normalized and data that is appropriately prepared to be ingested into the back end for analysis.

Another example can be seen in a social media-based lead-generation application. In this application, unstructured text data from sources such as Twitter and Facebook is ingested to extract sentiment and leads of various kinds. In this case, resource savings can be achieved if the text extraction is done on data as it is ingested and irrelevant data such as spam is eliminated. With volumes of 140 million tweets every day and growing, the storage requirements can add up quickly.

### 7.1.3 Bootstrap and enrichment

BigInsights can be used to analyze data over a large time window, which it has assimilated and integrated from various continuous and static data sources. Results from this analysis provide contexts for various online analytics and serves to bootstrap them to a well-known state. They are also used to enrich incoming data with other attributes that are required for downstream analytics.

As an example from the CDR processing use case, an incoming CDR might only list the phone number to which that record pertains. However, a downstream analytic might want access to all phone numbers a person ever used. Attributes from historical data are used to enrich the incoming data to provide all the phone numbers. Similarly, deep analysis results in information about the likelihood that this person might cancel their service. Having this information enables an analytic to offer a promotion online to keep the customer from leaving the network.

In the example of the social media-based application, an incoming Twitter message has only the ID of the person who is posting the message. However, historical data can augment that information with attributes such as influencer, which gives an opportunity for a downstream analytic to treat the sentiment that is expressed by this user appropriately.

### 7.1.4 Adaptive analytics model

Integration of the Streams and BigInsights platforms enables interaction between data-in-motion and data-at-rest analysis. The analysis can use the same analytic capabilities in Streams and BigInsights. It not only includes data flow between the two platforms, but controls flows to enable models to adapt to represent the real-world accurately, as it changes. The following interactions can occur:

- ▶ BigInsights to Streams Control Flow

Deep analysis is performed by using BigInsights to detect patterns on data that is collected over a long period. Statistical analysis algorithms or machine-learning algorithms are compute-intensive and run on the entire historical data set, in many cases it makes multiple passes over the data set to generate models to represent the observations. For example, the deep analysis can build a relationship graph that shows key influencers for products of interest and their relationships. After the model is built, it is used by a corresponding component on Streams to apply the model on the incoming data in a lightweight operation. For example, a relationship graph built offline is updated by analysis on Streams to identify new relationships and influencers that are based on the model, and take appropriate action in real time. In this case, there is control flow from BigInsights to Streams when an updated model is built and an operator on Streams can be configured to pick up the updated model mid-stream and start applying it to new incoming data.

► Streams to BigInsights Control Flow

After the model is created in BigInsights and incorporated into the Streams analysis, operators on Streams continue to observe incoming data to update and validate the model. If the new observations deviate significantly from the expected behavior, the online analytic on Streams might determine that it is time to trigger a new model-building process on BigInsights. This situation represents the scenario where the real-world deviated sufficiently from the model's prediction that a new model must be built. For example, a key influencer that is identified in the model might no longer influence others or an entirely new influencer or relationship can be identified. Where entirely new information of interest is identified, the deep analysis can be targeted to update only the model in relation to that new information; for example, to look for all historical context for this new influencer, where the raw data was stored in BigInsights but not monitored on Streams until now. This situation allows the application to not have to know everything that they are looking for in advance. It can find new information of interest in the incoming data and get the full context from the historical data in BigInsights and adapt its online analysis model with that full context. Here, an extra control flow from Streams to BigInsights in the form of a trigger is required.

### 7.1.5 Streams and BigInsights application development

This section describes how an application developer can create an application that spans two platforms to give timely analytics on data in motion while maintaining full historical data for deep analysis. We describe a simple example application that demonstrates the interactions between Streams and BigInsights. This simple application tracks the positive and negative sentiment that is expressed about products of interest in a stream of emails and tweets. An overview of the application is shown in Figure 7-2.

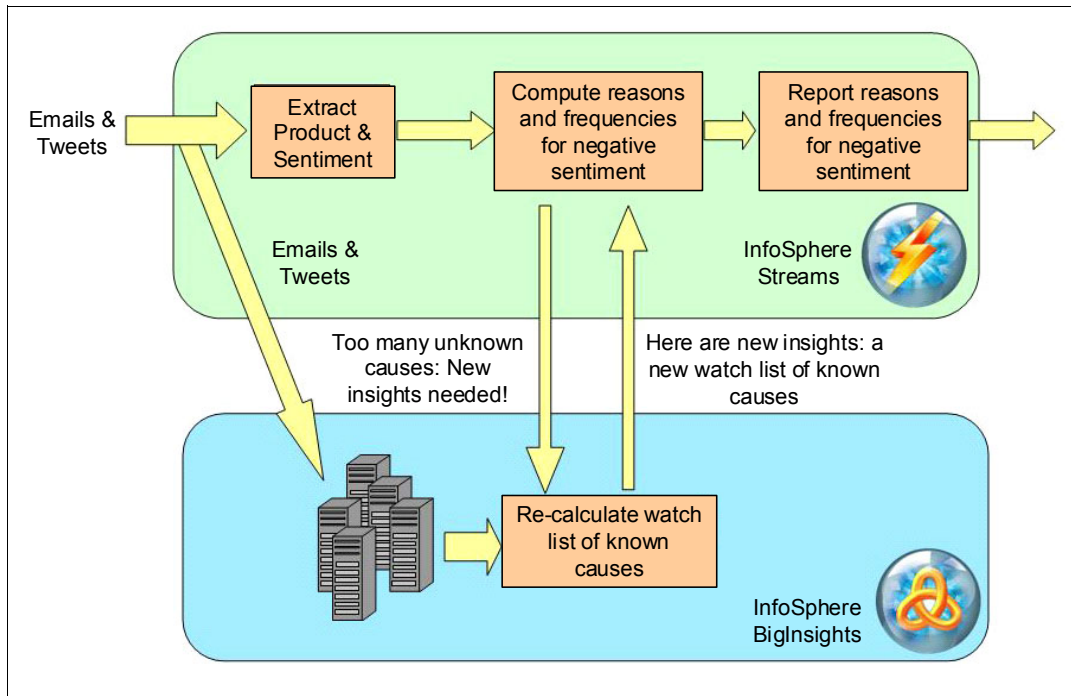


Figure 7-2 Streams and BigInsights application

Each email and tweet on the input streams is analyzed to determine the products that are mentioned and the sentiment that is expressed. The input streams are also ingested into BigInsights for historical storage and deep analysis. Concurrently, the tweets and emails for products of interest are analyzed on Streams to compute the percentage of messages with positive and negative sentiment that is expressed. Messages with negative sentiment are further analyzed to determine the cause of the dissatisfaction that is based on a watch list of known causes. The initial watch list of known causes can be bootstrapped by using the results from the analysis of stored messages on BigInsights. As the stream of new negative sentiment is analyzed, Streams checks if the percentage of negative sentiment includes an unknown cause (not in the watch list of known causes) that became significant. If it finds a significant percentage of the causes are unknown, it requests an update from BigInsights. When requested, BigInsights queries all of its data by using the same sentiment analytics that are used in Streams and recalculates the list of known causes. This new watch list of causes is used by Streams to update the list of causes to be monitored in real time. The application stores all of the information it gathers but monitors only the information currently of interest in real time, which uses resources efficiently.

## 7.1.6 Application interactions

Although this example is simple, it demonstrates the following main interactions between Streams and BigInsights:

- ▶ Data ingest into BigInsights from Streams
- ▶ Streams triggering deep analysis in BigInsights
- ▶ Updating the Streams analytical model from BigInsights.

The implementations of these interactions for this simple demonstration application are described in more detail in the following sections.

### Data ingest into BigInsights from Streams

Streams processes data by using a flow graph of interconnected operators. The data ingest is achieved by using a Streams-BigInsights Sink operator to write to BigInsights (see 7.1.7, “Enabling components” on page 167). The complexities of the BigInsights distributed file system that is used to store data are hidden from the Streams developer by the Streams-BigInsights Sink operator. The Sink operator batches the `DataStream` into configurable-sized chunks for efficient storage in BigInsights. It also uses buffering techniques to de-couple the write operations from the processing of incoming streams, which allows the application to absorb peak rates and ensure that writes do not block the processing of incoming streams. Like any operator in Streams, the Sink operator that is writing to BigInsights can be part of a more complex flow graph that allows the load to be split over many concurrent Sink operators that could be distributed over many servers.

### Streams triggering deep analysis in BigInsights

Our simple example triggered deeper analysis in BigInsights by using the same Streams-BigInsights Sink operator as described in “Data ingest into BigInsights from Streams”. BigInsights does deep analysis by using the same sentiment extraction analytic as used in Streams and creates a results file to update the Streams model. For more advanced scenarios, the trigger from Streams also could contain query parameters to tailor the deep analysis in BigInsights.

## Updating the Streams analytical model from BigInsights

Streams updates its analytical model from the result of deep analysis in BigInsights. The results of the analysis in BigInsights are processed by Streams as a stream that can be part of a larger flow graph. For our simple example, the results contain a new watch list of causes for which Streams analyzes the negative sentiment.

### 7.1.7 Enabling components

The integration of data-in-motion and data-at-rest platforms is enabled by the following main types of components:

- ▶ Common analytics
- ▶ Common data formats
- ▶ Data exchange adapters

The components that are used for this simple demonstration application are available on the Streams Exchange at Streams exchange on the IBM developerWorks® website:

<https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=d4e7dc8d-0efb-44ff-9a82-897202a3021e>

The components are described in the following sections.

#### Common analytics

The same analytics capabilities can be used on Streams and BigInsights. In this simple example, we use IBM BigInsights System T text analytic capabilities to extract information from the unstructured text that is received in the emails and tweets. Streams uses a System T operator to extract the product, sentiment, and reasons from the unstructured text in the feeds. The feeds are stored in BigInsights in their raw form and processed by using the System T capabilities when the deep analysis is triggered. System T uses AQL, a declarative rule language with a SQL-like syntax to define the information extraction rules. Streams and BigInsights use the same AQL query for processing the unstructured text.

#### Common data formats

Streams formatting operators can transform data between the Streams tuple format and data formats that are used by BigInsights. In this simple example, we use JSON as the data format for storage in BigInsights. The TupleToJSON operator is used to convert the tuples in Streams to a JSON string for storage in BigInsights. The JSONToTuple operator is used to convert the JSON string that is read from BigInsights to a tuple for Streams to process.

#### Common data exchange adapters

Streams source and sink adapters can be used to exchange data with BigInsights. In this simple example, we use HDFSSource, HDFSSink, and HDFSDirectoryScan adapter operators to exchange data with BigInsights. These adapters feature similar usage patterns to the fileSource, fileSink, and directoryScan adapter operators that are provided in the Streams Programming Language (SPL) Standard Toolkit. The HDFSSink is used to write data from streams to BigInsights. The HDFSDirectoryScan operator looks for new data to read from BigInsights by using the HDFSSource operator.

## 7.1.8 BigInsights summary

IBM big data platforms (InfoSphere Streams and InfoSphere BigInsights) enable businesses to operationalize the integration of data-in-motion and data-at-rest analytics on a large scale to gain current and historical insights into their data. This ability allows faster decision making without restricting the context for those decisions. In this section, we described various scenarios in which the two platforms interact to address the big data analysis problems.

## 7.2 Integration with IBM SPSS

In this section, we describe how to write and use an InfoSphere Streams operator to run an IBM SPSS Modeler predictive model in an InfoSphere Streams application by using the IBM SPSS Modeler Solution Publisher Runtime Library API.

### 7.2.1 Value of integration

IBM SPSS Modeler provides a state of the art environment for understanding data and producing predictive models. InfoSphere Streams provides a scalable, high-performance environment for real-time analysis of data in motion, including traditional structured or semi-structured data and unstructured data types.

The combination of the power of InfoSphere Streams and the sophisticated data modeling capabilities of the IBM SPSS Modeler is valuable, as shown in Figure 7-3.

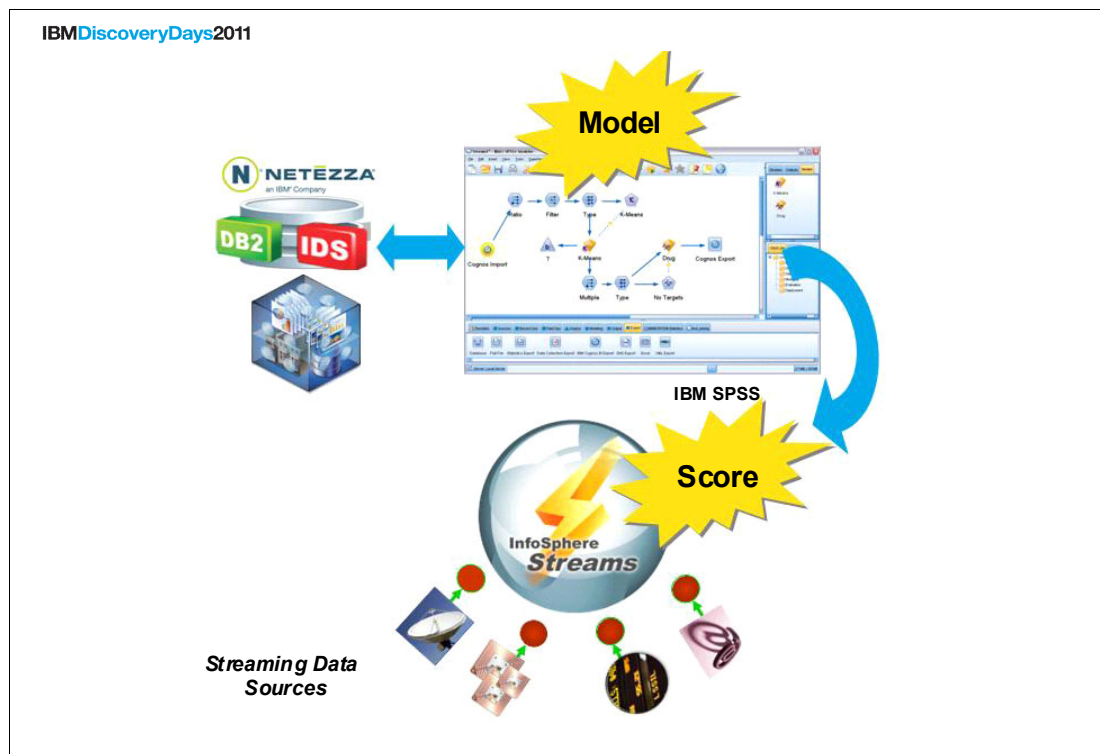


Figure 7-3 InfoSphere Streams and IBM SPSS scoring integration



The IBM SPSS Modeler can be used to discover affinities, or links, in a database by modeling and by visualization. These links correspond to groupings of cases in the data, and these groups can be investigated in detail and profiled by modeling. In the retail domain, such customer groupings might be used, for example, to target special offers to improve the response rates to direct mailings. The groupings also might be used to customize the range of products that are stocked by a branch to match the demands of its demographic base.

## **7.2.2 SPSS operators overview**

The SPSS Analytics Toolkit contains InfoSphere Streams operators that integrate with IBM SPSS Modeler and SPSS Collaboration and Deployment Services products. This product integration implements various aspects of SPSS Modeler predictive analytics in your InfoSphere Streams applications.

## **7.2.3 SPSSScoring operator**

The SPSSScoring operator integrates with SPSS Modeler Solution Publisher to enable the scoring of your SPSS Modeler designed predictive models in InfoSphere Streams applications.

The IBM SPSS Modeler can be used to discover affinities, or links, in a database by modeling and visualization. These links correspond to groupings of cases in the data, and these groups can be investigated in detail and profiled by modeling. In the retail domain, such customer groupings might be used to target special offers to improve the response rates to direct mailings or to customize the range of products that are stocked by a branch to match the demands of its demographic base.

## **7.2.4 SPSSPublish operator**

The SPSSPublish operator automates the SPSS Modeler Solution Publisher publish function. This function generates the executable images that are needed to refresh the model that is used in your InfoSphere Streams applications from the logical definition of an SPSS Modeler scoring branch that are defined in a SPSS Modeler file.

## 7.2.5 SPSSRepository operator

The SPSSRepository operator detects notification events that indicate changes to the deployed models that are managed in the SPSS Collaboration and Deployment Services repository. The operator also retrieves the indicated WebSphere Business Modeler file version for automated publish and preparation for use in your InfoSphere Streams applications, as shown in Figure 7-4.

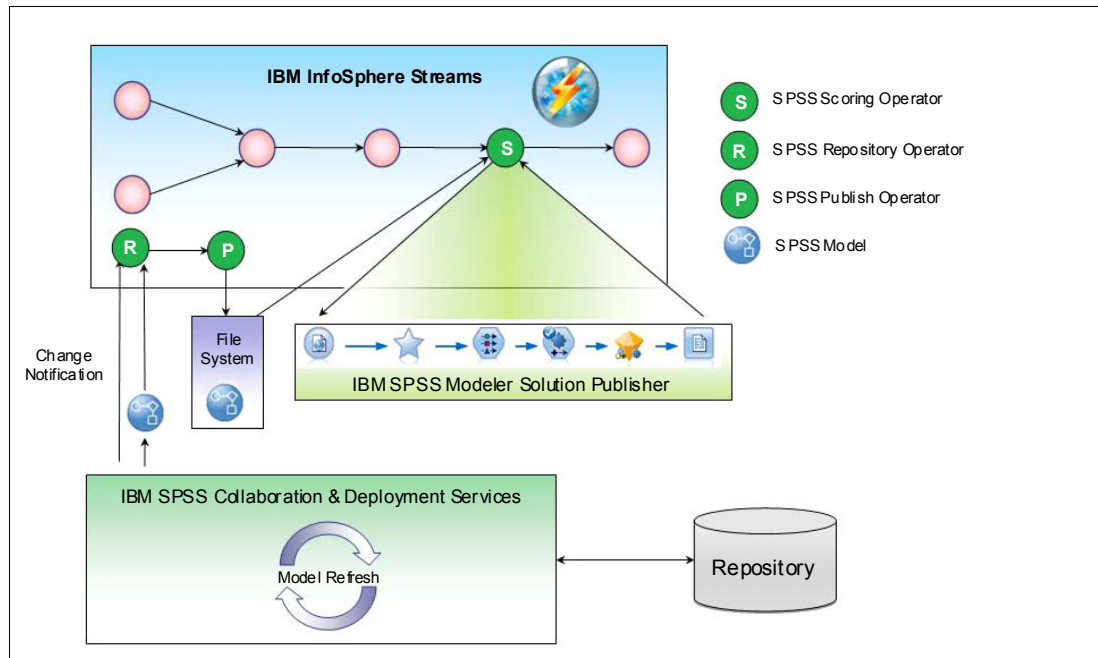


Figure 7-4 InfoSphere Streams and SPSS interaction

To implement SPSS scoring in Streams applications, you need SPSS Modeler Solution Publisher, SPSS Collaboration and Deployment Services, and Streams SPSS toolkit.

## 7.2.6 Streams and SPSS examples

The IBM SPSS Analytics Toolkit for InfoSphere Streams contains a set of simple sample applications that demonstrate how to use the various operators.

Each of these sample directories contains an SPL source file that defines the sample application and an `info.xml` file to describe the sample in InfoSphere Streams Studio and reference a common data subdirectory with the sample data and other assets that are needed to run the sample.

**Important:** To use these samples, you must publish the `model.str` file that is generating the required PIM, PAR, and XML files. The `publishHelper.sh` script is included to help you publish manually if you did not publish by using SPSS Modeler client.

The following samples are included:

▶ SPSSScoring

This sample application uses a simple SPSS Modeler scoring branch that is defined in the `model.str` file. This file is a small data file that is scored in the `input.csv` file and creates an output file that contains the predictions in the configured file sink.

▶ SPSSPublishScoring

This sample publishes any SPSS Modeler file that it is asked to according to its configuration and, in turn, trigger the refresh of the scoring in the SPSSScoring operator. The sample does not define the SPSS Collaboration and Deployment Services repository connectivity information, so the SPSS Modeler files that are presented cannot contain references to other objects in the repository. This sample must be edited to point to valid file directories on your development system.

**Important:** This application does not terminate on its own because of the DirectoryScan source operator that is used.

▶ SPSSRepositoryPublishScoring

This sample uses all three operators from this toolkit: SPSSRepository, SPSSPublish, and SPSSScoring. This sample requires an SPSS Collaboration and Deployment Services installation to work. You can choose to modify the configuration to point to any file object you deploy to the repository. After the sample is running, you can set the label that is configured or drop a second version of the object into the repository if you are not using the label recognition to get the notification that triggers the download and the publish and refresh operations.

**Important:** This application does not terminate on its own because it continually monitors notifications from the SPSS Collaboration and Deployment Services repository.

## 7.3 Integrating with databases

In this section, we describe the capability of Streams to connect to and interface with different databases. We do not describe the process that is used to set up the different databases; for that task, you must consult the database documentation.

The databases that are shown in Table 7-1 are supported by Streams and drivers.

Table 7-1 Databases supported by Streams

Database Product Name	Version	Type of Driver
DB2® Runtime Client and DB2 Client	10.1	DB2 ODBC
DB2 Runtime Client and DB2 Client	9.7	DB2 ODBC
IBM® Informix® Dynamic Server	11.5	IBM Informix ODBC
Oracle Database	11g Release 2	UnixODBC <sup>a</sup> , with an Oracle driver
IBM solidDB®	6.5	UnixODBC <sup>a</sup> , with a solidDB client driver
MySQL	5.1	UnixODBC <sup>a</sup> , with a MySQL driver <sup>b</sup>
Microsoft SQL Server	2008	UnixODBC <sup>a</sup> , with an EasySoft driver
Netezza®	6	UnixODBC <sup>a</sup> , with a Netezza driver

a. You must use unixODBC version 2.3 or later with database toolkit applications.

The unixODBC driver provides support for databases from multiple vendors. In addition to the unixODBC driver, you must install an appropriate driver for the database product. For example, solidDB connections are accomplished by linking to the database client through unixODBC with the solidDB client driver on the back end.

b. If you are using the MySQL ODBC Connector driver, it must be version 5.1.8 or later.

The following restrictions apply to Red Hat Enterprise Linux Version 6 (RHEL 6) only:

- ▶ On x86 systems that are running RHEL 6, Oracle databases are not supported.
- ▶ On IBM POWER7® systems that are running RHEL 6, IBM solidDB, Netezza, and Oracle databases are not supported.

The following sections describe steps to configure connectivity to the databases that were previously mentioned. We describe the basic concepts, some database-specific configuration steps, and present examples. When we refer to a database adapter, we mean an SPL operator that interacts with a database.

### 7.3.1 Concepts

In this section, we describe some of the basic concepts that you must understand before you connect a Streams application to a database.

#### Environment variables

For a Streams application to access a database, it must know which database product to connect to and where the client libraries for that database product are installed so the environment variables can be set. In particular, some of these environment variables must be set in the process that is running the database adapter's code; that is, in the corresponding Processing Element (PE). Each of the different database configurations requires different environment variables to be set.

Setting these environment variables can be done in one of the following ways:

- ▶ Set environment variables in the instance owner's `.bashrc` file, which is the preferred method. Make sure to restart Streams instance or Streams Studio to use the newly set environment variable.
- ▶ Specify the values of the environment variables in the `DNA.backDoorEvs` instance property, as shown in the following example:

```
streamtool setproperty -i myInstance  
DNA.backDoorEvs=OBCDINI=/mydir/odbc.ini,LD_LIBRARY_PATH=/mydir/libpath
```

In addition to the environment variables that are set in the runtime process, you must set an environment variable (named `STREAMS_ADAPTERS_ODBC_*`) at compile time that corresponds to the database that is accessed. The operators in the Streams Database Toolkit use ODBC, a standard software interface for accessing databases. The only exception to this rule is the `solidDBEnrich` operator, which connects directly by using the `solidDB` API.

At compile time, you also must set the environment variables `STREAMS_ADAPTERS_ODBC_[INC,LIB]PATH`, which is based on where the database client library is installed. Because these environment variables must be set at compile time, a Streams application can connect to only one database at a time. However, if you require that your application connects to multiple databases, the following solutions are available:

- ▶ Segment the application into multiple smaller applications, each of which is compiled with a different set of environment variables. The various segments can communicate with each other by using dynamic connections (import/export streams that are set up at run time).
- ▶ If all the databases that are considered are supported by the `unixODBC` driver manager, the same application could communicate with all of them by using this interface. For more information about `unixODBC`, see “`unixODBC: A common driver`” on page 174.

### 7.3.2 Configuration files and connection documents

The operators in the Streams Database Toolkit must be configured to connect to a database. This configuration information is specified in an XML document, which is known as the connection specification document. This document (named `connections.xml`) is often complex, detailed, and specific to a particular database, and is therefore kept separate from the SPL application code. We show an example of this document later in this section, where the same application code can connect to different databases by using different environment variables and a connection specification document.

The connection specification document contains the following kinds of specifications:

- ▶ **Connection specification:** This specification refers to the API that is used and provides the information that is needed to establish a connection to the database.

For ODBC, this information is needed for the `ODBC SQLConnect()` function, and includes three attributes (database name, user ID, and password).

For `solidDB`, this information is needed for the `solidDB SaConnect()` function and includes five attributes (communications protocol, host name, port number, user ID, and password).

- ▶ Access specification: This specification refers to the data resources that are accessed and depends on the following types of operator:
  - ODBCEnrich and ODBCSource operators run queries, and therefore must have a query element.
  - ODBCAppend operator adds records to a database table, and therefore must have a table element.
  - solidDBEnrich operator run queries against a database table, and therefore must have a table query element.

For operators that have parameters, the access specification also can include a mapping between the elements of the database table and parameters in the operator. Each access specification must also identify the connection specification to be used. Finally, the access specification must specify the schema of the data that is received from (or sent to) the database.

Streams V3.0 also includes a tool to help you test your database connections (if you are using UnixODBC) outside of a Streams application. This utility, called odbchelper, is part of the Database Toolkit that is shipped with the product. The source code is provided for this utility, and the makefile that is needed to build it. The utility can be started by using any of the following action flags:

- ▶ `help`: Displays the options and parameters that are available.
- ▶ `testconnection`: Tests the connection to an external data source instance with a user ID and password.
- ▶ `runsqlstmt`: Runs an SQL statement, either passed in on the command invocation or in a specified file.
- ▶ `runsqlquery`: Runs an SQL query, either passed in on the command invocation, or in a specified file. The results of the query are returned to STDOUT.
- ▶ `loaddelimitedfile`: Pass in a comma-delimited file that is used to create and populate a database table.

By using the `testconnection` flag, you can check whether the information in your `connections.xml` file is correct for your external data source. The other flags (`loaddelimitedfile`, `runsqlquery`, and `runsqlstmt`) are also useful. By using these flags, you can create database tables and run SQL queries or statements off it to help find and debug setup, connection, and configuration issues.

### **unixODBC: A common driver**

Among the many ODBC drivers, unixODBC deserves special attention because it provides support for databases from multiple vendors. In these cases, the database adapter's code links to the unixODBC libraries instead of directly to the database client libraries. In fact, if you were trying a database that is not listed in this section, the first thing to check is whether the unixODBC libraries support it. If it does, the database adapters can connect to it by using the unixODBC libraries by setting the environment variable `STREAMS_ADAPTERS_ODBC_UNIX_OTHER` when the application is compiled.

For the purposes of this section, we refer to Version 2.3 of unixODBC, which can be downloaded from the following website:

<http://www.unixodbc.org>

The details might be slightly different for older versions.

One thing to understand regarding the use of unixODBC is that regardless of what database to which you are connecting, there are different unixODBC configuration files. unixODBC comes with a graphical interface to manipulate these configuration files. The configuration files are designed so that the configuration can be split up across multiple files: the `odbcinst.ini`, `odbc.ini`, and `.odbc.ini` files. In this section, we present some examples, which show entire database configurations in one file (which is valid). For more information about the different unixODBC configuration files, see this website:

<http://www.unixodbc.org>

In the examples that are presented here, the entire configuration is in a nonstandard location, and the ODBCINI environment variable is set to point to that nonstandard location.

### 7.3.3 Database specifics

We present more details (including the specific environment variables) for each of the databases that are described. In particular, we specify the variables that must be set at compile time, and those variables that must be set in the processes that is running the database adapter code (PE run time). In addition to these environment variables, you must set the environment variables `STREAMS_ADAPTERS_ODBC_[INC,LIB]PATH` at compile time, which is based on where the database client library is installed. The environment variables `STREAMS_ADAPTERS_ODBC_*`, which are set at compile time and tell the compiler to which database product to connect, do not need to be set to any particular value. However, they must be set.

We do not necessarily describe the only way or even the best way to configure these databases. The following examples are provided as a guide to you and as a useful place to begin. As an example, in the section that describes SQLServer, we mention that the FreeTDS driver can be used on the back end. However, this driver is not the only driver that is available and you can as use other drivers (such as EasySoft). There also might be other client options available for other databases.

#### DB2

DB2 connections are accomplished by directly linking to the database client libraries. DB2 has a concept of an instance, and the environment variables dictate what DB2 instance should be used. The following environment variables must be set in the runtime processes:

- ▶ Environment variable for running database adapters with DB/2
- ▶ Name is Sample Value
- ▶ The `DB2INSTANCE` is `myDB2InstanceName`

The ODBC database name that is used in the `connections.xml` document should match the name of the database instance (`myDB2InstanceName`). The `STREAMS_ADAPTERS_ODBC_DB2` environment variable must be set before SPL applications that are accessing DB2 are compiled.

## Informix

IBM Informix® connections are accomplished by directly linking to the database client libraries. The STREAMS\_ADAPTERS\_ODBC\_IDS environment variable must be set before SPL applications that access Informix are compiled.

In addition, you need an sqlhosts file (pointed to by the INFORMIXSQLHOSTS environment variable), which contains the following information:

- ▶ dbservername: Database server name
- ▶ nettype: Connection type
- ▶ hostname: Host computer for the database server
- ▶ servicename: Alias for port number
- ▶ options: Options for connection (optional field)

A sqlhosts file is shown in the following example:

```
inf115  onsoctcp      myHost    55000
```

Informix also relies on an odbc.ini file that is similar to the files that are used by unixODBC. This odbc.ini file (pointed to by ODBCINI environment variable) looks similar to the following example:

```
[itest]
Description = Informix
Driver =<installed Informix directory>/lib/cli/iclit09b.so
APILevel=1
ConnectFunctions=YYY
DriverODBCVer=03.00
FileUsage=0
SQLLevel=1
smProcessPerConnect=Y
ServerName=inf115
Database=adaptersTest
Port=55000
CLIENT_LOCALE=en_us.8859-1
DB_LOCALE=en_us.8859-1
TRANSLATIONDLL=<installed Informix directory>/lib/esql/igo4a304.so
```

The data source name that is used in the connections.xml document should match the data source name in odbc.ini (itest). The ServerName field in the /mydir/odbc.ini file (inf115) should match the name of the server name that is specified in the /mydir/sqlhosts file. The environment variables must be set in the runtime processes, as shown in Table 7-2.

Table 7-2 Environment variables for running database adapters with Informix

Environment variable name	Sample value
INFORMIXDIR	<installed Informix directory>
INFORMIXSQLHOSTS	/mydir/sqlhosts
ODBCINI	/mydir//odbc.ini

## Oracle

Oracle connections are accomplished by linking to the database client through unixODBC with the Oracle driver on the back end. The STREAMS\_ADAPTERS\_ODBC\_ORACLE environment variable must be set before the SPL applications that are accessing Oracle are compiled.



You also need an `odbc.ini` file (pointed to by the `ODBCINI` environment variable), which looks similar to the following example:

```
[twins]
Driver                = <installed Oracle directory>/lib/libsqora.so.11.1
description           = test
ServerName            = 10.1.1.92:1521/twins.mydomain.com
server                = 10.1.1.92
port                  = 1521
Longs                  = F
```

The data source name that is used in the `connections.xml` document should match the data source name in `odbc.ini` (`twins`). With Oracle, you must know whether its libraries are dynamically or statically linked, which is set by `LD_LIBRARY_PATH`. The environment variables in Table 7-3 must be set in the runtime processes.

Table 7-3 Environment variables for running database adapters with Oracle

Environment variable name	Sample value
ODBCINI	/mydir/odbc.ini
ORACLE_HOME	<installed Oracle directory>
LD_LIBRARY_PATH	<installed Oracle directory>/lib (if using static linking)  OR  <installed Oracle directory>/lib:/mydir/unixodbc/lib (if NOT using static linking)

## SQLServer

SQLServer connections are also accomplished by linking to the database client through `unixODBC`. The `FreeTDS` driver can be used on the back end and downloaded from the following website:

<http://www.freetds.org>

The `STREAMS_ADAPTERS_ODBC_SQLSERVER` environment variable must be set before the SPL applications that are accessing SQLServer are compiled.

You also need an `odbc.ini` file (pointed to by the `ODBCINI` environment variable), which looks similar to the following example:

```
[mstest]
Driver                = /myTDSDir/freetds/lib/libtdsodbc.so
Description           = SQL Server
Trace                  = no
TDS_Version            = 7.0
Server                 = myHost.foo.com
Port                   = 1136
Database               = msdb
```

The data source name that is used in the `connections.xml` document should match the data source name in `odbc.ini` (`mstest`). The environment variable that is shown in Table 7-4 must be set in the runtime processes.

Table 7-4 Environment variable for running database adapters with SQLServer

Environment variable name	Sample value
ODBCINI	/mydir/odbc.ini

## MySQL

SQLServer connections are also accomplished by linking to the database client through unixODBC with the MySQL driver on the back end. The STREAMS\_ADAPTERS\_ODBC\_MYSQL environment variable must be set before the SPL applications that are accessing MySQL are compiled.

You need an `odbc.ini` file (pointed to by the ODBCINI environment variable), which looks similar to the following example:

```
[mysqltest]
Driver           =
/myMySQLDir/mysql-connector-odbc-5.1.7-linux-glibc2.3-x86-64bit/lib/libmyodbc5.so
Description     = Connector/ODBC 3.51 Driver DSN
Server          = myHost
PORT            = 3306
Database        = UserDB
OPTION          = 3
SOCKET          =
```

The data source name that is used in the `connections.xml` document should match the data source name in `odbc.ini` (`mysqltest`). The environment variable that is shown in Table 7-5 must be set in the runtime processes.

Table 7-5 Environment variable for running database adapters with MySQL

Environment variable name	Sample value
ODBCINI	/mydir/odbc.ini

## Netezza

Netezza connections are accomplished by linking to the database client through unixODBC with the Netezza driver on the back end. The STREAMS\_ADAPTERS\_ODBC\_NETEZZA environment variable must be set before the SPL applications that are accessing Netezza are compiled.

You also need an `odbc.ini` file (pointed to by the `ODBCINI` environment variable), which looks similar to the following example:

```
[netezza]
Driver           = /myNetezzaDir/lib64/libnzodbc.so
Setup           = /myNetezzaDir/lib64/libnzodbc.so
APILevel        = 1
ConnectFunctions = YYN
Description      = Netezza ODBC driver
DriverODBCVer   = 03.51
DebugLogging     = false
LogPath         = /tmp
UnicodeTranslationOption = utf8
CharacterTranslationOption = all
PreFetch        = 256
Socket          = 16384
Servername      = 127.0.1.10
Port            = 5480
Database        = streamstest
ReadOnly        = false
```

The data source name that is used in the `connections.xml` document should match the data source name in `odbc.ini` (`netezza`). The environment variables that are shown in Table 7-6 must be set in the runtime processes.

*Table 7-6 Environment variables for running database adapters with Netezza*

Environment variable name	Sample value
<code>ODBCINI</code>	<code>/mydir/odbc.ini</code>
<code>NZ_ODBC_INI_PATH</code>	<code>/mydir</code> (directory containing the <code>odbc.ini</code> file)

### **solidDB with ODBC operators**

Streams applications that use `ODBCSource`, `ODBCEnrich`, and `ODBCAppend` operators can connect to `solidDB` through `unixODBC` by using the `solidDB` client driver. The `STREAMS_ADAPTERS_ODBC_SOLID` environment variable must be set before the SPL applications that are accessing `solidDB` with `unixODBC` are compiled.

In addition, you need a `solid.ini` file (whose parent directory is pointed to by the `SOLIDDIR` environment variable), which includes the following code:

```
[Data Sources]
solidtest=tcp myHost 1964
```

You also need an `odbc.ini` file (pointed to by the `ODBCINI` environment variable), which looks similar to the following example:

```
[solidtest]
Description      = solidDB
Driver          = <installed solidDB directory>/bin/sac12x6465.so
```

The data source name that is used in the `connections.xml` document should match the data source name in `odbc.ini` (`solidtest`). The environment variables that are shown in Table 7-7 must be set in the runtime processes.

Table 7-7 Environment variables for running database adapters with solidDB (with unixODBC)

Environment variable name	Sample value
ODBCINI	/mydir/odbc.ini
SOLIDDIR	/mydir (directory containing the solid.ini file)

### solidDB with solidDBEnrich operator

solidDB connections that use the solidDBEnrichOperator are done with direct linking. No extra environment variables (other than the [INC/LIB]PATH) must be set when the database adapter's code is compiled or during run time.

## 7.3.4 Examples

In this section, we present some examples to show the two main ways of setting environment variables. We also describe the process that is used to connect to two of the databases (DB2 and Informix). In both examples, the application is the same (the Main Composite is named singleinsert). The SPL code in the application uses an ODBCAppend operator that is coded as shown in the following example:

```
( ) as writeTable = ODBCAppend(singleinsert) {
  param
    connection      : "DBPerson";
    access          : "PersonSinkSingleInsert";
    connectionDocument : "connections.xml";
}
```

### Environment variables that are set in connecting to DB2

In this example, we show how a DB2 connection can be configured by placing the runtime environment variables in the instance owner's .bashrc file.

The connections.xml file that is specified by ODBCAppend has a connection specification that looks similar to the following example:

```
<connection_specification name="DBPerson" > <ODBC database="spc97" user="mike"
password="myPassword" /></connection_specification>
```

Recall that DB2 connections are accomplished by directly linking to the database client libraries. The code is compiled by the following commands, first setting up the environment variables using the command line:

```
export STREAMS_ADAPTERS_ODBC_DB2
export STREAMS_ADAPTERS_ODBC_INCPATH=<installed DB2 directory>/include
export STREAMS_ADAPTERS_ODBC_LIBPATH=<installed DB2 directory>/lib64
sc -s -v -t /mydir/InfoSphereStreams/toolkits/com.ibm.streams.db -a -M
singleinsert
```

The following line is added to the .bashrc file so that the environment variables are picked up by the PEs:

```
export DB2INSTANCE=spc97
```

The Streams instance is created and started with the following default values:

```
streamtool mkinstance -i myInstance -hosts myHost
streamtool startinstance -i myInstance
```

When submitted, the application starts, connects successfully, and data is written to the database by using the following command:

```
streamtool submitjob -i myInstance output/singleinsert.adl
```

### **Environment variables that are set in connecting to Informix**

In this example, we show how an Informix connection could be configured by placing the runtime environment variables in the instance DNA.backDoorEvs instance property.

The connections.xml file that is specified by ODBCAppend has a connection specification that might look similar to the following example:

```
<connection_specification name="DBPerson" > <ODBC database="mike"
user="myPassword" password="db2expr1" /></connection_specification>
```

Recall that Informix connections are accomplished by directly linking to the database client libraries. The code is compiled by running the following commands:

```
export STREAMS_ADAPTERS_ODBC_IDS
export STREAMS_ADAPTERS_ODBC_INCPATH=<installed Informix directory>/incl/cli
export STREAMS_ADAPTERS_ODBC_LIBPATH=<installed Informix directory>/lib
sc -s -v -t /mydir/InfoSphereStreams/toolkits/com.ibm.streams.db -a -M
singleinsert
```

For Informix, we set up an sqlhost file and an odbc.ini file, as is previously described in this chapter.

The Streams instance is created by using the DNA.backDoorEvs property and then is started by using the following commands:

```
streamtool mkinstance -i myInstance -property
DNA.backDoorEvs=INFORMIXDIR=<installed Informix
directory>,INFORMIXSQLHOSTS=/mydir/sqlhosts,ODBCINI=/mydir/odbc.ini
streamtool startinstance -i myInstance
```

When submitted, the application starts, connects successfully, and data is written to the database by running the following command:

```
streamtool submitjob -i myInstance output/singleinsert.adl
```

## 7.4 Streams and DataStage

The IBM InfoSphere DataStage Integration Toolkit provides operators and commands that facilitate integration between IBM InfoSphere Streams (IS) and IBM InfoSphere DataStage.

### 7.4.1 Integration scenarios

InfoSphere Streams and DataStage integration could be used in the following scenarios:

- ▶ An enterprise might want to send data from DataStage to Streams to perform near real-time analytic processing (RTAP) on the DataStream. By sending data to Streams from the DataStage flow, Streams can perform RTAP in parallel to data that is loaded into a warehouse by DataStage.
- ▶ An enterprise might want to send data from Streams to DataStage. A typical use-case might be processing telco call details. The Streams job performs RTAP processing and then forwards the data to Data Stage to enrich, transform, and store the call details for archival and lineage purposes.
- ▶ A Streams application might require a data source that is not provided by Streams but has a first-class connector in DataStage (for example, Netezza or SAP).
- ▶ A Customer relationship management system stores inputs that are reached in clients unstructured data (text notes, pictures, and voice), DataStage extracts the clients unstructured data from the CRM and sends it to the Streams to perform RTAP (for example, to generate clients sentiment that is based on the text dispute notes content), DataStage sends back RTAP processing result to CRM.

### 7.4.2 Application considerations

Use the following procedure if you have an existing Streams application that you want to exchange data with by using DataStage:

1. Identify the Streams application data that must be exchange with IS.
2. Model the data with the Streams schema written in SPL.
3. Change the Streams Application SPL by adding the DSSink/DSSource Endpoints.
4. Compile the SPL application to generate an application description language (ADL) file.
5. Import the Endpoint metadata into IS metadata server by using the Streams metadata import by using InfoSphere Metadata Asset Manager (IMAM).
6. Design the job that exchanges data with the Streams application. The imported Endpoints can be used to configure the Streams connector, the connection, and the column interface of the connector.

If you want to add the Streams application processing to an existing DataStage job, the following procedure should be used:

1. Identify the data that you want to exchange with the Streams application.
2. Modify the DataStage job by adding the Streams connector to the job.
3. Configure the Streams connector properties and create the column interface.
4. Link the connector to the job data flow.
5. Configure the Streams connector connection and assign the names to the connection and the application scope.
6. Save and compile the job.
7. Generate an SPL stub that can be used as a template to create SPL code for the Streams application.

The import process generates for each Streams connector connection DSSource/DSSink SPL code with associated endpoint Schema. The name of the endpoint corresponds to the name of the connection that is used by the Streams connector in DataStage job. The SPL schema corresponds to the Streams connector column interface.

### 7.4.3 Streams to DataStage metadata import

To exchange Streams data into the DataStage job, you must import the Streams ADL file into DataStage metadata by using following procedure:

1. On the Streams side, run the **generate-ds-endpoint-defs** command to generate an Endpoint Definition File (EDF is in XML file format) from one or more ADL files:

```
/opt/ibm/InfoSphereStreams/toolkits/com.ibm.streams.etl/bin/generate-ds-endpoint-defs FeatureDemoDS/FeatureDemoDS.adl
```

2. Transfer the following EDF XML file to the IMAM client machine:

```
FeatureDemoDS/FeatureDemoDS.endpoints.xml
```

3. Run new Streams importer in IMAM to import EDF to the StreamsEndPoint model. Streams importer translates EDF XML into IMAM XML conforming to the Streams model schema
4. In the job design, select endpoint metadata from stage. The connection name and columns are populated accordingly.

## 7.4.4 Connector stage

DataStage version 9.1 or later is supported by this toolkit. DataStage V9.1 or later includes a new connector that is called Streams connector, as shown in Figure 7-5. This connector should be configured before it can be used in a DataStage job flow.

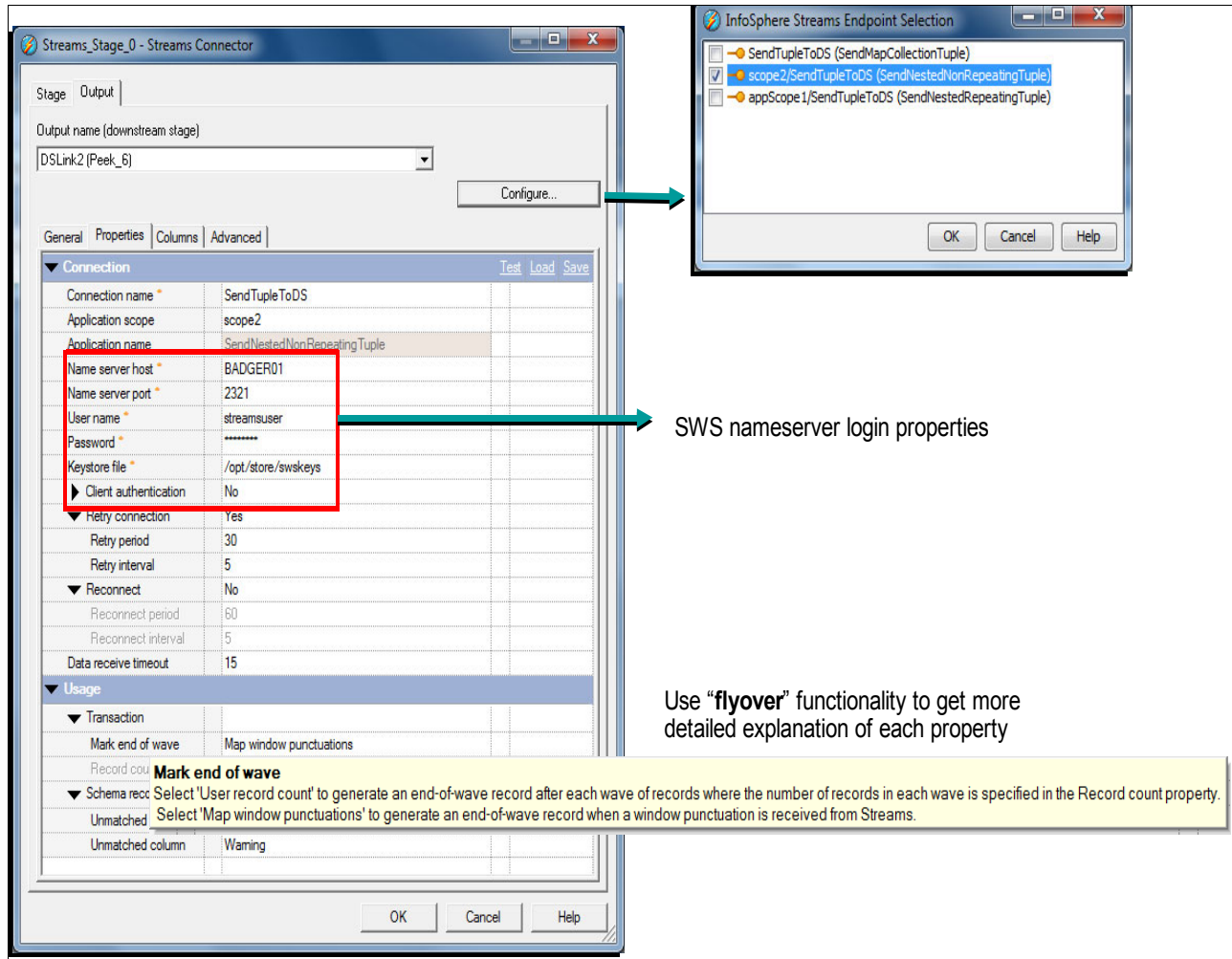


Figure 7-5 Stream Connector stage properties

The Configure button is used to start the Endpoint selection dialog. The user can select any Endpoint that is imported in the IS metadata server by using InfoSphere Streams IMAM importer.

The following parameters are stored in the Streams application SPL code. They are used to identify the DSSource DSSink endpoints. The Streams connector uses these properties as input for the REST query against the SWS Name Server:

- ▶ Connection name
- ▶ Application scope
- ▶ Application name



## 7.4.5 Sample DataStage job

Figure 7-6 shows how a Streams connector stage can be used to send and receive data.

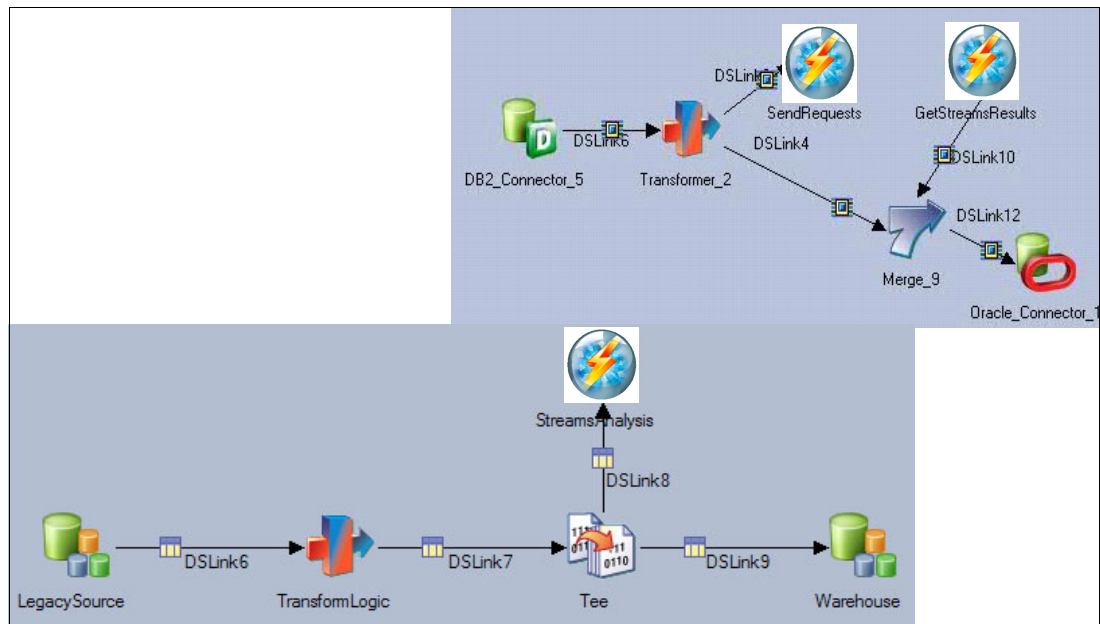


Figure 7-6 DataStage job to send and receive data from Streams

## 7.4.6 Configuration steps in a Streams server environment

IBM InfoSphere Streams must be installed and the STREAMS\_INSTALL environment variable must be set to the InfoSphere Streams installation directory.

Configure the SPL compiler to find the root directory of the toolkit. Set the STREAMS\_SPLPATH environment variable to the root directory of a toolkit or multiple toolkits by using a colon (:) as a separator, as shown in the following example:

```
export STREAMS_SPLPATH=$STREAMS_INSTALL/toolkits/com.ibm.streams.et1
```

When you compile your SPL applications, specify the **-t** or **--spl-path** command parameter on the **sc** command. For example, when you compile an SPL main composite that is called MyMain, specify the toolkit as shown in the following example:

```
sc -t $STREAMS_INSTALL/toolkits/com.ibm.streams.et1 -M MyMain
```

**Important:** These command parameters override the STREAMS\_SPLPATH environment variable.

Optionally (but not suggested), when you are developing your SPL application, add a use directive. For example, add the following clause in your SPL source file:

```
use com.ibm.streams.et1.datastage.adapters::*;
```

## 7.4.7 DataStage adapters

The InfoSphere DataStage Integration Toolkit includes the following adapter operators:

- ▶ DSSource operator: Use this operator in IBM InfoSphere Streams applications to receive data from IBM InfoSphere DataStage.
- ▶ DSSink operator: Use this operator to send data to IBM InfoSphere DataStage from IBM InfoSphere.

The operators share the following common namespace:

`com.ibm.streams.etl.datastage.adapters`, as shown in Figure 7-7.

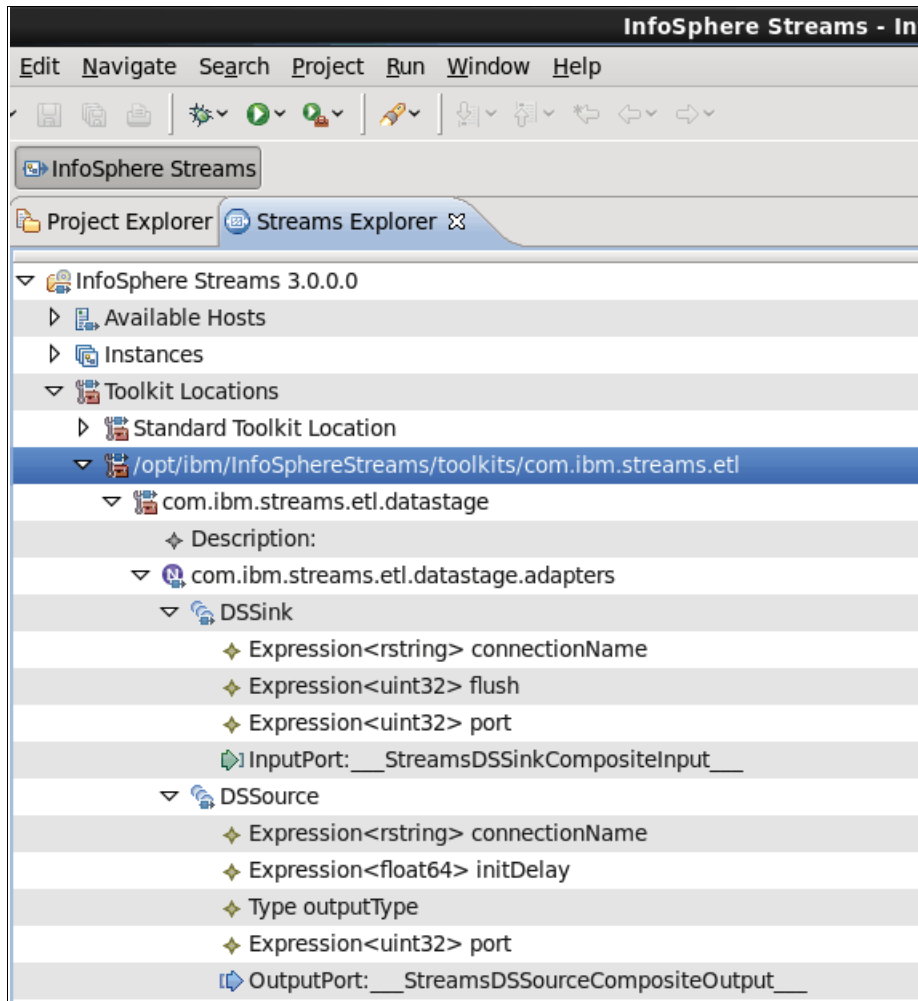


Figure 7-7 DataStage operators in Streams Studio

## 7.5 Integrating with XML data

Streams V3 supports XML data type as part of the standard SPL toolkit. The XML operator is called XMLParse. The XMLParse operator accepts a single input stream and generates tuples as a result.

### 7.5.1 XMLParse Operator

The XMLParse operator is available in InfoSphere Streams Version 3.0 and later and supports standard FileSink and FileSource operators.

#### ***Input Port***

The input to this operator is a single stream that contains an attribute with XML data to convert. The one attribute that contains XML data must have type `rstring`, `ustring`, `blob`, or `xml`. If the attribute type is `xml`, it represents a complete XML document. If the attribute type is `rstring`, `ustring`, or `blob`, the attribute might contain chunks of XML that are not well-formed and might be contained across multiple input tuples. The XMLParse operator acts as though the chunks are concatenated together. The concatenated XML might contain multiple sequential XML documents.

#### ***Output Ports***

The XMLParse operator is configurable with one or more output ports. The output ports are mutating and their punctuation mode is generating. Each output port generates tuples that correspond to one subtree of the input XML. The specific subtree of the XML document that triggers a tuple is specified by the trigger parameter that uses a subset of XPath. Each output stream corresponds to one expression on the trigger.

Tuples are generated as the XML documents are parsed, and a WindowMarker punctuation is generated at the end of each XML document. If errors occur when the XML is parsed, the errors are logged but the tuples are not generated until the start of the next trigger. Receipt of a WindowMarker punctuation resets the XMLParse operator, causing it to start parsing from the beginning of a new XML document. Tuples are output from a stream when the end tag of the element that is identified by the trigger parameter for that stream is seen.

## 7.5.2 XMLParse example

The example that us shown in Figure 7-8 uses FileSource as input where the format is specified as XML blob. The second operator is XMLParse where XML schema is defined. Finally, a FileSink operator is used to produce the result of the parsed document.

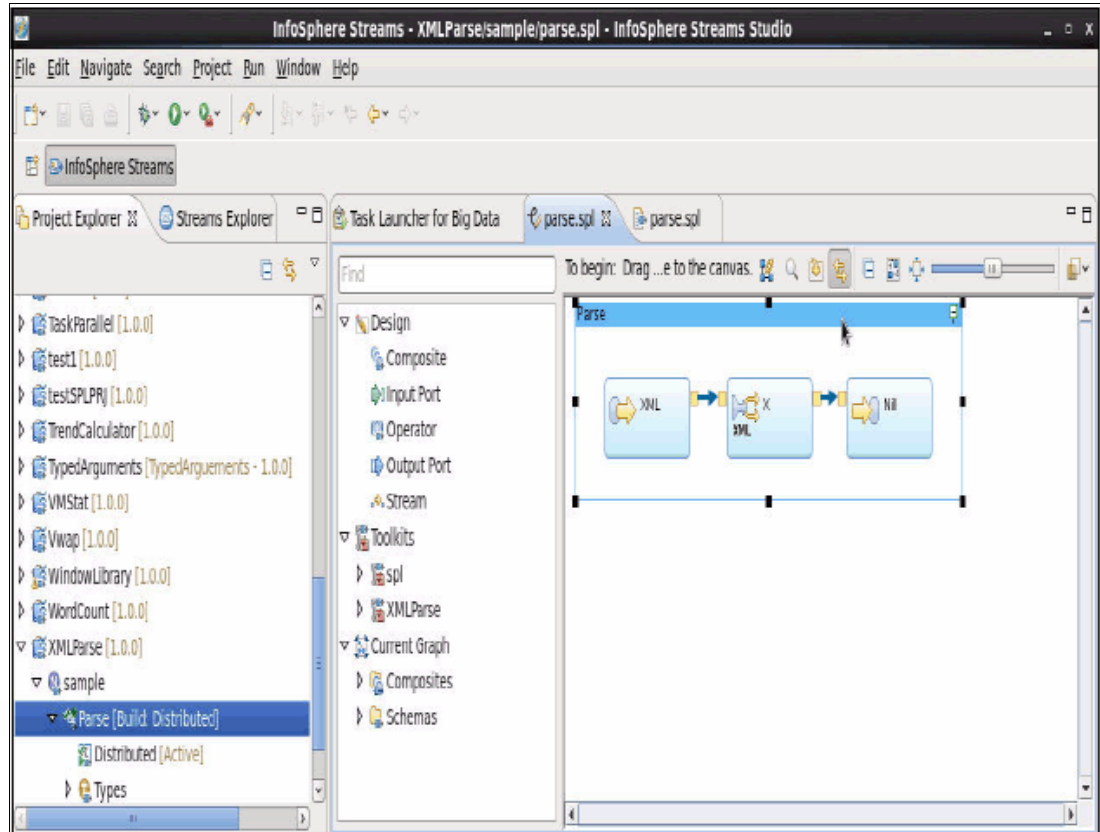


Figure 7-8 Example of XMLParse operator

The book catalog is represented in the following XML document:

```
<catalog>
  <book price="30.99">
    <title>This is a long and boring title</title>
    <author>John Smith</author>
    <author>Howard Hughes</author>
    <ignoredField>No one looks at me</ignoredField>
    <reference quality="-1">
      <book>The first reference</book>
    </reference>
    <reference quality="100">
      <book>Another Book</book>
    </reference>
  </book>
  <book price="9.99">
    <title>BestSeller #1</title>
    <author>Anonymous</author>
    <author><![CDATA[This can have <><>; in it ]]></author>
    <ignoredField>I mean it: no one looks at me</ignoredField>
    <!-- No references -->
  </book>
</catalog>
```

```

    </book>
</catalog>

```

The generated SPL code look similar to the following example:

```

composite Parse {
  type ReferenceType = tuple <int32 quality, rstring book>;
    BookInfo = tuple<rstring title, list<rstring> authors, decimal32 price,
list<ReferenceType> references>;

  graph

    // An spl.adapter::FileSource operator is configured to read in the XML from
    the file "catalog.xml" in 1K byte chunks and produce output
    // tuples each containing a 1K "blob" of XML. Other block sizes could be used
    if desired.
    stream <blob xmlData> XML = FileSource () {
      param file : "catalog.xml";
      format: block;
      blockSize : 1024u;
    }

    // An spl.XML.XMLParse operator parses the incoming block-form XML data into
    tuples. Each <book> XML element produces one output
    // tuple. Elements and attributes of the <book> element are assigned to the
    output tuple's attributes.
    stream<BookInfo> X = XMLParse(XML) {
      param trigger : "/catalog/book";
      parsing : permissive; // log and ignore errors
      output X : title = XPath ("title/text()"),
        authors = XPathList ("author/text()"),
        price = (decimal32) XPath ("@price"),
        references = XPathList ("reference", { quality = (int32) XPath
        ("@quality"),
        book = XPath
        ("book/text()") });
    }

    // An spl.adapter::FileSink operator writes the results of the parsed XML to
    the output file Results.txt.
    () as Nil1 = FileSink(X) {
      param file : "Results.txt";
    }
  }
}

```

## 7.6 Integration with IBM WebSphere MQ and WebSphere MQ Low Latency Messaging

Streams is integrated with WebSphere MQ by using one of the following processes:

- ▶ WebSphere MQ queue/topic integration

WebSphere MQ is a complete universal messaging backbone, enabling rapid, reliable, and secure transport of messages and data between applications, systems, and services. The WebSphere MQ toolkit provides adapters to read and write to and from WebSphere MQ queues or topics. This integration provides time independent message delivery to and from Streams. The XMSSink and XMSSource operators provide this functionality.

- ▶ WebSphere MQ Low Latency Messaging integration

IBM WebSphere MQ Low Latency Messaging is a high-throughput, low-latency messaging transport. It is optimized for high-volume, low-latency scenarios where the speed of data delivery is paramount. The InfoSphere Streams Low Latency Messaging transport uses Version 2.5 fix pack 1 (2.5.0.1) of the Low Latency Messaging Reliable Unicast Messaging (RUM) capabilities that support InfiniBand and TCP/IP over Ethernet. The MQRMMSink operator provides this functionality.

### 7.6.1 Architecture

Figure 7-9 shows how Streams links to WebSphere MQ by using XMSSink and XMSSource.

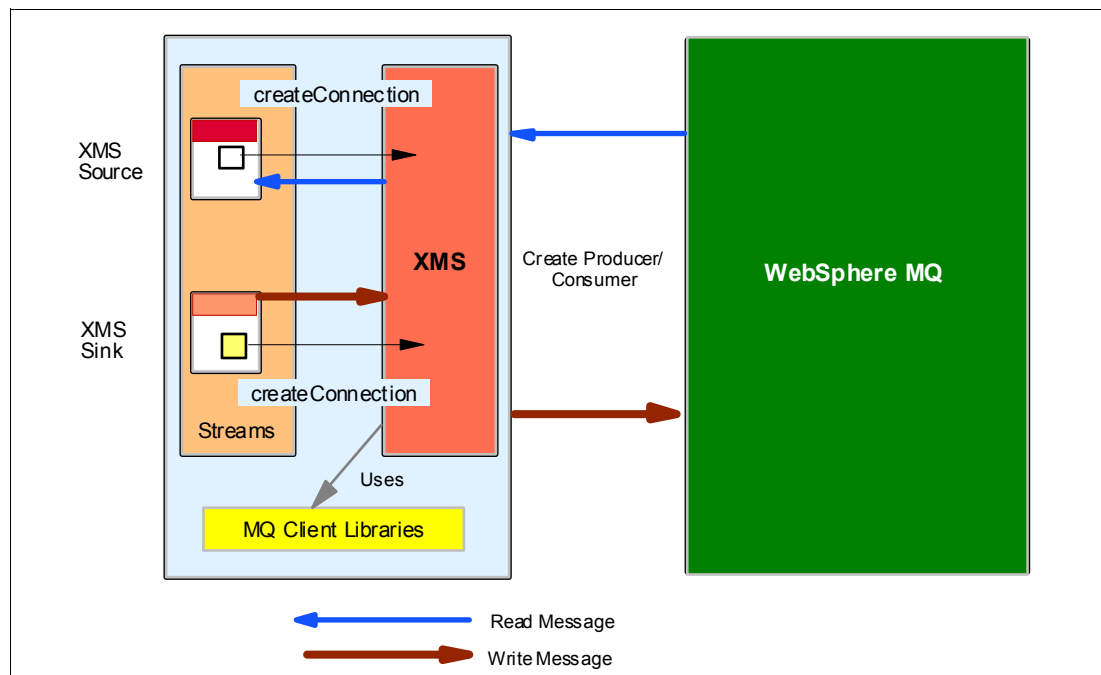


Figure 7-9 InfoSphere Streams and WebSphere Message Queue

## 7.6.2 Use cases

In this section, we describe a few applications where it is useful to link InfoSphere Streams to backend applications and databases by using WebSphere MQ.

### **Money laundering, fraud, and blacklist verification**

If someone commits a fraud or money laundering scheme on a global scale, you can set up a system to intercept such transactions. If a known offender commits a transaction from a suspicious country and sends money to a person in another country, this transaction must be verified and flagged. Streams can be used to validate and compute the results in real time and the aggregated data can be posted on to WebSphere MQ so that required lines of business (LOBs) and departments can use and act on this important transaction information and try to prevent such fraudulent activity or act swiftly to reverse such transactions. This ability protects the organizations against fraud and can meet regulatory and risk compliance requirements.

### **Real-time campaign management**

Credit card companies monitor customer purchases on a regular basis. The companies know where the transaction is committed and can offer various cross sell and up sell products. The transaction that is committed is captured by Streams and then Streams processes this information and posts the message onto the messaging queue. A campaign management application can further connect to the messaging queue and instantiate a real-time campaign.

### **Department of Motor Vehicles: Track suspects and stolen vehicles or devices**

Consider the Department of Motor vehicles (DMV), which issues licenses to many vehicles. Electronic Toll Collection (ETC) allows for electronic collection of tolls by identifying cars that are enrolled under the program and electronically debiting the account of registered car owners without requiring them to stop. When a car passes through a toll booth, the sensor that is there automatically detects the vehicle ID and debits the amount by using the transponder that is attached to that vehicle. The signal from this transponder is sent to a computer that sends appropriate signals to a centralized system that debits the account according to the toll and time of the day. Many vehicles pass through these toll booths on a daily basis.

But some of these vehicles are stolen. Information from ETC can be used for smarter crime detection and prevention. Streams can use this information for detection and prevention of crime in the following scenarios:

- ▶ **Tracking stolen cars**

When a car theft is reported to enforcers, an entry is made to the systems that contains details about stolen cars. Assuming that all toll information is continuously sent and filtered against the stolen car list by Streams, when the stolen car passes a toll, it can be flagged and sent to business process management (BPM) systems. Because BPM systems are mostly compliant with WebSphere MQ, data from Streams can be sent through XMSSink to WebSphere MQ, which, in turn, is coupled to BPM systems. These systems can issue the necessary alert to police about the last whereabouts of the stolen car and thus assist them to find the vehicle.

- ▶ Tracking suspects

This scenario considers tracking suspects who were blacklisted for criminal activities. The information from ETC is sent to Streams and Streams looks up a list of cars that belong to suspects. This information can be sent to BPM systems for further action. Because BPM systems are generally compliant with WebSphere MQ, when a match is found, XMSSink can write this data to WebSphere MQ, which, in turn, is coupled to BPM systems. An alert can be raised and necessary actions can be taken.

- ▶ Tracking stolen devices

When the theft of a device is reported to police, an entry is made to systems that contain details about stolen devices. Assuming that all toll information is continuously sent and filtered against the stolen device list by Streams, when a car that contains the stolen device passes a toll, it can be flagged and sent to BPM systems. Because BPM systems are mostly compliant with WebSphere MQ, data from Streams can be sent through XMSSink to WebSphere MQ, which, in turn, is coupled to BPM systems. These systems can issue the necessary alerts to the police about the last whereabouts of the car that contains the stolen device and thus assists them to find the device.

To further extend these use cases, we can use the geographic information system (GIS) toolkit to obtain the tentative location of the vehicle and, based on the speed and the route in which that vehicle took, we can help track it. This information also can be plotted in a map and displayed to interested parties.

### 7.6.3 Configuration setup for WebSphere MQ Low Latency Messaging in Streams

InfoSphere Streams provides a default Low Latency Messaging configuration file with advanced configuration settings. The default configuration includes values that can improve the performance of Streams applications. To add the default configuration to a Streams instance that was created before Version 1.2.0.1, enter the following command:

```
streamtool setproperty -i <instance-id>  
LLMInputPortConfigFile="%STREAMS_INSTALL%/etc/cfg/llm.inputport.properties"  
LLMOutputPortConfigFile="%STREAMS_INSTALL%/etc/cfg/llm.outputport.properties"
```

When the Low Latency Messaging transport is used, several instance configuration properties exist for setting certain Low Latency Messaging parameters. Information regarding these properties can be obtained by entering the `streamtool man properties`.

These configuration files can be found in `$STREAMS_INSTALL/etc/cfg` directory, with names `llm.inputport.properties` and `llm.outputport.properties`. The default configuration files provide settings for advanced configuration parameters that can improve the performance of Streams applications.

The environment variables `STREAMS_ADAPTERS_MQ_LLM_INCPATH` and `STREAMS_ADAPTERS_MQ_LLM_LIBPATH` must be defined to be the path names of the directories where the external libraries and header files for WebSphere MQ Low Latency Messaging 2.2 are installed.



## WebSphere MQ Low Latency Messaging operator MQRMMSink

The MQRMMSink operator outputs tuples in a stream over an WebSphere MQ Low Latency Messaging RMM transport, as shown in Figure 7-10.

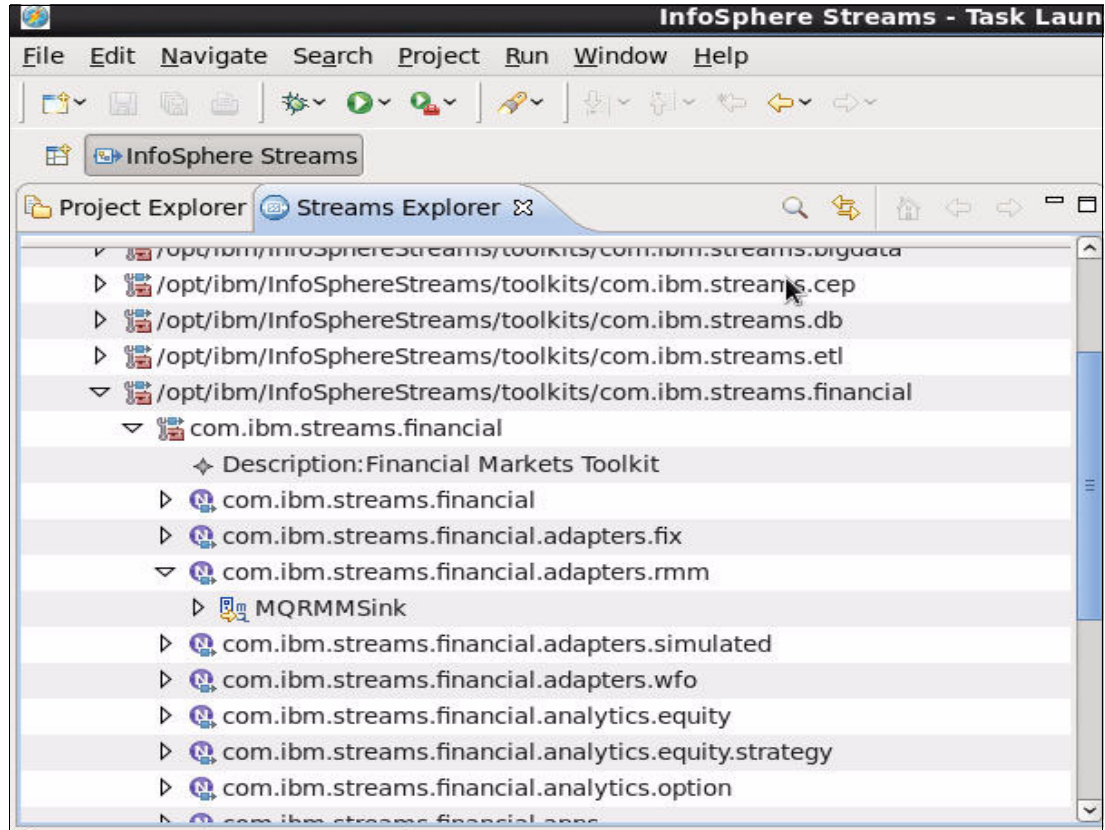


Figure 7-10 Location of MQRMMSink operator

The operator is in `com.ibm.streams.financial` toolkit. The MQRMMSink operator is declared as shown in the following example:

```
use com.ibm.streams.financial.adapters.rmm::MQRMMSink;
() as <ID> = MQRMMSink ( <input-stream> )
{
  param
    connectionDocument : "<connections-document-filename>";
    connection : "<connection-specification-name>" ;
    access : "<access-specification-name>";
}
```

The use of the MQRMMSink operator looks similar to the following example:

```
use com.ibm.streams.financial.adapters.rmm::MQRMMSink;
() as MySink1 = MQRMMSink (TradeQuoteStream)
{
  param
    connectionDocument: "connections.xml";
    connection: "MqRmmConnection";
    access: "MqRmmSinkAccess";
}
```

## 7.6.4 Connection specification for WebSphere MQ

As described in 7.6.3, “Configuration setup for WebSphere MQ Low Latency Messaging in Streams” on page 192, a sample connectionDocument is shown in Example 7-1.

*Example 7-1 Sample connection specification document*

---

```
<st:connections xmlns:st="http://www.ibm.com/xmlns/prod/streams/adapters"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <connection_specifications>
        <connection_specification name="conn1">
            <XMS initial_context="file:///var/mqm/EndToEndJndi/"
connection_factory="EndToEndConFac"/>
        </connection_specification>
    </connection_specifications>

    <access_specifications>

        <access_specification name="access1">
            <destination identifier="EndToEndDest"
delivery_mode="persistent" message_class="map" />
            <uses_connection connection="conn1"/>
            <native_schema>
                <attribute name="ID" type="Long" />
                <attribute name="transID" type="Long" />
                <attribute name="vehID" type="String" length="15"
/>
                <attribute name="dateTime" type="String"
length="30" />
            </native_schema>
            <attribute name="tollName" type="String" length="10" />
        </access_specification>
    </access_specifications>

</st:connections>
```

---

## 7.6.5 WebSphere MQ operators

The WebSphere MQ toolkit provides XMSSource and XMSSink operators to connect to a WebSphere MQ queue or topic. The operators are in `com.ibm.streams.messaging.xmls`, as shown in Figure 7-11.

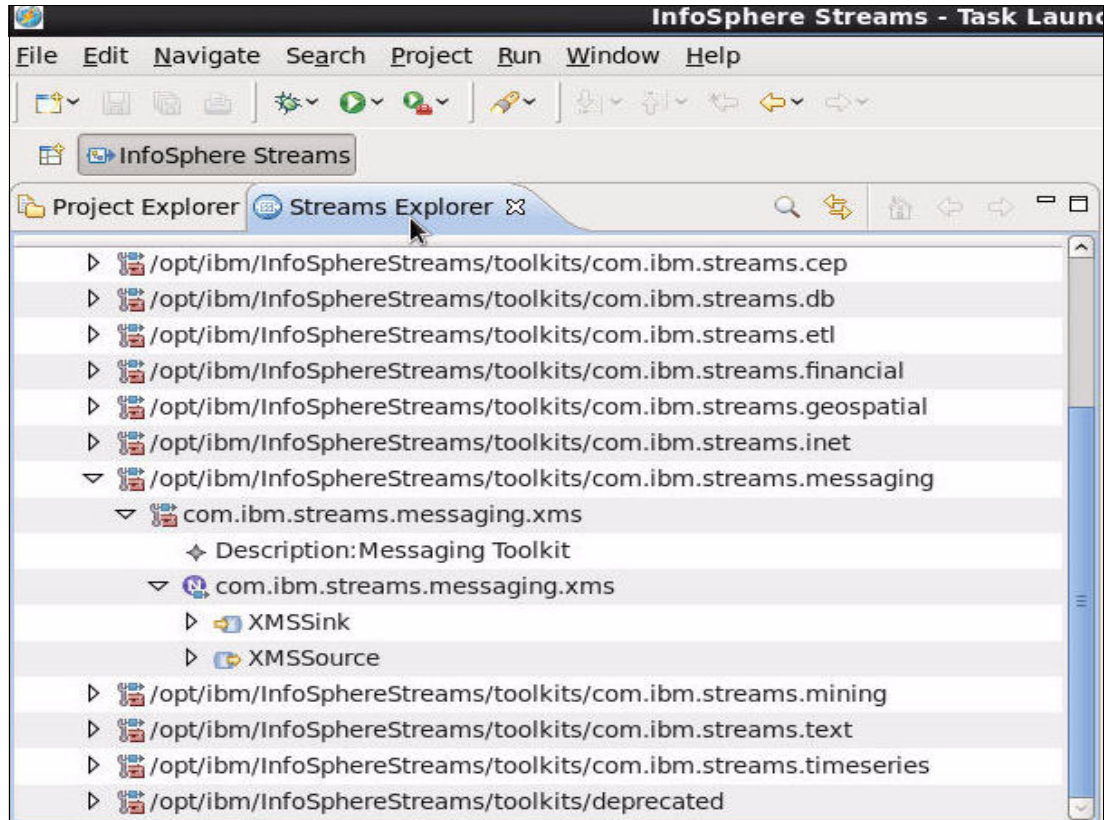


Figure 7-11 Location of XMSSource and XMSSink operators

### XMSSource

This operator reads data from a WebSphere MQ queue or a topic and pushes the data into Streams server. The operator validates if the Streams application can connect to WebSphere MQ during run time and that a queue or topic that is specified exists.

### XMSSink

This operator publishes data from Streams to a WebSphere MQ queue or topic. The operator validates if the Streams application can connect to WebSphere MQ during run time and that a queue or topic that is specified exists.

## 7.6.6 Software requirements

IBM Message Service client for C/C++ (which is also known as XMS client) version 2.0.1 is required. The XMS client must be installed on the same machine as Streams, and an environment variable (XMS\_HOME) must be set to point to the installation location of XMS Client.

The WebSphere MQ Server and WebSphere MQ Client must be installed at the following levels:

- ▶ WebSphere MQ 7.0 + All Fix Pack 7.0.0.1 or later
- ▶ WebSphere MQ 6.0 + Refresh Pack 6.0.1.1 or later
- ▶ WebSphere MQ 7.1 is not supported by the current version of XMS: 2.0.1

## 7.7 Integration with Data Explorer

With the integration with DataExplorer, you can push data into InfoSphere Data Explorer infrastructure. The operator is shipped and is part of Streams big data toolkit called DataExplorerPush.

InfoSphere Data Explorer provides secure federated navigation and discovery across a range of enterprise content and big data to maximize return on information.

Any organization that is starting a big data project can benefit from InfoSphere Data Explorer to enable discovery and navigation of big data. Organizations that have many enterprise applications that manage data could potentially be used to augment and drive big data analyses.

By delivering information to the right people at the right time, InfoSphere Data Explorer ensures that organizations gain the full value of their data, which enables improved operations, real-time decisions, better understanding of customers, increased innovation, and actionable insight.

Data Explorer contains the following features and benefits:

- ▶ Federated navigation and discovery across a broad range of applications, data sources, and file types. Includes access to enterprise applications, such as content management systems, customer relationship management systems, supply chain management, email systems, relational database management systems, web pages, and networked file systems.
- ▶ Access to non-indexed systems, such as premium information services, supplier and partner portals and catalogs, and older applications. Results from these non-indexed sources can be merged with results from the InfoSphere Data Explorer index or kept separate.
- ▶ Rapid deployment of 360 degree information applications that combine structured, unstructured, and semi-structured data to provide unique contextual views and insights that are not obvious when data is viewed from a single source or repository.
- ▶ Highly relevant navigation results for precise discovery against large and small structured and unstructured data sets. Relevance model accommodates diverse document sizes and formats while consistent results are delivered.
- ▶ Sophisticated security model, including cross-domain and field-level security to ensure that users only see content they see if they are logged into the source application.

- ▶ Unique position-based index structure that is compact and versatile and enables features such as rapid refresh, real-time searching, field-level updates, and security.
- ▶ Rich analytics, including clustering, categorization, entity and metadata extraction, faceted navigation, conceptual search, name matching, and document de-duplication.
- ▶ Collaboration capabilities to enable users to tag, comment, organize, and rate content to enhance navigation and discovery results for other users.
- ▶ Highly elastic, fully distributed architecture offering fault-tolerance, vertical and horizontal scalability, master-master replication, and shared nothing deployment.

## 7.7.1 Usage

For big data deployments, InfoSphere Data Explorer provides the following features:

- ▶ Ability to explore the full variety of enterprise information to discover data that enhances big data analytic projects.
- ▶ Ability to fuse enterprise information with data that is stored in BigInsights and other Hadoop-based systems for unique insights and analytic results.
- ▶ Rapid search, navigation, and visibility into data that is managed in BigInsights for data scientists, developers, and end-users.
- ▶ Integration with IBM InfoSphere Streams for indexing and analysis of in-motion data.

The DataExplorerPush operator inserts records into DataExplorer index by using Big Search API.

As shown in Figure 7-12, InfoSphere Data Explorer infrastructure and Streams can be in the same or different machines.

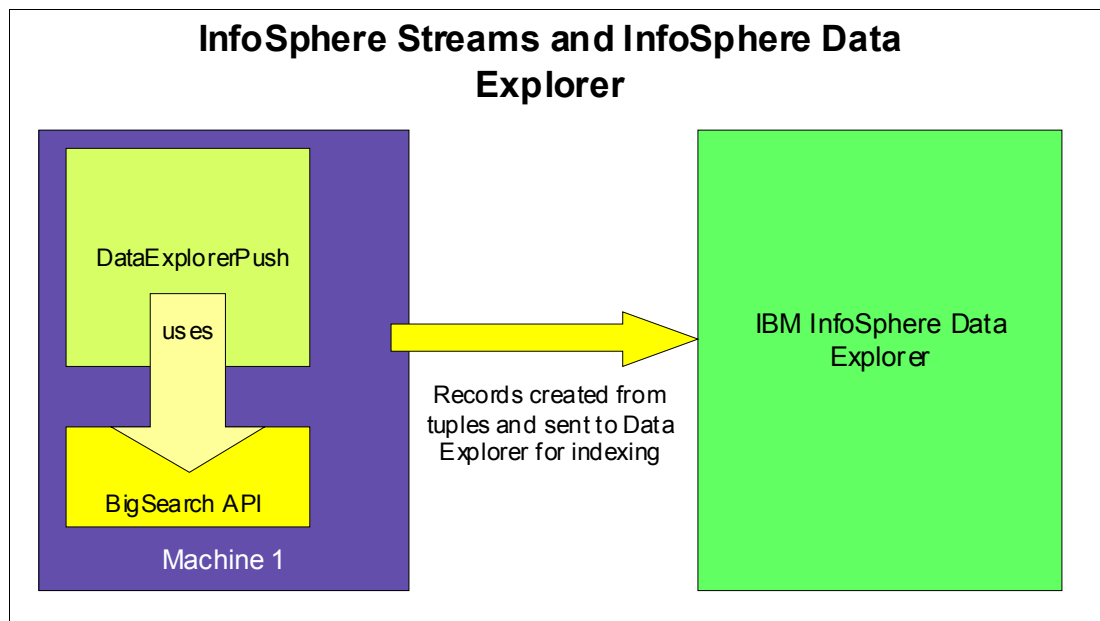


Figure 7-12 InfoSphere Streams and InfoSphere Data Explorer

The environment variable BIGSEARCH\_JAR must be set before the operator is compiled. This variable is the name of bigsearch\_api jar file, as shown in the following example:

```
export BIGSEARCH_JAR=/opt/ibm/DataExplorer/lib/bigsearch1.jar
```

The environment variable must be set in the command line before Eclipse is started if Streams Studio is used for application development.



# IBM InfoSphere Streams administration

This chapter describes the important steps to administer an InfoSphere Streams environment. We explore in detail the Streams Console application for instance monitoring and configuration tasks.

You find information about creating and updating a Streams instance by using the Streams Instances Manager and how to monitor and configure your running applications by using the Streams Console. We also describe the InfoSphere Streams Recovery database feature and how to set it up for your Streams environment.

The following topics are covered in this chapter:

- ▶ The InfoSphere Streams Instances Manager application  
The Streams Instances Manager application provides you a graphical interface to create, configure, and manage Streams instances.
- ▶ The InfoSphere Streams Console application  
The Streams Console is a web-based GUI that is provided by the Streams Web Service (SWS).
- ▶ The InfoSphere Recovery database service  
The InfoSphere Streams recovery database is an IBM DB2 database that records the state of instance services.

## 8.1 InfoSphere Streams Instance Management

We are going to use the InfoSphere Streams First Steps application to start the Instances Manager application. The InfoSphere Streams First Steps provides useful post-installation assistants so you can get your Streams application environment running quickly, as shown in Figure 8-1.

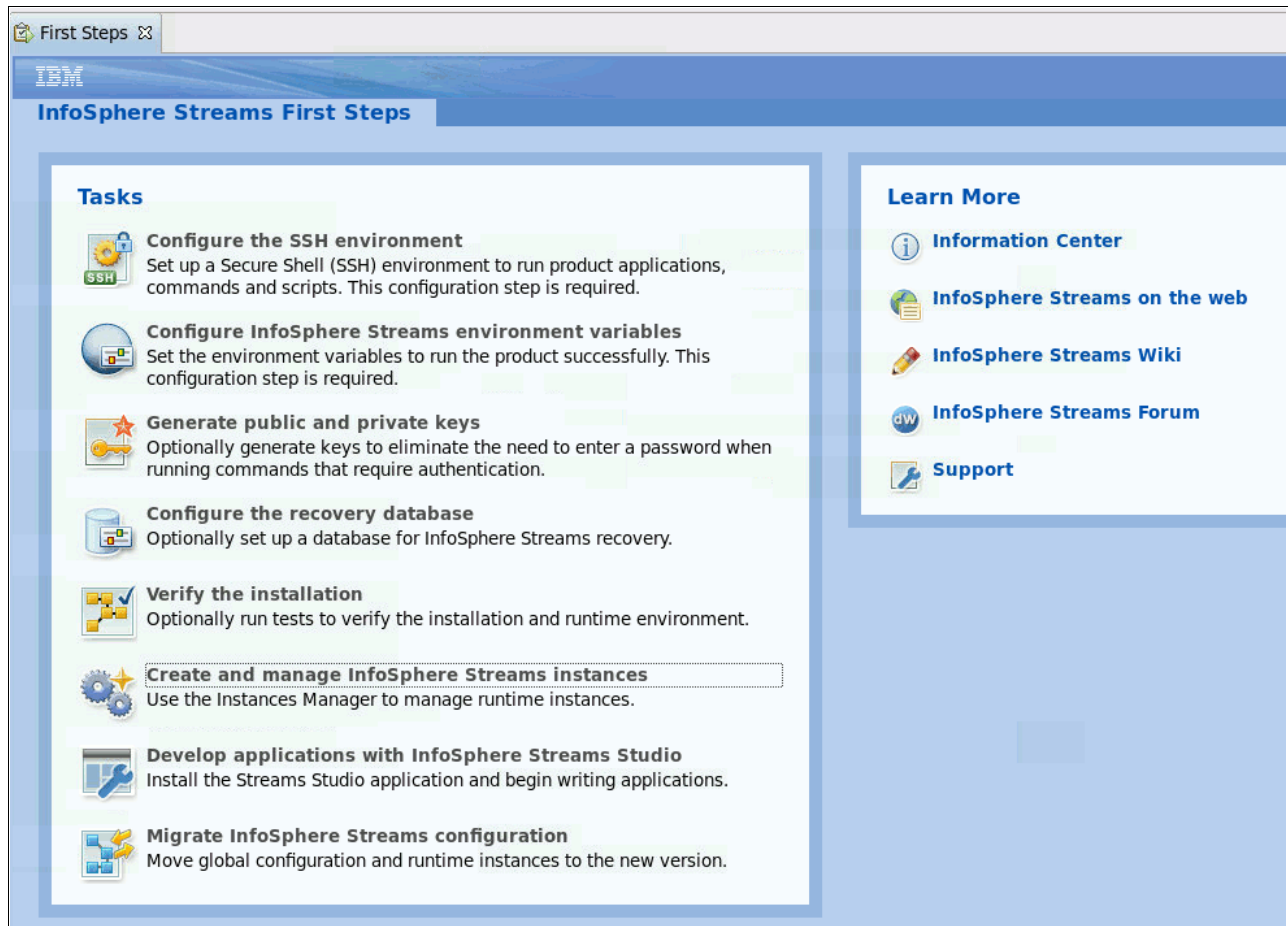


Figure 8-1 InfoSphere Streams First Steps window

To launch the First Steps Application, enter the following command:

*Example 8-1 Shows the command syntax that is used to launch the First Steps application*

```
streamtool launch --firststeps
```

**Important:** You must be logged in as a Streams administrator.

**For more information:** For more information about First Steps, see IBM InfoSphere Streams First Steps configuration on page 217.



## 8.1.1 Instances Manager

The Streams Instances Manager application provides a GUI that is used to create, configure, and manage Streams instances. When you create an instance, you specify the instance name and properties, the hosts that run the instances, and the runtime services on each host.

**Restriction:** A user with root authority cannot create an instance.

For more information about Streams instances, see the InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

## 8.1.2 Creating and configuring an instance

Complete the following steps to create a Streams instance:

1. In the First Steps page, click **Create and manage InfoSphere Streams Instance**. The Instance Manager application appears, as shown in Figure 8-2.

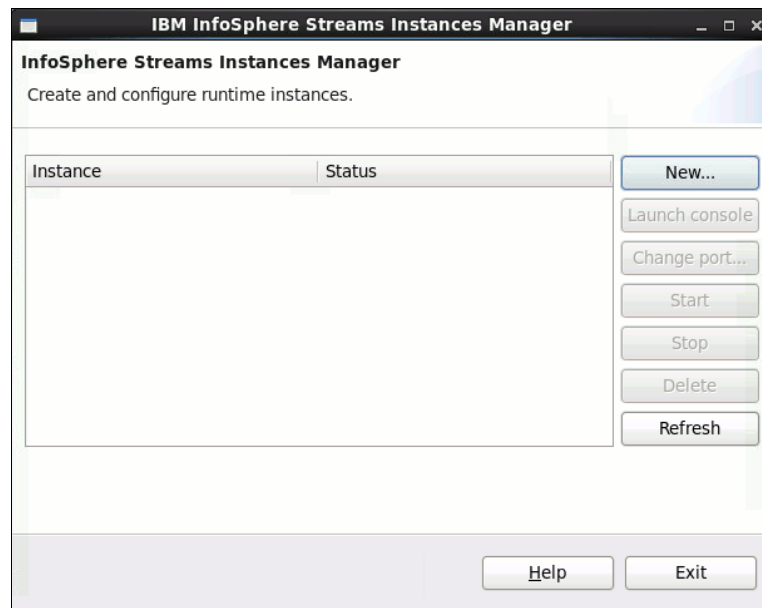


Figure 8-2 Streams Instances Manager

2. Click **New**.
3. Enter the instance name, as shown in Figure 8-3.

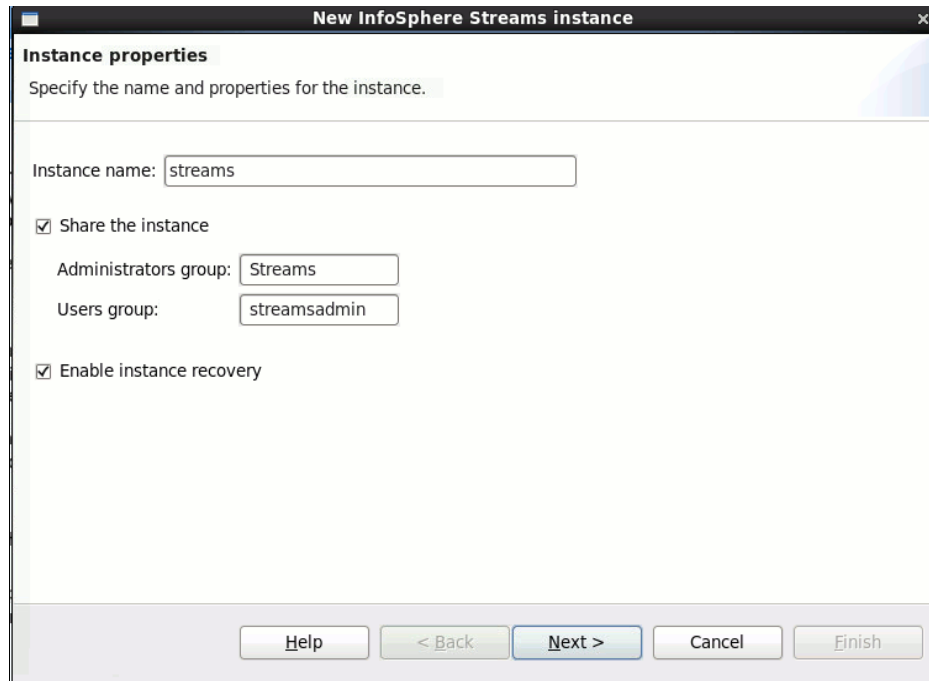


Figure 8-3 New instance properties options window

The instance name must satisfy the following requirements:

- The name must be unique among the instances that are defined by you, the instance owner.
- The name must contain alphanumeric characters and begin with an alphabetic character. Each of the alphanumeric characters must be one of the following ASCII characters: 0 - 9, A - Z, or a - z.

The name can contain a dash (-), an underscore (\_), or a period (.).

- Share the instance

InfoSphere Streams uses access control lists (ACLs) to enforce security. An ACL is composed of the type of instance object to secure and the actions that a group or user is authorized to perform against the object. The ACLs are initialized when you create an instance.

For a shared instance, InfoSphere Streams uses a shared authorization policy template that is optimized for use by several users. The administrators group and an optional users group have rights to a shared instance. The shared authorization policy template ensures that users that belong to an administrators group can access administrative functions, and that users that belong to a users group can work with their application jobs.

The group must exist in your user authentication service repository for the instance to function correctly. You can use Pluggable Authentication Module (PAM) or an external Lightweight Directory Access Protocol (LDAP) server to perform user authentication for InfoSphere Streams. By default, InfoSphere Streams uses the PAM authentication service.

After the instance is created, you can view and manage the access permissions for the instance by using the Streams Console or the streamtool **getac1** and **setac1** commands.

To enable instance users to sign in by using their RSA Security keys, you must manually copy the public keys of the users to the directory that is identified by the SecurityPublicKeyDirectory instance property.

- Enable instance recovery

To enable the recovery of failed runtime components, InfoSphere Streams can work with an IBM DB2 database server to store runtime data in a recovery database. If recovery is enabled, the administrator can move or restart failed runtime services while the instance is running. Otherwise, the administrator must shut down and restart the instance if a runtime service fails.

**Recovery option:** The InfoSphere Streams recovery database must be configured to enable the recovery option. For more information, see 8.5, “Streams Recovery Database” on page 247.

4. Click **Next**.
5. Add or select at least one host for the instance, as shown in Figure 8-4.



Figure 8-4 The Instance hosts configuration window options

To add hosts, you specify the host name or IP address, or select from a list of available hosts. You define the available hosts by using a host file or a streams\_get\_available\_hosts script.

### **Configuring an available host file for InfoSphere Streams**

To define a list of available hosts and their host tags, you can configure an available host file for InfoSphere Streams, as shown in Example 8-2 on page 204. This host file is named hostfile and must be in the ~installation-owner/.streams/config/ or ~/.streams/config/ directory.

When you install InfoSphere Streams, the installation utility attempts to create a default host file in the `~installation-owner/.streams/config/` directory if a directory does not exist on your system. The default host file contains a single host, which is the host where the installation is performed. This host is tagged as a build and execution host.

**Important:** The available host file and the host file that is specified on the `streamtool` command `--hfile` option do not use the same syntax and are not interchangeable.

*Example 8-2 An InfoSphere Streams available host file*

---

```
# A line whose first non-whitespace character is '#' is a comment.
# A blank or whitespace-only line is also a comment.
# Host declaration lines are of the form: <hostid> [, <csv_hosttag_list>].
# A <hostid> can be a host name or an IP address as needed in your environment.
# The host name or IP address must not resolve to a loopback address
(127.*.*.*).
# A host declaration that does not have a host tag list is returned for all
queries.
# If you want a host to be ignored, add a '#' at the start of the host
declaration line.
host1, build, execution
host2, execution
host3
host4, build
```

---

**Important:** If a host in the default host file does not meet the name resolution requirements for InfoSphere Streams or you do not want to include the host in your instance configuration, you can modify the default host file. For more information about host requirements, search for “name resolution requirements” at the Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp?>

You can use host tags to identify hosts that have different physical characteristics or logical uses. You can use the host tags that are defined by InfoSphere Streams or create your own.

The following host tags are defined by InfoSphere Streams:

- `build`: A build host is one that includes tools, such as compilers, linkers, and libraries that are necessary to build an SPL application. The SPL compiler uses one or more of these hosts when remote parallel builds are performed.
- `execution`: An execution host can be used to run SPL applications. These hosts are the hosts that can be included in a runtime instance. To assign or define host tags, use the Assign host tags option.

- Select one or more Hosts for running applications and also assign each service of each host, as shown in Figure 8-5.

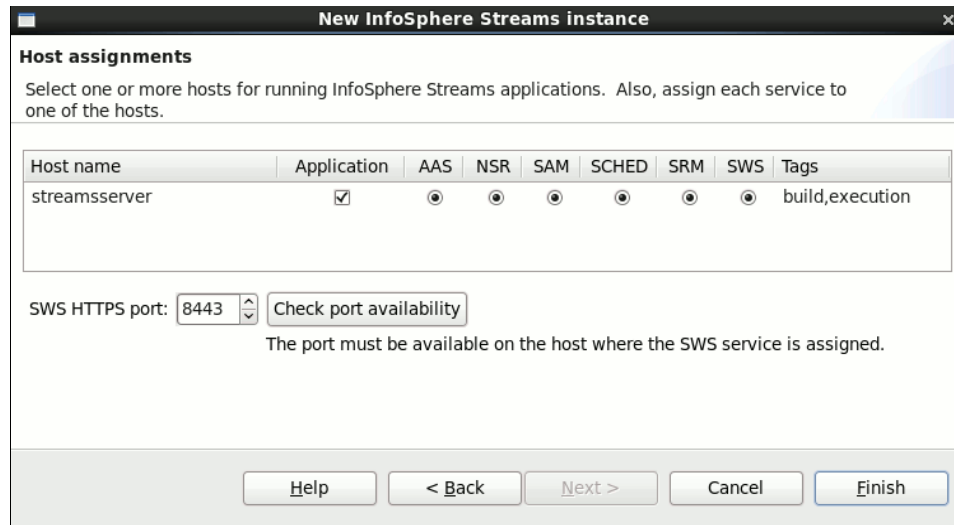


Figure 8-5 Host assignments window

The following types of InfoSphere Streams hosts are available:

- Application: A host that runs applications only.
- Management: A host that runs management services only. Management services include the AAS, NSR, SAM, SCHED, SRM, and SWS services.
- Mixed: A host that runs management services and applications.

The preferred configuration for optimal performance is to reserve one host exclusively for InfoSphere Streams management services and use one or more hosts exclusively for running applications. Sharing hosts across multiple instances can also affect performance.

**Restriction:** If you selected the Enable instance recovery option for the instance, the name service (NSR) cannot be assigned. In this case, InfoSphere Streams uses the shared file system for the instance.

The Streams Console is accessed by using the SWS HTTPS port number that you specify. A unique port is required for each instance that you own, regardless of with which installation (or version of Streams) the instance was created. The port cannot be in use when the instance is started. The **Check port availability** option can be used to see whether the port is active. This option is enabled only if SWS is selected to be run on the same host on which the new function is run.

7. Click **Finish**. The new instance now is listed and available in the Console in a Stopped state, as shown in Figure 8-6.

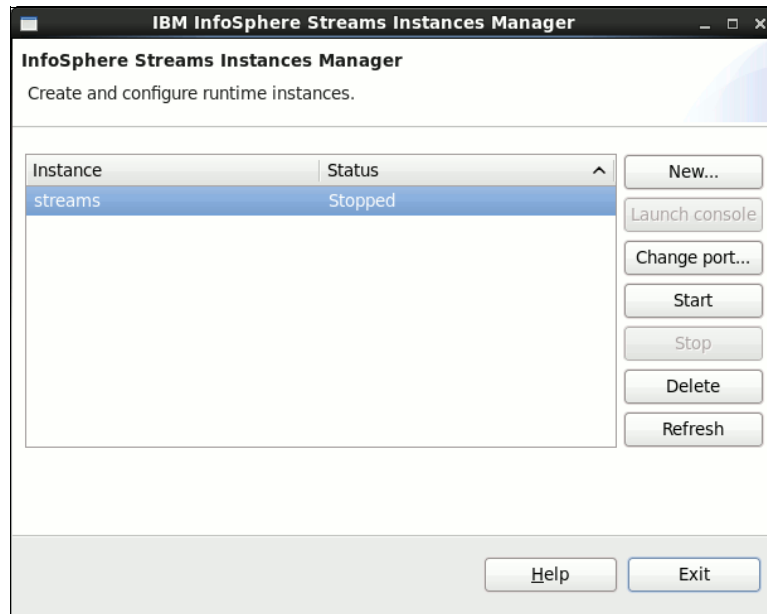


Figure 8-6 A new instance that is ready to be used

8. Click **Start** to start the new instance.

Your new instance is ready to be used when you deploy your Streams applications.

### **Launch console**

The Streams Console is a web-based GUI with which you can monitor and manage instances. You can use this application to update the configuration of an existing instance, and to start, stop, and delete instances. To launch the Streams Console, use the Launch console option. Before you can launch the Streams Console, the instance must be started. The console launches in your default web browser.

For more information, see 8.2, “InfoSphere Streams Console” on page 208.

### Change port

The SWS HTTPS port for the Streams Console is specified when you create an instance. To change the port number after an instance is created, use the Change port option, as shown in Figure 8-7.

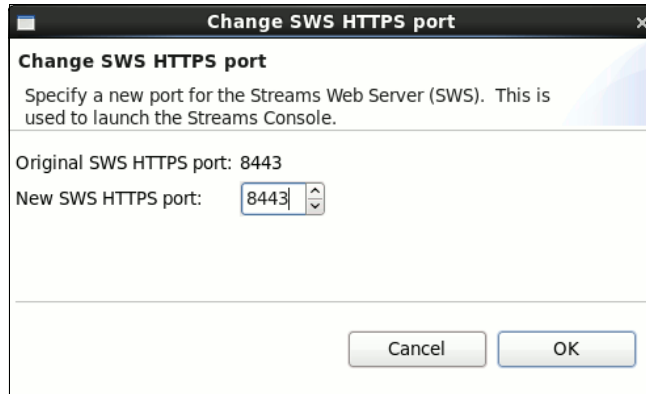


Figure 8-7 Port configuration

A Secure Shell (SSH) environment must be configured before you begin. You can use the First Steps application to configure the SSH environment by clicking **Configure the SSH environment**.

**Tip:** If you encounter errors when an instance is stopped, the application provides an option to ignore the errors and force the instance to stop. Select the **Force instance to stop** option in the Stop confirmation dialog, as shown in Figure 8-8.

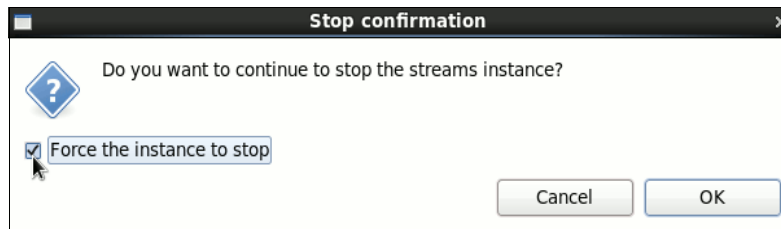


Figure 8-8 Shows the force instance to stop option

## 8.2 InfoSphere Streams Console

You can use the Streams Console to monitor and manage your instances and applications from any computer that includes a web browser and HTTP connectivity to the server that is running the SWS.

By default, the SWS is included on the host configuration when you create an instance by using the `streamtool mkinstance` command. The default SWS HTTPS port that is used is 8443. The use of port 8443 might not work for some deployments because other web servers and applications also use 8443 as their default port.

**Port is in use:** If you try to start an instance and the port is used, Streams issues a message that the port is in use and the instance startup fails.

You can specify a different port number when you create an instance or after the instance is created by using a `setproperty` or `mkinstance` command with the property option, as shown in Example 8-3.

*Example 8-3 The `setproperty` command for changing an SWS port*

---

```
streamtool setproperty -i <instance> --property SWS.httpsPort=port-number
```

---

**Important:** You can use the Instances Manager to create an instance and specify the port and to change the SWS port, as described in 8.1.1, “Instances Manager” on page 201.

To start the Streams Console, enter the command that is shown in Example 8-4.

*Example 8-4 Command that is used to start the instance*

---

```
streamtool startinstance -i instance-id
```

---

After the instance is started, enter the command that is shown in Example 8-5 to ensure that SWS is running on a host in the Streams instance.

*Example 8-5 The command that is used to retrieve the instance running state*

---

```
streamtool getresourcestate -i instance-id
```

---

The command output example that is shown in Example 8-6 shows that SWS is running on host.

*Example 8-6 SWS status output information*

---

```
Host / Status / Schedulable / Services / Tags  
host1 RUNNING yes RUNNING:aas,hc,nsr,sam,sch,srm,sws execution...
```

---



## 8.2.1 Starting the InfoSphere Streams Console

To open the Streams Console in your default web browser, you can use the **streamtool Launch** command or the Instances Manager by using the streamtool launch command that is shown in Example 8-7.

*Example 8-7 Command that is used to launch the Streams Console*

```
streamtool launch -i instance-id
```

You can also use the Instances Manager application to start the Streams console, as described in “Launch console” on page 206.

Figure 8-9 shows the Streams Console login window.

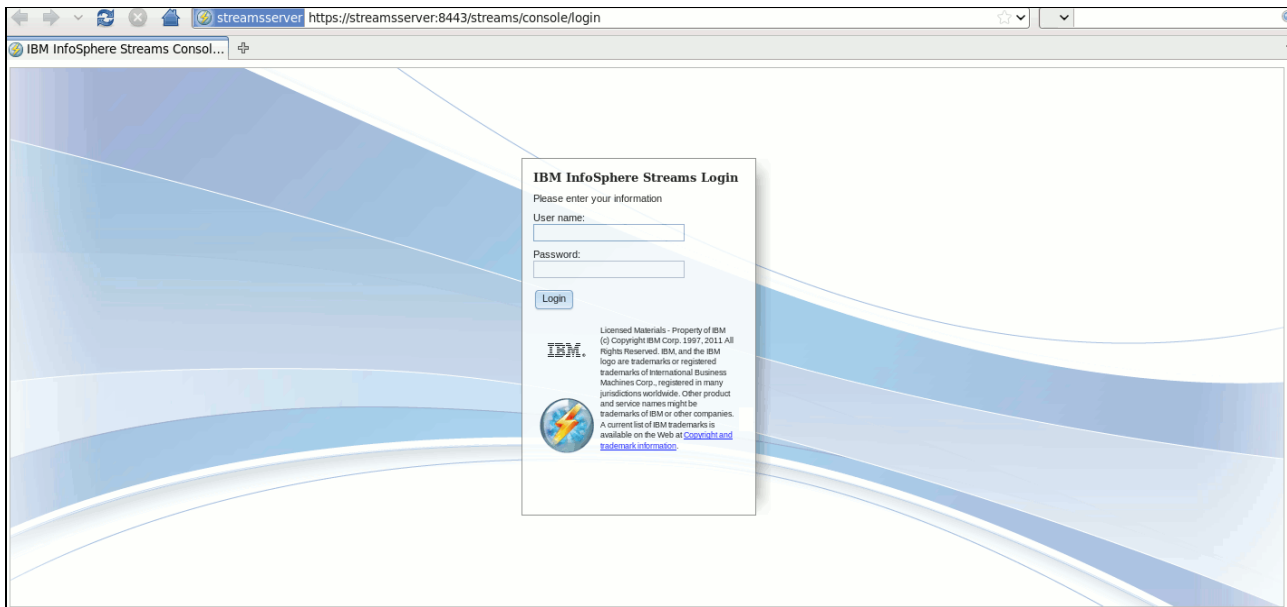


Figure 8-9 Streams Console login window

Log in to the Streams Console by using the application administrator ID and password that you selected when you installed the product.

**For more information:** For more information about creating and managing server certificates and setting up client authentication for the Streams Console, see the InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

## 8.3 Instance administration

The Streams Console provides general information about the InfoSphere Streams instance and the overall status of instance hosts, runtime services, and application jobs. You can also use the Console to view the topology of applications that are running in the instance, download a log, and trace data.

**Important:** You can work with only one InfoSphere Streams instance at a time by using the Streams Console. Each instance features its own console.

After you log in to the Streams Console, you are redirected to the Instance Status page, as shown in Figure 8-10.

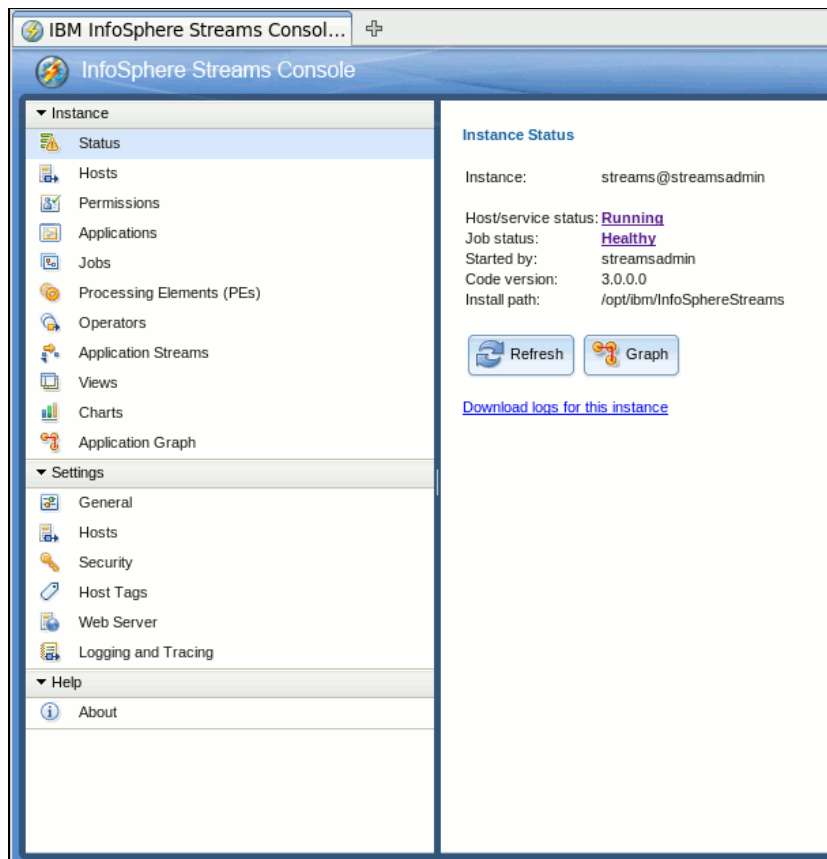


Figure 8-10 Instance status page

The Instance Status page shows the health of the jobs that are running in the instance. It shows only those jobs for which the user is authorized.

**Important:** To show the functions of the Streams Console, we are running the Commodity Purchase demonstration application that is shipped with the product. For more information, see Appendix C, “Commodity purchasing application demonstration” on page 283.

The application status is similar to the information that is shown in the following example:

```
Instance: streams@streamsadmin
Host/service status: Running
Job status: Healthy
Started by: streamsadmin
Code version: 3.0.0.0
Install path: /opt/ibm/InfoSphereStreams
```

If you select the Host/service status information in **Running** option, you are redirected to the Hosts page. Selecting the **Healthy** option in Job status, you are redirected to the Jobs page.

The following Host/service status values are available:

- ▶ **Running:** All instance hosts are running in a normal state.
- ▶ **Partially running:** At least one host in the instance is running normally, but at least one host in the instance is only partially active.
- ▶ **Partially failed:** At least one host in the instance is running normally, but at least one host in the instance is down or experiencing problems.
- ▶ **Failed:** All instance hosts are down.
- ▶ **Starting:** The instance is in the process of starting.
- ▶ **Stopping:** The instance is in the process of stopping.

The Graph option redirects you to the Application Graph page. It provides a graphical representation of the applications that are running in the instance and their status.

**Tip:** If the instance contains many running jobs or jobs that have many PEs and operators, you might want to use the Jobs page to view a graphical representation of a subset of the jobs that are running in the instance.

When you select **Download logs for this instance**, you are prompted to save the log and trace data in a compressed tar file, as shown in Figure 8-11.

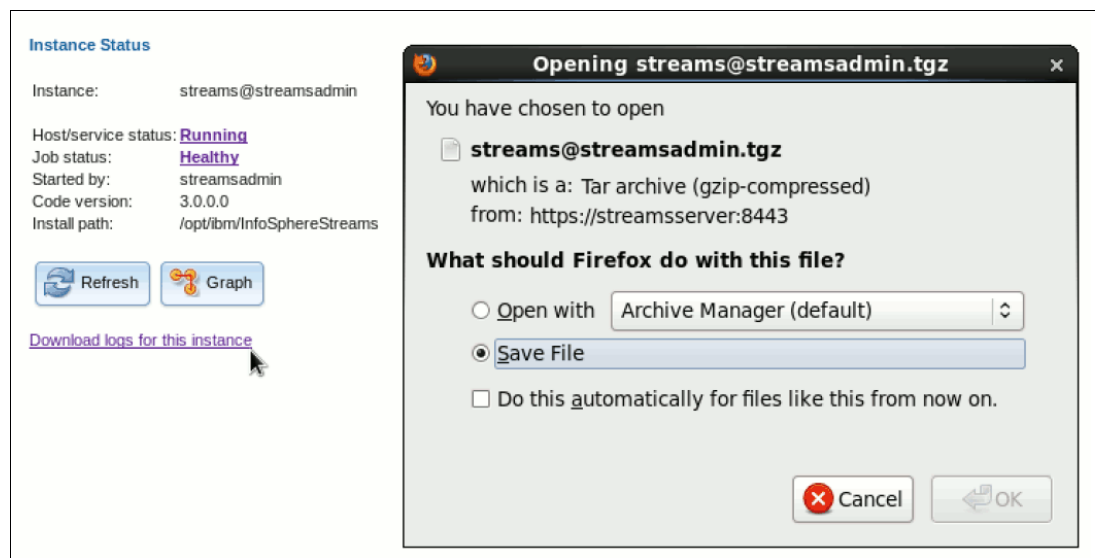


Figure 8-11 Log file download option window

The log and trace data in the compressed tar file is the same as the data that is collected by the streamtool `getlog` command.

### 8.3.1 Hosts

Use the Hosts page to display the overall status of hosts in the Streams instance. It is used to display the status for a list of hosts.

The page includes the services that are running on each host and the status of each service, as shown in Figure 8-12.

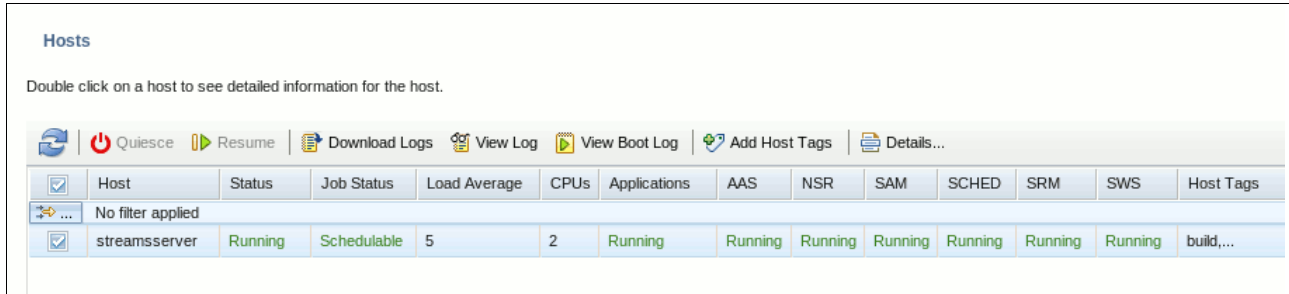


Figure 8-12 Hosts page with the running services

The following possible Host status values are available:

- ▶ Running: All services that are configured on the host are active.
- ▶ Partially running: At least one service that is configured on the host is active, but not all configured services are active.
- ▶ Failed: None of the services that are configured for the host are active.
- ▶ Quiescing: A transitional state that occurs while a host is quiesced (temporarily taken offline).
- ▶ Quiesced: The host is still active, but is in offline mode, which means that jobs are no longer scheduled to run on this host.
- ▶ Starting: A transitional state that occurs when an instance is starting, the host is added to the instance, or the host is resumed.
- ▶ Stopping: A transitional state that occurs when an instance is stopping or the host is removed from the instance.

Double-click a host in the list to view detailed information for that host, as shown in Figure 8-13.

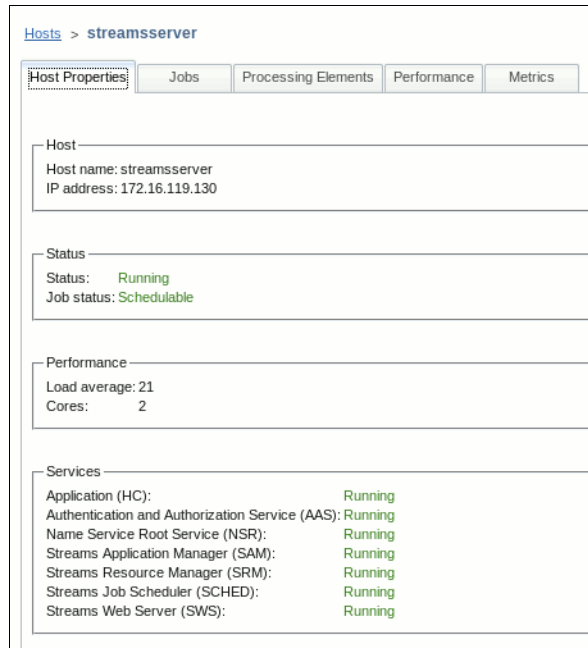


Figure 8-13 Detailed information about the selected Host

Click **Hosts** in the left panel to return to the main hosts list.

## Hosts page options

The following controls for the hosts list management are available, as shown in Figure 8-12 on page 212:

- ▶ Refresh: Update all of the contents of this page.
- ▶ Quiesce: Quiesce the currently selected host or hosts. Quiescing a host means to shut down the application service and PE processing on the host in a controlled way. When possible, the running jobs or PEs are relocated to a different application host within the instance.

**Important:** Hosts that are running any services other than the Application service cannot be quiesced.

- ▶ Resume: Resume operations on the selected host or hosts. This action restarts the application service processing and makes the host available for scheduling jobs or PEs.

**Important:** Only hosts with a quiesced status are valid for a resume operation.

- ▶ Download Logs: Download the logs for the selected host or hosts. The log file is downloaded as a tar-gzip (.tgz) file.
- ▶ View Boot Log: View the boot log for the selected host. The boot log is displayed as a text file within the browser.
- ▶ Add Host Tags: Add tags to the selected host or hosts. The host tags are used by the Streams job scheduler to determine which host can run portions of a submitted job.

## Job status

This status indicates the ability of the host to accept new jobs.

The following Job Status values are used:

- ▶ **Schedulable:** The host is ready to accept new job submissions. (This status is the normal state.)
- ▶ **Failed:** The application service that failed on this host.
- ▶ **Starting:** The application service is starting on this host.
- ▶ **Degraded:** The application service is not working properly on this host.
- ▶ **Draining:** The host is performing a controlled shutdown as a result of a quiesce or remove operation.
- ▶ **Stopped:** The host application service was stopped.
- ▶ **Not applicable:** The host is a Management-only host and cannot be scheduled.

## Load average

The value that is displayed is a Linux host load average, which is adjusted for InfoSphere Streams usage. The adjusted load average also takes into account how many CPUs are on the host.

For more information about the host load average calculation, see the InfoSphere Streams Jobs scheduler documentation.

## CPUs

The number of CPUs (cores) on this host.

## Services

The following services can be configured to run on a host:

- ▶ Application
- ▶ AAS (Authentication and Authorization Service)
- ▶ SAM (Streams Application Manager)
- ▶ SCHED (Job Scheduler)
- ▶ SRM (Streams Resource Manager)
- ▶ SWS (Stream Web Server)

Each service can have the following state:

- ▶ **Running:** The service is available for processing requests.
- ▶ **Down:** The service is not available for processing requests.
- ▶ **Degraded:** The service is available for processing requests, but recently failed a check for availability.
- ▶ **Starting:** A transitional state is used when the instance is starting, a host is added to the instance, or the service is added to the host.
- ▶ **Stopping:** A transitional state that is used when the instance is stopping, a host is removed from the instance, the host is quiesced, or the service is removed from the host.
- ▶ **Not started:** The service was ended by quiescing the host or removing the service from the host.

The state might be blank if a service is not configured to run on this host.

## Host Tags

The list of host pool tags that are associated with this host. You can use host tags to identify hosts that have different physical characteristics or logical uses. The Streams job scheduler uses host tags to match hosts against host pool requirements that are specified in Streams processing language (SPL) applications.

### 8.3.2 Permissions

Use the Permissions page, as shown in Figure 8-10 on page 210, to configure the permissions or access rights to services and jobs within the InfoSphere Streams instance.

**Private instance:** If the user setup a private instance, adding a user or group here makes it a shared instance.

#### ► Objects tab

The objects are displayed as a tree or list on the left side of the page. The tree shows the hierarchical relationship of the objects. The list view shows the objects in a sortable list and can be used to find an object alphabetically if the list is large.

Use the **List view** and **Tree view** options at the bottom of the tab to switch between the list and tree views of the objects, as shown in Figure 8-14.

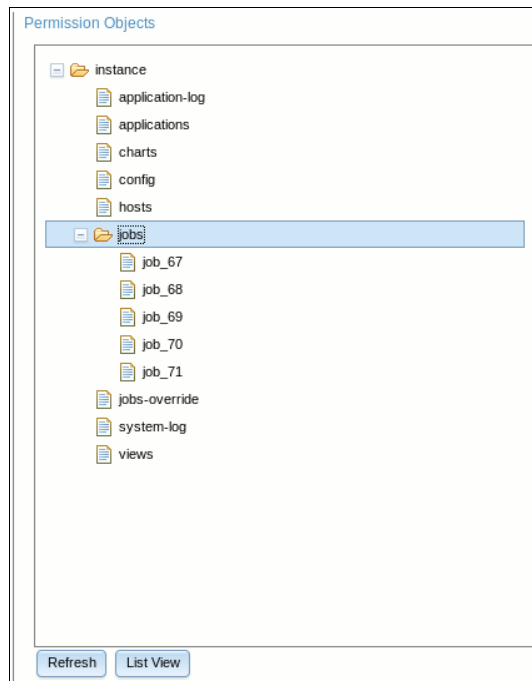


Figure 8-14 Objects in a tree view list

To manage a user's permissions, select the object and then the user, as shown in Figure 8-15.

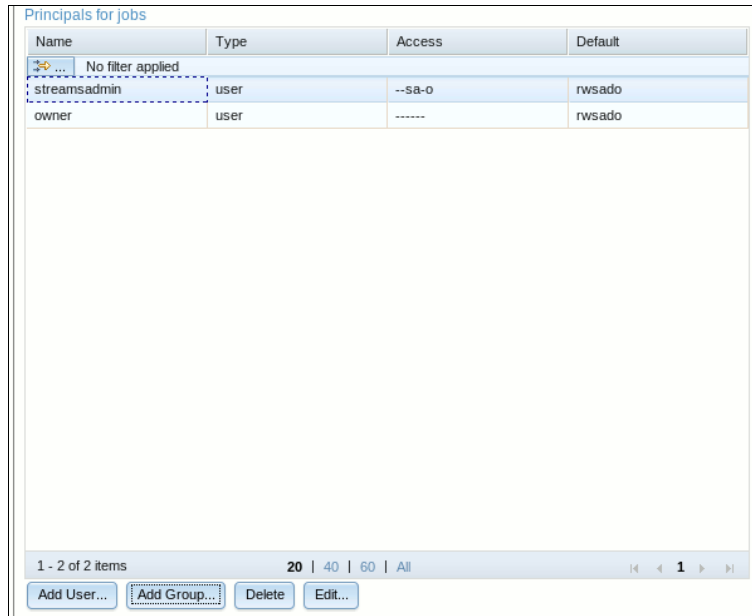


Figure 8-15 Object permissions control window

Click **Add User** to add user-level privileges for the selected object.

Click **Add Group** to add group-level privileges for the currently selected object.

Click **Delete** to delete the selected user or group privilege for the selected object.

Click **Permissions** to display or modify the selected user's or group's permissions to the selected object, as shown in Figure 8-16.

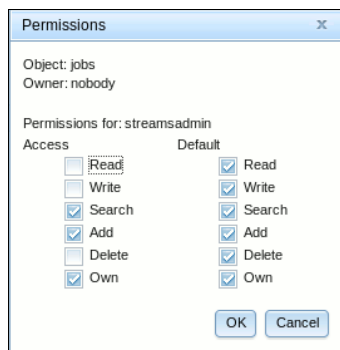


Figure 8-16 Permissions selections window



## Object details

Objects feature the following details:

► Access

The access permission for the selected user or group for the selected object. Defines how the access permissions are used by the components that use the object.

► Default

The default permission of the selected user or group for the selected object. Default permissions are used when a child object is created. The child object inherits the parent object's default permissions at creation time.

For example, complete the following steps to grant a user access to this instance:

1. Select **Instance** from the object tree or list.
2. Click **Add user** and add the user to the users and groups list.
3. Select the user in the users and groups list
4. Click **Permissions**.
5. Select the permissions to grant the user. To give a user access to an instance, grant the user the Search permission.

## Users and groups tab

By using this tab, you can manage Streams permissions by a user or group.

► Users and groups:

The users and groups list shows the list of users and groups that have permissions in this instance. To manage the permissions for a user or group, select a user or group from this list, and update the permissions in the permissions list. See in the Figure 8-17 on page 217.

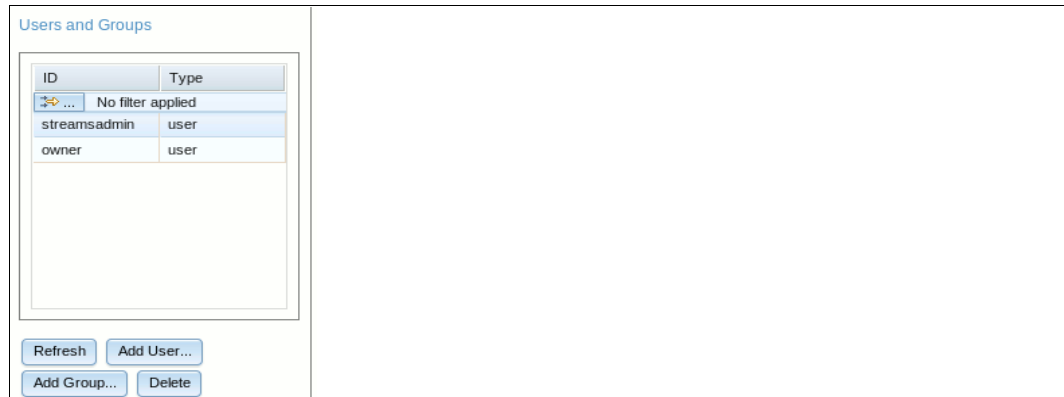


Figure 8-17 Users management controls of the Permissions page

The following controls for the permissions management are available:

- ▶ Refresh
 

Refresh the users and groups list from the server.
- ▶ Add User
 

Add a user to this instance. The new user has Search permission to the instance permission object by default.
- ▶ Add Group
 

Add a group to this instance. The new group has Search permission to the instance permission object by default.
- ▶ Delete
 

Delete a user or group from the list. All permissions for the user or group are removed from the instance.
- ▶ Permissions
 

Permissions for selected user or group are listed in the permissions table. The permissions are grouped by permission objects, with access permissions for each object that is listed in a row, and default permissions that are listed in the following row as shown in Figure 8-18 on page 218.

For more information about access and default permissions, see the Object Details section.

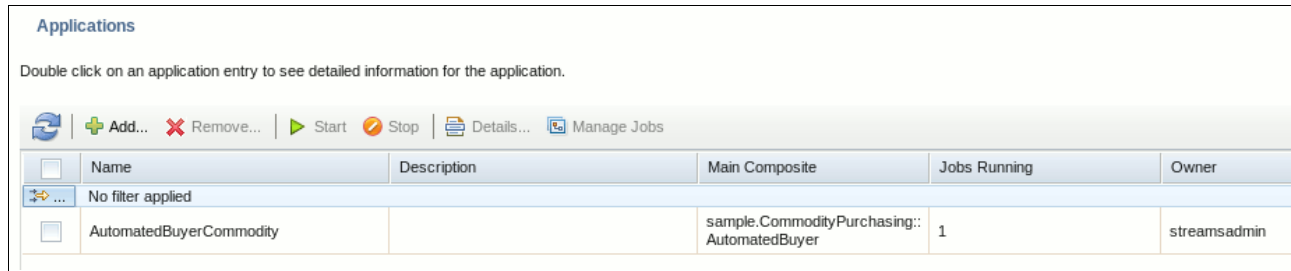
Figure 8-18 shows the permissions for a user or group. Explicit permissions shows what was set directly for the user or group. Resolved permissions show the merging of user permissions and any group permissions that apply to the user.

Explicit Permissions		Resolved Permissions	
<b>application-log</b>			
Access	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
Default	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
<b>applications</b>			
Access	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
Default	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Write	<input type="checkbox"/> Search
<b>charts</b>			
Access	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
Default	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Write	<input type="checkbox"/> Search
<b>config</b>			
Access	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
Default	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
<b>hosts</b>			
Access	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
Default	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
<b>instance</b>			
Access	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
Default	<input type="checkbox"/> Read	<input type="checkbox"/> Write	<input type="checkbox"/> Search
<b>job_67</b>			
Access	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Write	<input checked="" type="checkbox"/> Search
Default	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Write	<input checked="" type="checkbox"/> Search

Figure 8-18 Permissions controls of a user

### 8.3.3 Applications

Use the Applications page to access and manage the Streams Console application repository as shown in Figure 8-19.



<input type="checkbox"/>	Name	Description	Main Composite	Jobs Running	Owner
<input type="checkbox"/>	No filter applied				
<input type="checkbox"/>	AutomatedBuyerCommodity		sample.CommodityPurchasing::AutomatedBuyer	1	streamsadmin

Figure 8-19 Instance's deployed applications list

The application repository for an InfoSphere Streams instance contains references to applications that are in the file system. You can use the Streams Console to submit jobs for applications that are in the repository. If an application includes submission time parameters, you can modify and save parameter values for later use.

You can add applications to the repository by using one of the following ways:

- ▶ Use the Streams Studio Publish to Streams Console application repository option. By using this option, the application is published directly into the application repository.
- ▶ Use the Streams Studio Publish or Export option to create a .zip file that contains the application information. By using this option, you must import the information in the application .zip file into the repository by using the Add option.

If the Streams Console is configured to use client authentication for the Streams instance, you must create a Streams Studio SSL client authentication keystore before you can add applications to the repository by using Streams Studio. Client authentication is not enabled for the Streams Console by default.

**Important:** For more information about the use of Streams Studio to publish applications or create an application .zip files, see “Publishing SPL applications” in the InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

For more information about setting up client authentication, see the Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

When you add an application to the repository, you add a reference to the application that is in the file system. If an application is changed, you must use Streams Studio to update the application and then add the application to the repository again.

To give other users access to applications in the repository, use the Streams Console Permissions page.

The Applications page displays the following information about each application:

- ▶ **Name**

The application name. To change the application name, click **Details**. This name must satisfy the following requirements:

  - Contains a maximum of 128 characters.
  - Can contain alphanumeric characters, spaces, periods, and colons.
  - Can contain parentheses. No other special characters are allowed.
- ▶ **Description**

The application description, which is optional. This description can contain a maximum of 150 characters.
- ▶ **Main Composite**

The main composite operator for this application. The main composite name is in the InfoSphere Streams application.
- ▶ **Jobs Running**

The number of jobs that are running in the application. This number increases when a new job is started and decreases when a job is stopped.
- ▶ **Owner**

The owner of the application in the application repository. The owner is the user who imported the application into the repository.

## Applications control options

The following controls are available for application management:

- ▶ **Refresh**

Refresh the list of applications to display the current list of applications and their states.
- ▶ **Add**

Add an application to the repository.

**Important:** The application information must be in a .zip file that you create by using the Streams Studio Publish or Export option. The combination of the application name, description, and path of the application description language (.ad1) file must be unique.

- ▶ **Remove**

Remove this application from the repository. This action removes the reference to the application in the Streams Console repository. It does not remove the application from the file system.
- ▶ **Start**

Start a job for this application. If the application includes submission time parameters, you are prompted for parameter values before the job is submitted to the InfoSphere Streams instance.
- ▶ **Stop**

Stop all jobs for this application.
- ▶ **Details**

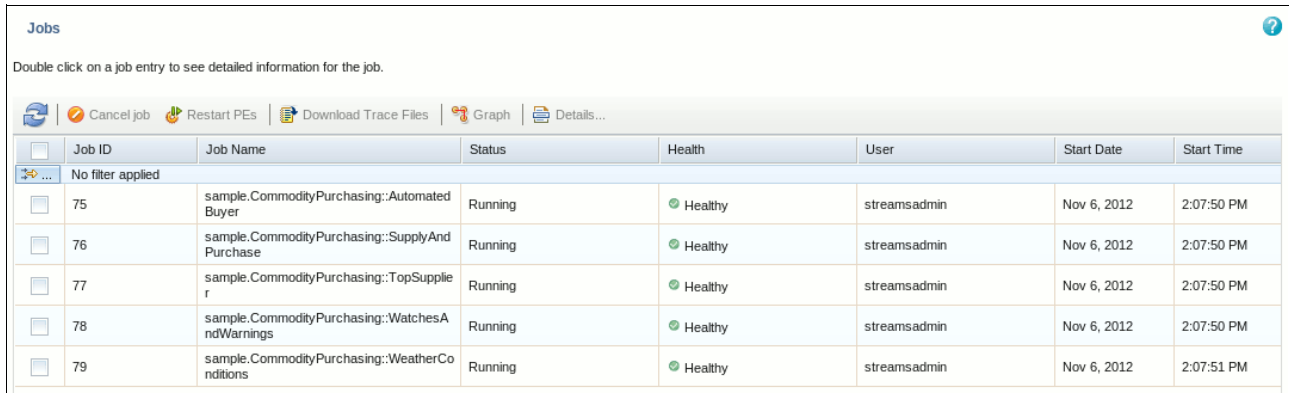
Display the application repository ID, name, description, and submission time parameters for this application. You can modify the application name, description, and submission time parameter values.

► Manage Jobs

Display a list of the jobs that are running in this application. For more information about the actions that you can perform on jobs, see the Streams Console Jobs page.

### 8.3.4 Jobs

Use the Jobs page to view detailed information about the application jobs in the InfoSphere Streams instance and work with those jobs. You also can use this page to view the topology of applications that are running in the instance, and to download trace data for jobs, as shown in Figure 8-20.



Job ID	Job Name	Status	Health	User	Start Date	Start Time
75	sample.CommodityPurchasing::Automated Buyer	Running	Healthy	streamsadmin	Nov 6, 2012	2:07:50 PM
76	sample.CommodityPurchasing::SupplyAnd Purchase	Running	Healthy	streamsadmin	Nov 6, 2012	2:07:50 PM
77	sample.CommodityPurchasing::TopSupplier	Running	Healthy	streamsadmin	Nov 6, 2012	2:07:50 PM
78	sample.CommodityPurchasing::WatchesAndWarnings	Running	Healthy	streamsadmin	Nov 6, 2012	2:07:50 PM
79	sample.CommodityPurchasing::WeatherConditions	Running	Healthy	streamsadmin	Nov 6, 2012	2:07:51 PM

Figure 8-20 Application Jobs list

The jobs list includes only the jobs that the current user is authorized to work with. To sort the jobs list, click any column heading in the table. To view detailed information about a job, double-click the job in the list.

#### Jobs control options

The following controls are available for jobs management:

► Refresh icon

Retrieve a snapshot of the jobs that are active. The filter criteria are used when the jobs list is refreshed.

► Cancel job

Cancel the selected job or jobs.

► Restart PEs

Restart the processing elements (PEs) that are associated with the selected job or jobs. Only restartable PEs with a status of Ended are restarted. Running PEs are not restarted.

► Download Trace Files

Download the trace data for the selected job or jobs. InfoSphere Streams saves the trace data in a compressed tar file.

**Tip:** The trace data in the compressed .tar file is the same as the data that is collected by the `streamtool getlog` command.

► Graph

View a graphical representation of the selected job or jobs, and their status.

## 8.3.5 Processing Elements

The PE page is used to display and work with a list of PEs. The page functions allow PEs to be stopped and restarted and PE logs to be downloaded or viewed directly in the browser. The list of PEs can be sorted by clicking any column heading within the appropriate table.

Double-click a PE to view detailed information and work with a specific PE, as shown in Figure 8-21.

ID	Status	Reason	Health	Job Name	Operators	Restartable PE	Moveable PE	Host	Process ID	Launch Count	Job ID
0	Running		Healthy	sample.CommodityPurchasing::AutomatedBuyer	TopSupplierAndCurrentSupply,...	false	false	streamsserver	3828	1	0
1	Running		Healthy	sample.CommodityPurchasing::SupplyAndPurchase	PurchaseSensorFeed,...	false	false	streamsserver	3829	1	1
2	Running		Healthy	sample.CommodityPurchasing::TopSupplier	WeatherScoringAttributes,...	false	false	streamsserver	3838	1	2
3	Running		Healthy	sample.CommodityPurchasing::WatchesAndWarnings	WatchesWarningsSource,...	false	false	streamsserver	3839	1	3
4	Running		Healthy	sample.CommodityPurchasing::WeatherConditions	SupplierLocation,...	false	false	streamsserver	3849	1	4

Figure 8-21 Processing Elements list

The following views also are available for the PEs list:

- ▶ The Performance is a list of restart recommendations, in priority order. If PEs are marked as moveable/restartable and they are not performing well, an administrator can choose to restart them on a less used host in the instance, as shown in Figure 8-22.

PE ID	Restart Priority	Job Name	Operators	Host	CPU %	Load Average	# of candidate hosts	Restartable PE	Moveable PE	Job ID
1	None	sample.CommodityPurchasing::SupplyAndPurchase	PurchaseSensorFeed,...	streamsserver	5	4	0	false	false	1
0	None	sample.CommodityPurchasing::AutomatedBuyer	TopSupplierAndCurrentSupply,...	streamsserver	5	4	0	false	false	0
3	None	sample.CommodityPurchasing::WatchesAndWarnings	WatchesWarningsSource,...	streamsserver	5	4	0	false	false	3
4	None	sample.CommodityPurchasing::WeatherConditions	SupplierLocation,...	streamsserver	10	4	0	false	false	4
2	None	sample.CommodityPurchasing::TopSupplier	WeatherScoringAttributes,...	streamsserver	5	4	0	false	false	2

Figure 8-22 List of PEs in Performance view mode

- ▶ The Metrics tab, as shown in Figure 8-23.

	ID	Job	CPU (ms)	CPU/Sec	Tuples Out	Tuples Out/Sec	Tuples In	Tuples In/Sec	Bytes Out	Bytes Out/Sec	Bytes In	Bytes In/Sec	Memory Consumption (KB)	Host	Stale
No filter applied	0	0	1130	0	4	0	60	0	52	0	1206	1	683312	streamserver	false
	1	1	1480	1	12	0	60	0	239	0	1174	1	906072	streamserver	false
	2	2	1470	1	52	0	76	0	1070	1	3963	3	829308	streamserver	false
	3	3	1600	1	24	0	36	0	632	0	1395	0	834556	streamserver	false
	4	4	75500	1	48	0	0	0	3228	3	0	0	910376	streamserver	false

Figure 8-23 List of PEs in the Metrics view mode

## Processing Elements control options

The following controls are available for PE management:

- ▶ Refresh
  - Retrieve a snapshot of the active PEs. The filter criteria is used during the refreshing of the PEs list.
- ▶ Stop
  - Stop the selected PE or PEs.
- ▶ Restart
  - Restart the selected PE or PEs.
- ▶ Download Trace Files
  - Download the trace files for the selected PE or PEs. The files are downloaded as a tar-gzip (.tgz) file.
- ▶ View Trace
  - View the trace data for the selected PE. The trace data is displayed as text in a new browser window.
- ▶ View Console Trace
  - View the console trace data for the selected PE. The console trace data is the standard out for the PE process.

► Details page

Displays detailed information and work with a specific PE, as shown in Figure 8-24.

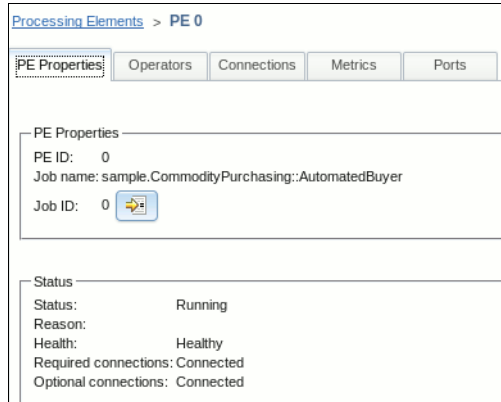


Figure 8-24 Detailed information about a PE

### 8.3.6 Operators

The Operators page is used to display and work with a list of operators. The Operators page function allows viewing application data logs for an operator. The list of operators can be sorted by clicking any column heading within the table, as shown in Figure 8-25.

Operators

Operators Metrics

Double click on an operator to see detailed information for the operator.

View Trace Details...

	Operator Name	PE Status	Job Name	Job ID	PE ID	Host
...	No filter applied					
<input type="radio"/>	TopSupplierAndCurrentSupply	Running	sample.CommodityPurchasing::AutomatedBuyer	0	0	streamserver
<input type="radio"/>	PurchaseOpportunity	Running	sample.CommodityPurchasing::AutomatedBuyer	0	0	streamserver
<input type="radio"/>	AutomatedPurchases	Running	sample.CommodityPurchasing::AutomatedBuyer	0	0	streamserver
<input type="radio"/>	PurchaseSensorFeed	Running	sample.CommodityPurchasing::SupplyAndPurchase	1	1	streamserver
<input type="radio"/>	Purchases	Running	sample.CommodityPurchasing::SupplyAndPurchase	1	1	streamserver
<input type="radio"/>	PurchaseTransactions	Running	sample.CommodityPurchasing::SupplyAndPurchase	1	1	streamserver
<input type="radio"/>	AveragePurchaseWeatherScore	Running	sample.CommodityPurchasing::SupplyAndPurchase	1	1	streamserver
<input type="radio"/>	ConsumptionSensorFeed	Running	sample.CommodityPurchasing::SupplyAndPurchase	1	1	streamserver
<input type="radio"/>	CurrentStock	Running	sample.CommodityPurchasing::SupplyAndPurchase	1	1	streamserver
<input type="radio"/>	PurchaseAverage	Running	sample.CommodityPurchasing::SupplyAndPurchase	1	1	streamserver

Figure 8-25 Operators list



## Operators control options

The following controls are available for PE management:

- ▶ Refresh

Retrieve a snapshot of the active operators. The filter criteria is used during the refreshing of the operators list.

- ▶ View Trace

View the operator trace data. The operator trace file contains only application developer trace data; it does not contain Streams product trace data. The trace data is displayed as text in a new browser window.

Double-click an operator to view detailed information and work with a specific operator, as shown on Figure 8-26.

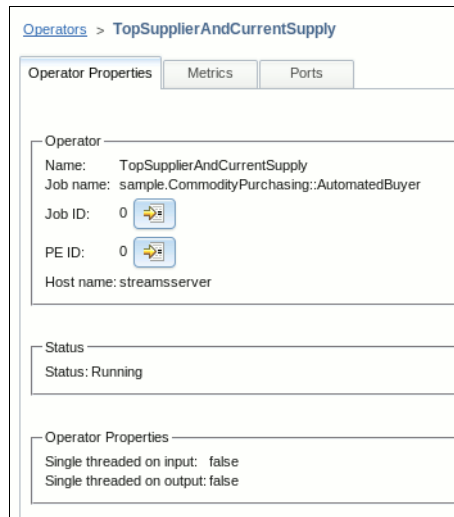


Figure 8-26 Detailed information for a selected operator

## 8.3.7 Application streams

Use the Application Streams page to display the streams that are exported or imported by the applications that are running in the InfoSphere Streams instance, as shown in Figure 8-27.

Export Job Name	Export Job ID	Export Operator Name	Export Properties	Export Filterable	Import Job Name	Import Job ID	Import Operator Name	Import Subscription	Import Filter	Import by Name
No filter applied										
sample.CommodityPurchasing::SupplyAndPurchase	1	SupplyExport		true	sample.CommodityPurchasing::AutomatedBuyer	0	CurrentStock			sample.CommodityPurchasing::SupplyAndPurchase.CurrentStock
sample.CommodityPurchasing::AutomatedBuyer	0	AutomatedPurchaseExport		true	sample.CommodityPurchasing::SupplyAndPurchase	1	AutomatedPurchases			sample.CommodityPurchasing::AutomatedBuyer.AutomatedPurchases
sample.CommodityPurchasing::WatchesAndWarnings	3	ExportAlerts		true	sample.CommodityPurchasing::TopSupplier	2	CurrentAlerts			sample.CommodityPurchasing::WatchesAndWarnings.CurrentAlerts

Figure 8-27 Application streams list

If an export or import features a matched pair, they appear in the same row of the table. If an export or import does not have a matched pair, the job might not be running or running correctly.

**Tip:** For more information about export and import options, see the descriptions of the Export and Import operators in the IBM Streams Processing Language Standard Toolkit Reference, which is available at this website:  
<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

### 8.3.8 Views

Use the Views page to create views for applications that you want to display in a chart or table format. You also can use this page to view or update the settings for existing views, as shown in Figure 8-28.

Name	Application	Operator Name	Port Name	Port ID	Owner	Started	Charts	Description
No filter applied								
Purchase View	sample.CommodityPurchasing::SupplyAndPurchase	PurchaseTransactions	PurchaseTransactions	0	streamsadmin	1	1	

Figure 8-28 View in a started state

Before you can create a chart or table, you must create at least one view by using this page. After you create one or more views, use the Charts page to create line charts, bar charts, and tables.

A view defines the set of attributes that can appear in a chart or table for a specific viewable data stream. A view also defines the buffering scheme and throttling rates for the visualization of the data. When you configure a view for an InfoSphere Streams application, you associate that view with a viewable stream of a running job. A viewable stream is the data stream that is produced by an output port that is defined as viewable in the InfoSphere Streams application. By default, all output port data streams are viewable.

#### Views control options

The following views are available for PE management:

- ▶ Refresh icon  
Update the contents of this page.
- ▶ Start  
Allow the view to collect data from the data streams. The view can provide data for a chart or table.

**Requirement:** At least one job that matches the application for the view must be running for the view to start. If more than one running job matches the application, you are prompted to select the job to associate with the view.

- ▶ **Stop**  
Do not allow the view to collect data from the data streams. The view cannot provide data for a chart or table.
- ▶ **Add**  
Create a view for an application, which is based on a viewable stream of a currently running job.
- ▶ **Remove**  
Delete the selected view. You must stop a view before you can delete it.
- ▶ **Settings**  
View or update the following settings for the selected view:
  - Basics: Application, operator, port, and view information.
  - Attributes: Tuple attributes from the application that are available for this view.
  - Filter: Keep all data (default) or configure one of the view attributes to be used to filter the incoming tuples. If you choose a filter attribute, the data that is accumulated in the view includes only events for tuples that match the filter.
  - Buffer: The amount of historical data to buffer in the view and the maximum tuple rate.
  - Startup: When to start buffering data.

Complete the following steps to create a view:

1. Click **Add**.
2. Enter a name for the view that satisfies the following requirements:
  - Contains a maximum of 40 characters
  - Can contain alphanumeric characters, spaces, periods, and colons
  - Can contain parentheses. No other special characters are allowed
  - Optional: Enter a view description. This description can contain a maximum of 150 characters.
3. Select the running application job that is associated with the view.
4. Select the output port data stream that is associated with the view.
5. Select one or more attributes for the view: Accept the default values for the following settings or specify a different setting. Keep all data (default), or specify a data filter.

**Specify data filter:** To specify a data filter, select **Keep only data that meets the following specification**. InfoSphere Streams uses the filter attribute and regular expression that you specify to determine which tuples to keep. You can filter on only one attribute at a time.

6. Use the default tuple rate, or specify a different rate.
7. Start buffering data when data is requested from the view (default), or start buffering data when data is received.
8. Click **Finish**.

**Important:** If you attempt to remove a view that is used by an existing chart or table, InfoSphere Streams displays a message that the chart or table is not operational if the view is deleted. You can confirm the deletion or cancel the request. To remove the chart or table that is using the view, select **Remove the associated charts I have permissions for**. If you do not select this option, the chart or table is listed on the Charts page with a View Name of View not found, and you must delete it manually.

For more information, see the Streams Console section about monitoring and managing applications in *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888 or the InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

### 8.3.9 Charts

Use the Charts page to create chart definitions for application data that you want to display in a chart or table format. You also can use this page to view or update the settings for existing charts and tables.

Before you can create a chart or table, you must first create at least one view by using the Views page. After you create one or more views, you can use this page to configure chart definitions for displaying line charts, bar charts, and tables, as shown in Figure 8-29.

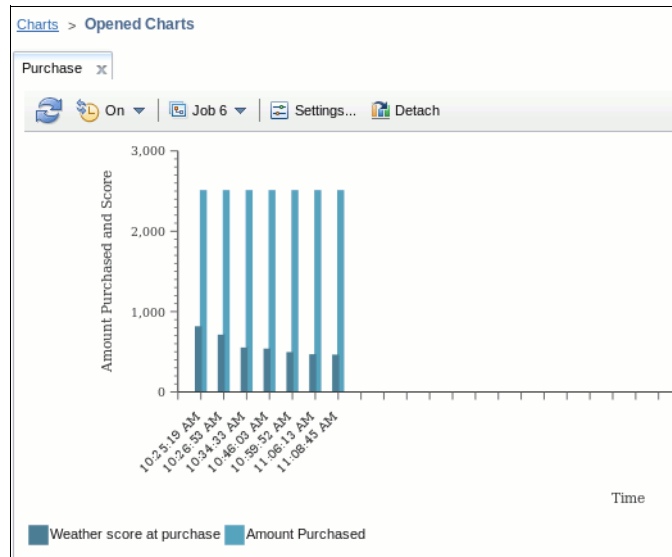


Figure 8-29 Chart monitoring an automated commodity purchase job activity

## Charts controls options

The following chart control options are available:

- ▶ Refresh icon  
Update the contents of this page.
- ▶ Open  
Open the selected chart or table.

- ▶ Add Chart  
Create a line or bar chart.

Complete the following steps to create a chart:

- a. Click **Add Chart**.

Enter a name for the chart that satisfies the following requirements:

- Contains a maximum of 40 characters.
- Can contain alphanumeric characters, spaces, periods, and colons.
- Can contain parentheses. No other special characters are allowed.
- Optional: Enter a chart title, and specify the location of the chart title. The chart title must satisfy the same requirements as the chart name.

- b. Specify the type of chart.

- c. Select one of the Views that was previously created in the Views page.

Optional: Enter a chart description. This description can contain a maximum of 150 characters.

- d. Configure the Y axis of the chart.

**Attribute label:** After you add an attribute, the attribute label is in quotes. This configuration indicates that the attribute name is shown in the chart.

Complete the following steps to show the value of the attribute:

- i. Select the attribute and click **Edit**.
- ii. Select the **Attribute** option, and then select the attribute in the list.
- iii. Click **OK**.

The attribute label is no longer in quotes, which indicates that the value of the attribute is shown in the chart.

- e. Configure the X axis of the chart.

- f. Configure additional chart settings or accept the default settings.

**Chart themes:** The chart themes are standard Dojo charting themes and determine the colors that are used in the chart.

- g. Click **Finish**.

To view the chart, select the chart in the list and click **Open**.

► **Add Table:**

Create a table view. Complete the following steps to create a table:

- a. Click **Add Table**.
- b. Enter a name for the table that satisfies the following requirements:
  - Contains a maximum of 40 characters
  - Can contain alphanumeric characters, spaces, periods, and colons
  - Can contain parentheses. No other special characters are allowed

- c. Select a view.

Optional: Enter a table description. This description can contain a maximum of 150 characters.

- d. Configure the table columns and refresh rate.

- e. Click **Finish**.

To view the table, select the table in the list and **click Open**.

► **Remove**

Remove the selected chart or table.

► **Settings**

Update the settings for the selected chart or table.

For more information, see the Streams Console section about monitoring and managing applications in *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888 or the InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

### 8.3.10 Application Graph

Use the Application Graph page to view a graphical representation of the applications that are running in the InfoSphere Streams instance.

You also can launch the application graph from the Instance Status and Jobs pages.

**Tip:** To display a subset of jobs in the graph, you must launch the graph from the Jobs page. Selecting a subset of jobs can be useful when the instance contains a large number of running jobs, or contains jobs with a large number of PEs and operators.

By default, the application graph displays a maximum of 2000 PEs and PE connections. If the maximum number is exceeded, InfoSphere Streams displays a warning message.

You can increase the maximum number of objects that are displayed in the graph, but do so with caution. If the maximum number is too large, the browser can run out of memory and crash. To increase the maximum number, update the value of the `SWS.graphThreshold` property by using one of the following methods:

- ▶ Update the Application graph object load threshold value on the Streams Console General Settings page.
- ▶ Update the `SWS.graphThreshold` property value by using the `streamtool setproperty` command, for example:

```
streamtool setproperty -i instance1 SWS.graphThreshold=10000
```

You can refresh the graph by clicking the **Refresh** icon or enabling the automatic refreshing option. By default, automatic refreshing is set to Off. Refreshing the graph can affect performance.

You can use the application graph to visualize the overall status of running applications and view information about the application jobs, PEs, operators, ports, and connections. This information can help you to identify runtime and performance problems.

After you identify which application component is causing a problem, you can view or download the component logs to obtain more information about the problem. To select logs to view or download, right-click the component in the graph.

The application graph displays the following components:

- ▶ Jobs are displayed as rectangles with rounded edges. The Job ID appears in the center and a plus sign (+) appears in the upper right corner of the job node, as shown in Figure 8-30.

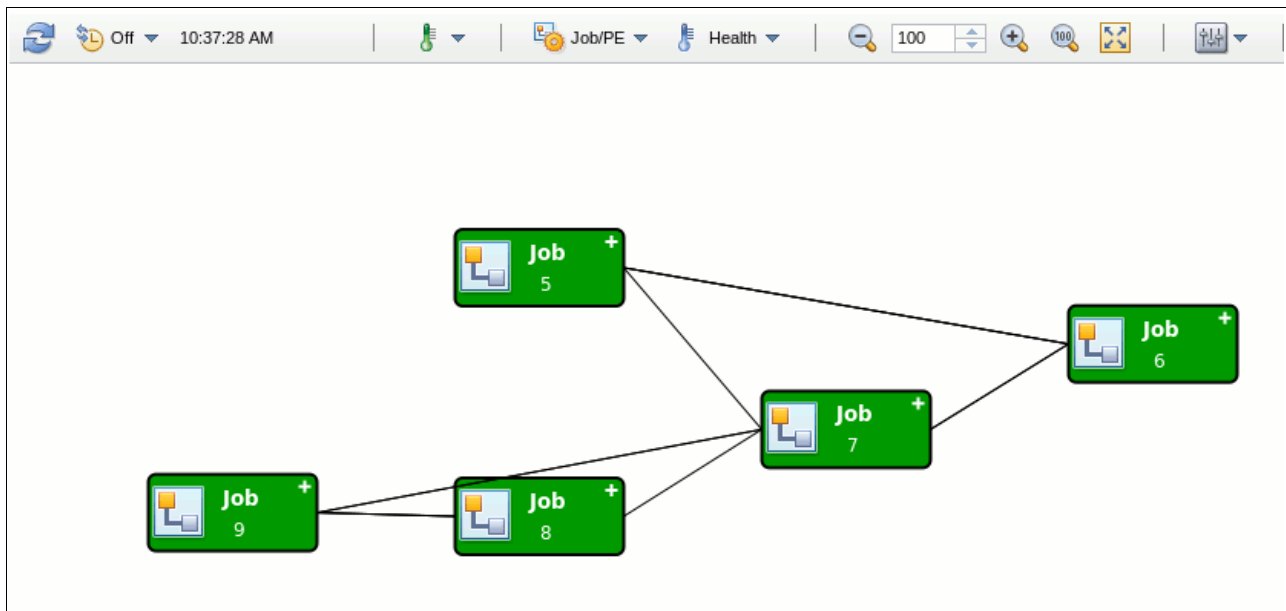


Figure 8-30 Application graphic in an initial state (a green box represents a job)

- ▶ PEs are displayed as rectangles without rounded edges. A black line runs horizontally across the PE node, the PE ID appears above this line, and a plus sign (+) appears in the upper right corner of the PE node.

- ▶ Operators are displayed as rectangles with rounded edges. The operator type and name appear in the center of the operator node. There is no plus sign (+) on operator nodes, as shown in Figure 8-31.

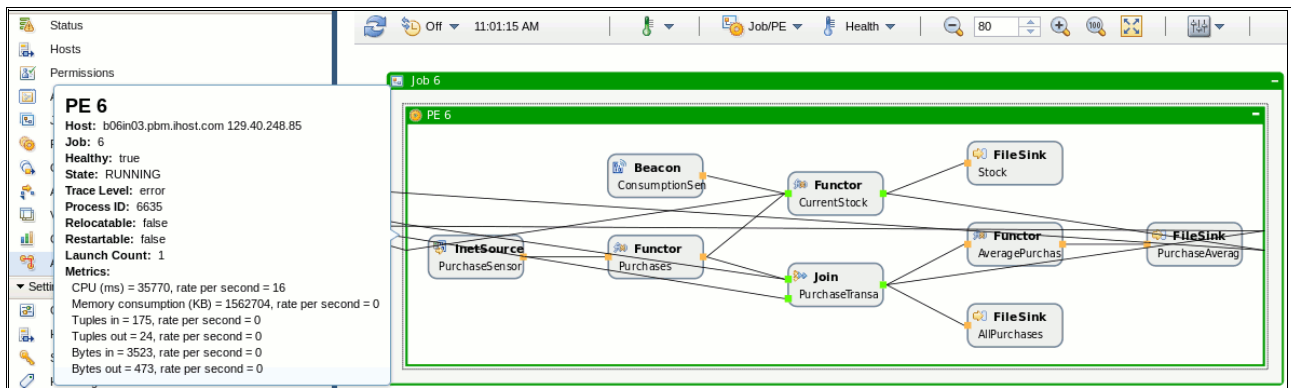


Figure 8-31 Graph in a detailed level of a job

- ▶ Ports are displayed as small squares on the left and right edges of PE and operator nodes. The port color indicates the following connection conditions:
  - Orange-colored ports indicate connections to other nodes inside the same job.
  - Green-colored ports indicate connections to nodes in different jobs.

Connections between nodes are displayed as black lines between nodes inside the same job, or as green lines between nodes in different jobs.

To display information about an application component, mouse over the component. This information might include component identification information, component host placement information, component status, and component metrics.

To expand the application component to show associated components, click the plus sign (+). A minus sign (-) indicates that the component is expanded. For example, clicking the plus sign (+) on a job node shows which PEs are associated with that job. If you click the minus sign (-) that replaces the plus sign, the job node returns to its original state and the associated PEs are no longer displayed.

For more information about how application components are displayed in a graph, see Graph layout views and Stream attributes views in the IBM Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>



## 8.4 Settings

The Console settings provide a graphical interface with which you can set up several settings for a Streams application environment.

### 8.4.1 General

The General Settings page is used to configure the recovery mode and host load protection of the InfoSphere Streams instance. Changes to these settings are not applied until the InfoSphere Streams instance is restarted.

The recovery mode indicates what mode should be used by the InfoSphere Streams runtime components for fault tolerance and recoverability.

The host load protection keeps InfoSphere Streams jobs from over-burdening the hosts within the instance, as shown in Figure 8-32.

**General Settings**

Recovery mode:  On  Off

Host load protection:  Enable host load protection  
Host load threshold:

Application graph object threshold:  
Application graph object load threshold:

**Name service:** FS:/home/streamsadmin/.streams/instances/streams@streamsadmin/namespace

Figure 8-32 General Settings options

The following selections can be made on the General Settings page:

► Recovery mode

If a recovery mode of On is specified, the system enables relocation and restarting of failed stateful runtime services. The system attempts to restart any incomplete jobs that were running when the service went down. The history of all completed jobs is retained. Persisting this information requires an external database to be set up.

A recovery mode of Off indicates that the system should not attempt to recover failed services. Instead, if one of the stateful services becomes unavailable, you must shut down and restart the instance to recover. The job state is not retained for completed or incomplete jobs. This mode eliminates the database setup requirement and might be desirable for application developers who are using a single user installation.

► Host load protection

Enabling host load protection prevents new InfoSphere Streams jobs or PEs from starting on hosts whose current load is above the threshold. If a host within the InfoSphere Streams instance is running at or above the specified host load threshold, no new jobs are started on that host. If all hosts within the instance are at or above the threshold, no new jobs can be started in the instance until at least one host's activity level falls below the threshold. New jobs can be forced to start by specifying the **--override** option on the **streamtool submitjob** command.

► Host load threshold

The host load threshold where new jobs are not started when a host's load is above this value. The value must be between 1 - 999,999. The host load threshold value also controls the PE performance recommendations that are found in the Performance section of the InfoSphere Streams Console interface.

► Application graph object load threshold

The application graph object load threshold enables a warning message within the application graph when the number of PEs and PE links exceeds the value. The value must be between 100 - 50,000. The setting is provided to allow intervention before large graph is loaded that might take a long time, hang the browser window, or even crash the browser window.

► Name Service

The name service value is a display only field that shows the location and type of the name service repository. The name service repository can be file system-based (signified by a name that starts with FS:) or distributed name server-based (signified by a name that starts with DN:).

## General control options

The following general control options are available:

► Apply

Apply all settings updates for this page and any updates that are made on other InfoSphere Streams settings pages to the InfoSphere Streams instance configuration.

► Reset

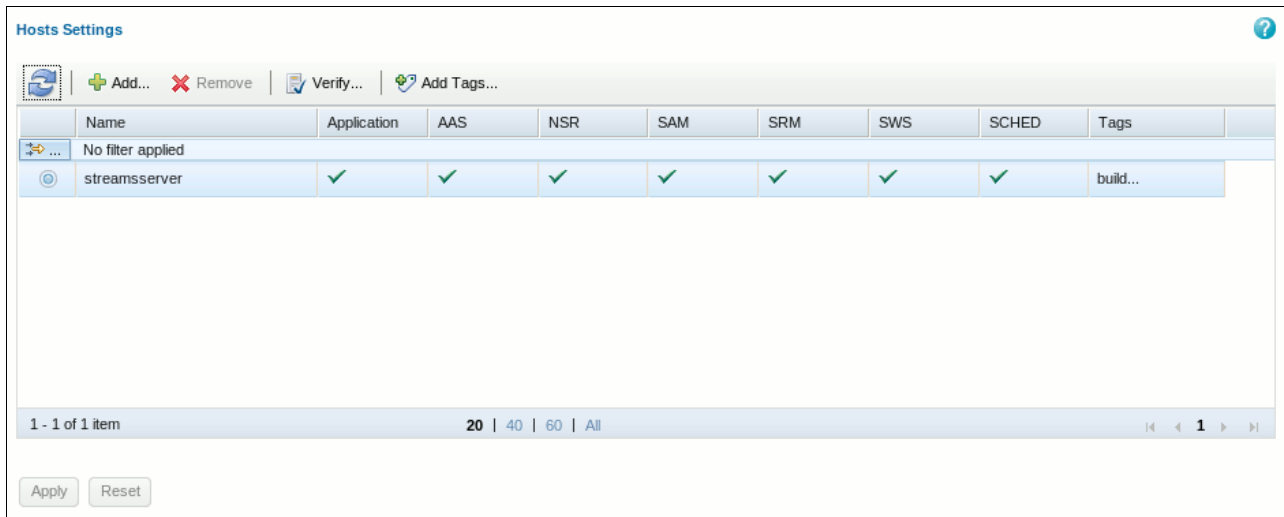
Reset all settings for this page and all other InfoSphere Streams settings pages back to the settings that were active when this set of configuration updates was started.

► Refresh

Reloads the settings for this page (and all other settings pages) from the instance configuration.

## 8.4.2 Hosts Settings

The Hosts settings page is used to display the list of hosts that are providing services for this InfoSphere Streams instance. Other hosts can be added to the InfoSphere Streams instance from this page. Any other host that is added hosts the application service by default, as shown in Figure 8-33.



Name	Application	AAS	NSR	SAM	SRM	SWS	SCHED	Tags
No filter applied								
streamsserver	✓	✓	✓	✓	✓	✓	✓	build...

Figure 8-33 Hosts Settings list

The following InfoSphere Streams services can be configured to run on a host:

- ▶ Application
- ▶ Authentication and Authorization Service (AAS)
- ▶ Streams Application Manager (SAM)
- ▶ Job Scheduler (SCHED)
- ▶ Streams Resource Manager (SRM)
- ▶ Stream Web Server (SWS)

### Hosts control options

The following host control options are available:

- ▶ Add Host

Displays a window that allows a host to be added to this InfoSphere Streams instance. The host to be added can be specified as a fully qualified host name or the host's static IP address.

- ▶ Add Hosts

Displays a window that allows a group of hosts (contained in a file) to be added to this InfoSphere Streams instance. The hosts that are listed in the file can be specified by their fully qualified host name or by their static IP address. The file format can be a list that is separated by a comma, tab, or line.

- ▶ **Verify**

Displays a window that shows if the selected host meets the software requirements, hardware requirements, and configuration to host InfoSphere Streams services within this Streams instance. An overall status is shown, and the valid states are Passed, Warning, and Failed. If the overall status is Warning or Failed, look through the other pages of detailed dependency information to determine what must be corrected before this host is used actively in an InfoSphere Streams instance. For more information, see “Hosts control options” on page 235.
- ▶ **Remove**

Removes the selected host from the InfoSphere Streams instance. Any active jobs that are running on the selected host are stopped before the host is removed from the instance. Hosts that are running services other than the application service cannot be removed from the instance. If an administrative or management host must be removed from the instance, use the streamtool command-line interface to move the management services to a new node, and then perform the remove host operation.
- ▶ **Add Tags**

Displays a window that allows tags to be added to the selected host. The tags that are associated with a host can control which application workloads are run on a particular host.
- ▶ **Refresh**

Retrieves an updated list of hosts. The list of hosts that is displayed in the web browser might become out of sync if other administrators are updating the instance after initial retrieval of the list of hosts for this InfoSphere Streams instance.
- ▶ **Apply:**

Applies all settings updates for this page and any updates that are made on other InfoSphere Streams settings pages to the InfoSphere Streams instance configuration.
- ▶ **Reset:**

Resets all settings for this page and all other InfoSphere Streams settings pages back to the settings that were active when this set of configuration updates was started.

**Reference host:** The host where the InfoSphere Streams Console and SWS services are running is used as the reference host for all verification checking for new hosts within the instance. The reference host is used for tasks such as determining whether the new host's operating system architecture is consistent with the instance.

## Host verification information

This panel shows the overall verification status of the selected host.

The architecture of the selected host is displayed as the Actual architecture, and the architecture of the instance's reference host is shown as the Expected architecture, as shown in Figure 8-34.

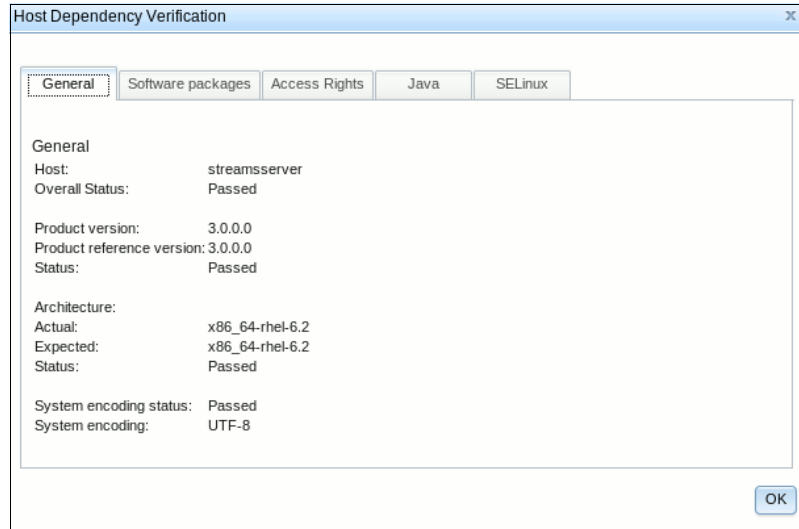


Figure 8-34 Host verification page with tabs for gathering more host status

A warning status is shown if the architecture of the selected host does not match the architecture of the reference host.

In addition to the General tab that is shown, the following tabs are available in the Host Dependency Verification window:

- ▶ Software packages

This tab shows the RPM packages that are required by InfoSphere Streams. Streams requires specific versions of many of these packages. The individual status for each package is displayed. An overall status for the package list is also displayed on this page.

- ▶ Access Rights

This tab shows a list of access rights that must be met for Streams to run correctly.

- ▶ Java

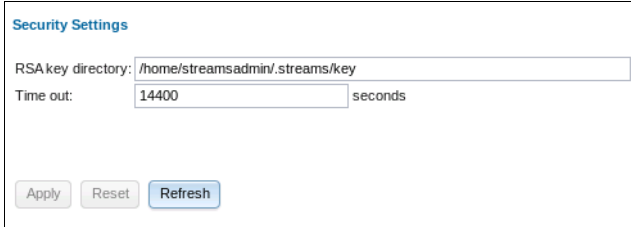
This tab shows the version of Java on the selected host, and the version of Java that is required by InfoSphere Streams.

- ▶ SELinux

This tab shows the current setting of SELinux on the selected host. It also shows the current SELinux policy setting of the Streams instance.

## 8.4.3 Security

Use the Security settings page to configure information that is related to the security credentials that are used within the InfoSphere Streams instance, as shown in Figure 8-35.



Security Settings

RSA key directory:

Time out:  seconds

Figure 8-35 Security Settings controls

The following options are available:

- ▶ **Public Key Directory**  
The directory where the public keys are stored.
- ▶ **Time Out**  
The amount of time a security token remains available for use. When the token expires, the user must reauthenticate to InfoSphere Streams.
- ▶ **Apply**  
Applies all settings updates for this page, and any updates that are made on other InfoSphere Streams settings pages to the InfoSphere Streams instance configuration.
- ▶ **Reset**  
Resets all settings for this page and all other InfoSphere Streams settings pages back to the settings that were active when this set of configuration updates was started.
- ▶ **Refresh**  
Reloads the settings for this page and all other settings pages from the instance configuration.

## 8.4.4 Host tags

The Host Tags page is used to work with the list of host tags for this Streams instance. The host tags from this list can be used to identify that a host system includes the necessary hardware or configuration to host specific types of workloads that are contained in Streams applications, as shown in Figure 8-36.

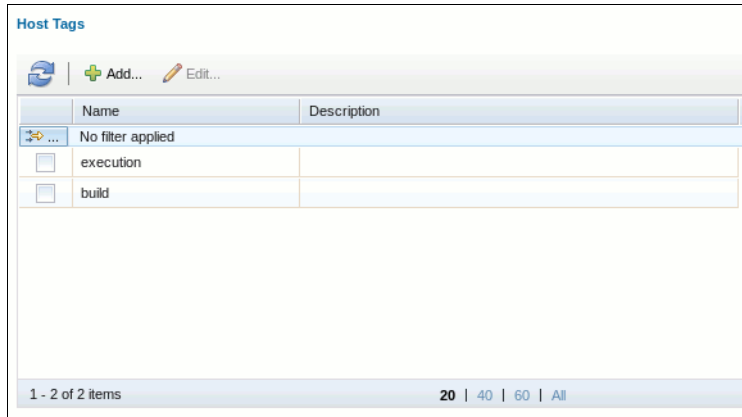


Figure 8-36 Host Tags controls

Host tags can be added to the instance's host tag list. A description for the host tag is optional. Existing tags' descriptions also can be modified from this page. Instance host tags cannot be removed while a Streams instance is running. If the need arises where a tag should no longer appear in the tag list for the instance, shut down the Streams instance and use the Streamtool command-line interface to remove the tags from the instance.

## 8.4.5 Web Server

The Web Server settings page is used to configure the security settings and port for SWS. The SWS is the service that provides the InfoSphere Streams Administration Console web interface. It also provides the web services that are used by the Administration console to control various aspects of the InfoSphere Streams instance and its configuration, as shown in Figure 8-37.

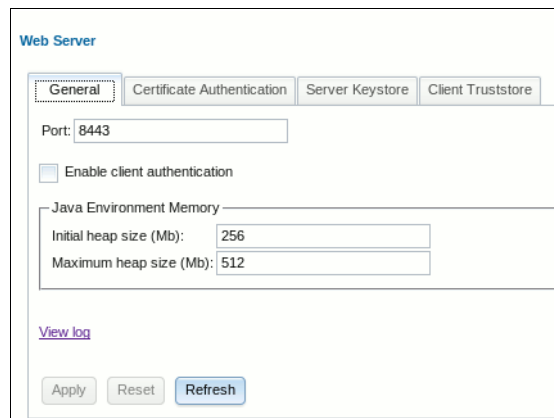


Figure 8-37 Web Server settings page

The following tabs are available:

► General

The following options are available on the General tab:

- Port  
The secure HTTPS port that is used by the Streams Web Server or InfoSphere Streams Administration Console.
- Enable Client Authentication  
Select this option to enable client authentication. For more information about client authentication, see “Client Authentication” on page 243.
- Initial heap size  
The initial size of the heap that is allocated to the Streams web application server. The size that is specified is in Mb.
- Maximum heap size  
The maximum size of the heap that can be allocated to the Streams web application server. The size that is specified is in Mb, and cannot exceed the amount of memory that is available on the host where the SWS service is running.  
The valid heap size memory range is 64 - 65536 Mb.
- View log  
View the web application server log file contents. Viewing this log might reveal problems that are occurring within the web server that hosts the SWS service.
- Apply  
Applies all settings updates for this page and any updates that are made on other InfoSphere Streams settings pages to the InfoSphere Streams instance configuration.
- Reset  
Resets all settings for this page and all other InfoSphere Streams settings pages back to the settings that were active when this set of configuration updates was started.
- Refresh  
Reloads the settings for this page, and all other settings pages, from the instance configuration.

► Certificate Authentication

Certificates can be used to represent users and allow the user the ability to log in to the Streams Console without the use of a user and password pair. Certificate authentication uses Client Authentication to authenticate the client connection and extracts the user information from the certificate's distinguished name (DN).

The following options are available:

- User ID pattern  
The pattern that is used to extract the Streams user ID from the certificate's DN.
- Server Keystore:  
Server keystore is used to store the certificates information that is used to identify the server SWS is on. For more information about the actions to take to create your own signed certificate, see “Securing SWS Connection” on page 242.
- New:  
Create a new self-signed certificate. A prompt for certificate information is displayed before the certificate is created.



- Generate Signing Request
 

Generate a new signing request that is based on the self-signed certificate. A file is downloaded. Save this file and send it to a signing authority.
- Renew
 

Renew the certificate in the keystore.

If the certificate is a self-signed certificate, a new self-signed certificate is generated, which replaces the old certificate.

If the certificate is a signed certificate, a new signing request that is based on the certificate is generated and downloaded. Save this file and send it to a signing authority.
- Import Chain
 

Import root/chain certificates. If a certificate signer returns one or more root/chain certificates, use this option to import them into the keystore.
- Import
 

Import a signed certificate. If root/chain certificates are returned with this certificate, they should be imported before the signed certificate is imported.
- Delete Pending Request
 

Delete the pending renew signed certificate information from the keystore.
- Set Password
 

Change the keystore password. The keystore is shipped with a default password.

For more information about the default password, see the InfoSphere Streams documentation.
- ▶ Client Truststore
 

Client truststore is used to store the certificate information that is used to identify the clients that are allowed to connect to SWS if client authentication is enabled.

For more information about the actions to take to enable client authentication, see “Securing SWS Connection” on page 242.

The following options are available:

  - Add
 

Add a client certificate to the truststore. A prompt for an ID that identifies the client and the certificate file is shown.
  - Remove
 

Remove a client certificate from the truststore. Select the client entry and click **Remove** to remove the entry.
  - Update
 

Update the certificate for a client entry. Select the client entry and click **Update**. You are prompted for the new certificate file.
  - Set Truststore Password
 

Change the truststore password. The truststore is shipped with a default password.

For more information about the default password, see the InfoSphere Streams documentation.

## Securing SWS Connection

For security reasons, SWS allows only HTTPS connections. InfoSphere Streams ships a default self-signed certificate with SWS. This certificate is used to identify the SWS server so HTTPS works without the need to configure certificates. However, browsers that are connecting to SWS servers by using the shipped certificate can result in the following warnings:

- ▶ The certificate is self-signed, not signed by a trusted authority.
- ▶ The certificate is not valid because it is created for `www.ibm.com` and not the server SWS is running on.

To establish the HTTPS connection, you must add the certificate exception and trust the self-signed certificate from the browser. The browser should be able to connect to the SWS server after the self-signed certificate is trusted.

To better secure the connection, you should create your own certificate and have it signed by a trusted authority. For more information about creating your own signed certificate, see “Renewing a self-signed certificate” on page 243.

To provide more security, client authentication can be used to restrict SWS connections to a list of clients. For more information about how to configure and enable client authentication, see “Server Authentication” on page 242.

## Server Authentication

The SWS server uses the server certificate in the server keystore to identify itself to the clients. A self-signed certificate is shipped with SWS and is used by default. You can create your own self-signed or signed certificate by using the processes that are described next.

### *Creating a self-signed certificate*

You can create your own self-signed certificate. The certificate is created for the host that SWS is running on. Creating your own self-signed certificate can reduce the number of warning messages the browser produces.

Complete the following steps to create a self-signed certificate:

1. Click **New**.
2. Enter the information for the certificate.
3. Click **Create** to create a self-signed certificate.

**Warning:** The new certificate replaces the current certificate in the server keystore, even if it is a signed certificate.

### *Creating a signed certificate*

Complete the following steps to create and use a signed certificate:

1. Click **New** to create a self-signed certificate. You are prompted for information for the new certificate.
2. Click **Generate New Signing Request** to generate a certificate signing request. Save the certificate signing request file.
3. Send the certificate signing request to a certificate signing authority.
4. Click **Import Chain** to import the root/chain certificates if the signer returns root/chain certificates.
5. Click **Import** to import the signed certificate.
6. Restart the Streams instance.

## Renewing a self-signed certificate

To renew a self-signed certificate, click **Renew**. If the certificate that is in use is a self-signed certificate, a new certificate is generated that is based on the current certificate. The Streams instance must be restarted for the change to take effect.

Complete the following steps to renew a signed certificate:

1. Click **Renew** to generate a certificate signing request.
2. Send the certificate signing request to a certificate signing authority.
3. Click **Import Chain** to import the root/chain certificates if the signer returns root/chain certificates.
4. Click **Import** to import the signed certificate.
5. Restart the Streams instance.

## Client Authentication

Client authentication is used to allow only authorized clients to connect to SWS. The client browser has the present a certificate that identifies itself, and SWS must have the corresponding certificate information before it allows the connection.

Complete the following steps to configure Client Authentication:

1. Import the certificate that identifies the client by clicking **Add**. Use a unique name for the client ID to identify the client.
2. Import the client certificate into the client browser.
3. Repeat steps 1 and 2 for each client that is allowed to connect.
4. From the General tab, select **Enable client authentication**.
5. Restart the Streams instance.

Complete the following steps to update the Client Certificate:

1. Click **Update** from the Client Truststore tab.
2. Select the new certificate to use for the selected client.

The client certificate is replaced with the new certificate. The Streams instance must be restarted for the change to take effect.

The following settings can be updated:

► **User ID Pattern**

User ID pattern is a pattern of reserved keywords and regular expressions that specifies how to extract the Streams user ID from the client certificate's distinguished name (DN).

► **Distinguished Name**

The DN in a certificate consists of a list of name-value pairs, as shown in Example 8-8.

*Example 8-8 A DN name/pairs record in a list*

---

```
EMAILADDRESS=bob@mycompany.com, cn=Bob, OU=mydept, O=myorg, L=Rochester, ST=MN, C=US
```

---

The name of each name-value pair can be used within the user ID pattern. For example, specifying the pattern of `${cn}` extracts the value of the `cn` pair, which results in the user ID being Bob.

► Replacing String

Values that are extracted from the DN can be changed by replacing a string with other strings. Non-alphanumeric characters can be used as delimiters to specify the substitution strings. For example, `#{cn,o,e}` results in “Bob”. The search string can be a regular expression.

For example, the use of data from the Location field (L), `#{L/[o-z]/_}` substitutes all occurrences of characters from o-z with `_`, resulting in `R_che__e_`.

► Adding String

Strings that are contained within `{}` are added to the resulting string. For example, `#{cn}@{C}` results in `Bob@US`.

## 8.4.6 Logging and Tracing

Use the Logging and Tracing Settings pages to update the log and trace properties for the InfoSphere Streams product and its components, as shown in the Figure 8-38.

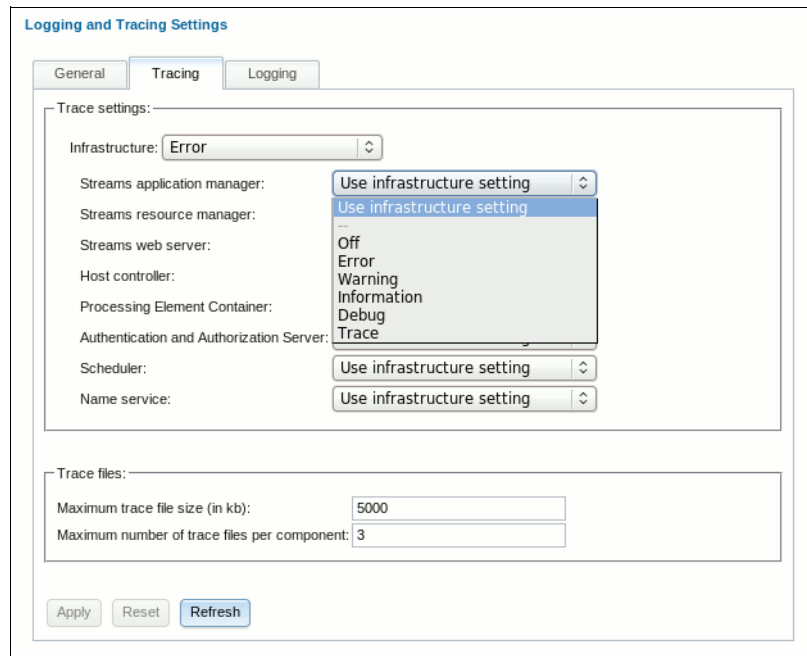


Figure 8-38 Logging and Tracing Settings panel

### Logging and Tracing controls options

To modify log and trace settings, you must be the instance owner. Modifying trace settings can affect the performance of InfoSphere Streams. In most cases, it is advisable to change to trace settings that are based on guidance from IBM Software Support.

Updates to the following settings do not require a restart of the instance to take effect:

- Infrastructure trace level
- Runtime service trace levels
- Logging level

For all other settings, you must restart the instance for the change to take effect.

**Important:** When you click **Apply**, InfoSphere Streams saves and applies all settings updates for this page, and all other settings pages, to the instance configuration.

When you click **Reset**, InfoSphere Streams restores the settings on this page, and all other settings pages, to the settings that were active before any unapplied updates were saved.

When you click **Refresh**, InfoSphere Streams reloads the settings for this page, and all other settings pages, from the instance configuration.

The owner can update the following log and trace files settings:

► Log and Trace file location

The default root directory where InfoSphere Streams stores log and trace files is the /tmp directory. You can change the root directory only. InfoSphere Streams creates the names of the subdirectories under the root directory. You cannot change the names of the subdirectories.

The following tracing settings are available:

– Infrastructure trace level

The amount of trace data that is generated by InfoSphere Streams.

– Runtime service trace levels

By default, the Infrastructure trace level is used for all runtime services. If you specify another trace level for a runtime service, the runtime service level overrides the Infrastructure level.

The following options are settings of Trace:

- Off
- Error
- Warning
- Information
- Debug
- Trace

► Maximum trace file size

The maximum size in kilobytes of the trace files for each InfoSphere Streams component that traces data. The default value is 5000.

► Maximum number of trace files per component

The maximum number of trace files for each InfoSphere Streams component that traces data. The default value is 3.

## Logging settings

The following options are available for the Logging setting page:

- ▶ Log type

If you use the default log type of File, InfoSphere Streams logs events and messages in rolling log files on each host. To configure InfoSphere Streams to log events and messages in the Linux system log, select System Log.
- ▶ Logging level

The logging severity level. The following options are in the order of severity:

  - Error (Highest severity)
  - Warning (Default setting)
  - Information (Lowest severity)
- ▶ Maximum log file size

Valid if the log type is File. The maximum size in kilobytes of the rolling log files for each InfoSphere Streams component that logs data. The default value is 5000.
- ▶ Maximum number of log files per component

Valid if the log type is File. The maximum number of rolling log files for each InfoSphere Streams component that logs data. The default value is 3.

**For more information:** For more information, see the section about managing log and trace data by using the Streams Console in the *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888 or the InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

## 8.5 Streams Recovery Database

The InfoSphere Streams recovery database is a DB2 database that records the state of instance services. When instance services fail, InfoSphere Streams restarts them and restores their state by retrieving the appropriate state data from the recovery database. The Streams recovery database records information, which enables the Streams instance administrator to manually restart the following management services, move these management services to a known state, or both:

- ▶ Authentication and Authorization Service (AAS)
- ▶ Streams Application Manager (SAM)
- ▶ Streams Resource Manager (SRM)

For example, if a host is running the SAM service and the host crashes, the Streams instance administrator can restart the SAM service on another cluster host and Streams restores its state from the recovery database. The information that is stored in the recovery database includes the following states for an instance and its applications:

- ▶ Instance resource state (instance state, instance daemon placement, and state)
- ▶ Job state
- ▶ PE placement, stream connection state
- ▶ Authentication and authorization state

A steady state is when all jobs are submitted and running and there are no changes to the state of the instance. The recovery database is not active when an instance is in the steady state. In addition, the recovery database is unaffected by the activity of Streams applications.

By default, the configuration of the recovery database is shared among the Streams instances that are created by the same Streams installation. An administrator can isolate the recovery data of Streams instances by creating a separate Streams installation for each instance. For more information about using a separate installation for each instance, see *Multiple installations of InfoSphere Streams on the same host in the product Installation and Administration guide*.

If your InfoSphere Streams instance is configured with the `RecoveryMode` property set to `On` and you configured a recovery database on one of your cluster hosts, individual InfoSphere Streams management components can be recovered if they fail. If your instance is not configured with the `RecoveryMode` and one of the instance's management services or hosts fails, you must restart the instance to restore it to a fully operational state.

Management service failures can be recovered only if you set up an InfoSphere Streams recovery database on one of your cluster hosts, the recovery mode is enabled, and a file system-based name service is used.

Failures in application PEs and hosts can be recovered without a recovery database.

**Important:** By default, the `NameServiceUrl` property for a Streams instance is set to `DN:`, and a distributed name service is used. A distributed name service does not support a Streams recovery database. Use a file system-based name service when recovery of management services is critical. To use a file system-based name service, set the `NameServiceUrl` property to `FS:`. For more information about this property, enter `streamtool man properties`.

### DB2 recovery database requirements

InfoSphere Streams supports DB2 for Linux, UNIX, and Windows Versions 9.7 and 10.1.

## Setting up the recovery database

The First Steps application configures the DB2 database for InfoSphere Streams, which includes customizing the database for the product, and creating or updating the `dbglobalconfig.conf` database configuration file in the `installation-owner-home-directory/.streams/config` directory.

The First Steps application also tests the connection to the InfoSphere Streams recovery database. You must be the InfoSphere Streams installation owner to change the database information as shown in Figure 8-39.



**Configure the recovery database**

Configure an existing database to be used for InfoSphere Streams recovery.

This connection information will be stored in the runtime database configuration file.

Database

Name: streams

Host: localhost

Port: 50002

User ID: streams

Password: .....

Cancel OK

Figure 8-39 Configuration parameters for setting up a Recovery database

Figure 8-40 Shows the database ID and Password prompt window, which is required during the Recovery database setup process.



**Administrator ID and password**

This configuration is needed to customize the database for the product.

This configuration is not stored.

Administrator ID:

Password:

Cancel OK

Figure 8-40 Database ID and Password prompt window

**For more information:** For more information about the Recovery database, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888 or the InfoSphere Streams Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>





# A

## Installing InfoSphere Streams

In this appendix, we provide detailed information about the installation and configuration of the IBM InfoSphere Streams (Streams) product. These step-by-step instructions guide you through the process of creating a single- or a multi-host Streams environment. By completing these steps, you can run the SPL sample application that is shipped with the Streams installation media and the examples that are included in this book.

## Hardware requirements for InfoSphere Streams

Before you install IBM InfoSphere Streams, ensure that the target system satisfies the minimum hardware requirements that are shown in Table A-1.

Table A-1 Hardware requirements

Component	Minimum requirements	Comments
System	x86_64 (64-bit)	InfoSphere Streams supports Red Hat Enterprise Linux (RHEL) and the Community Enterprise Operating System (CentOS) on x86_64 systems.
	BM POWER7 (64-bit)	IBM POWER7 systems support the RHEL operating system.
Display	1280 x 1024	Lower resolutions are supported but not recommended for Streams Studio.
Memory	2 GB	<p>The amount of memory that is required by InfoSphere Streams is dependent on the applications that are developed and deployed on the InfoSphere Streams hosts.</p> <p>The Commodity Purchasing sample application, and many of the other available samples that are provided with InfoSphere Streams, can be compiled and run with 2 GB of memory.</p>
Disk space	4 GB	<p>Includes disk space that is required for installation and development resources.</p> <p>The locally installed InfoSphere Streams Information Center requires approximately 100 MB of disk space.</p>

# Software requirements for InfoSphere Streams

Before IBM InfoSphere Streams is installed, ensure that the target system satisfies the software requirements that are described in this section.

## Operating system requirements for InfoSphere Streams

InfoSphere Streams supports RHEL and CentOS.

**Important:** If you are running InfoSphere Streams across multiple hosts, the hosts can run in a mixed operating system environment (RHEL or CentOS). However, all InfoSphere Streams hosts must run on the same hardware system (x86\_64 or IBM POWER7), system architecture (64-bit), and major operating system version (Version 5 or 6). For example, if one host is running on a 64-bit x86\_64 system with RHEL 5, other hosts must run on a 64-bit x86\_64 system with RHEL 5 or CentOS 5.

## Supported operating system versions for InfoSphere Streams

All InfoSphere Streams hosts must run a supported version of RHEL or CentOS.

Table A-2 lists the supported operating system versions on x86\_64 and IBM POWER7 systems.

*Table A-2 Supported operating system versions for InfoSphere Streams*

System hardware and architecture	Supported versions	Supported versions
<b>RHEL</b>	<b>RHEL 5</b>	<b>RHEL 6</b>
x86_64 (64-bit)	Version 5.3, or later	Version 6.1, or later
IBM POWER7 (64-bit)		Version 6.1, or later
<b>CentOS</b>	<b>CentOS 5</b>	<b>CentOS 6</b>
x86_64 (64-bit)	Version 5.3, or later	Version 6.1, or later

## Operating system migration considerations and restrictions

Before you upgrade RHEL or CentOS from Version 5 to Version 6, review the InfoSphere Streams considerations and restrictions that are described in this section.

The following considerations are particularly important:

- ▶ Ensure that you download the correct InfoSphere Streams installation package for your target system. The operating system and architecture of the target system and the product installation package must be the same. For example, if your target system is a 64-bit RHEL 5 system, you must download the InfoSphere Streams 64-bit RHEL 5 installation package.
- ▶ Upgrade your operating system from Version 5 to Version 6 before installing the Version 6 package for InfoSphere Streams. For more information about Streams migration, see “Upgrade options and requirements for Streams” on the Information Center at this website:  
<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

## Required Java version for InfoSphere Streams

InfoSphere Streams requires the IBM Java SE Version 6 SDK, which includes the following conditions:

- ▶ On x86\_64 systems, you can use the Java SE 6 JDK from Oracle as your Java development environment for Streams Studio and SPL applications. The IBM Java SE Version 6 SDK and Java SE 6 JDK from Oracle were tested with Streams Studio and SPL applications with operators that were implemented by using the InfoSphere Streams Java Operator API.
- ▶ The Java SE 6 JDK from Oracle is not supported on IBM POWER7 systems.

The IBM Java SE Version 6 SDK is included with the InfoSphere Streams product in the `product-installation-package-directory/StreamsInstallFiles/rpm` directory.

Before Streams is installed, run the dependency checker script to verify the requirements on the hosts on which you plan to install InfoSphere Streams. If you are installing the product on a cluster of hosts, run the dependency checker script on each host in the cluster.

## Dependency checker script for InfoSphere Streams

The dependency checker script identifies any InfoSphere Streams incompatibilities that are related to your operating system and software settings. This script also identifies any software dependencies that are required to run the product. In a multiple-host environment, you must run this script on all hosts and correct any incompatibilities before you use InfoSphere Streams.

Complete the following steps to run the dependency checker script before you install InfoSphere Streams:

1. Uncompress the product installation package and enter the following command:

```
tar -zxvf product-installation-package-name.tar.gz
```

The **tar** command creates the `StreamsInstallFiles` directory, which contains the self-extracting binary file (`InfoSphereStreamsSetup.bin`) and other installation files.

2. Change to the InfoSphere Streams installation files directory to the following directory:

```
cd product-installation-package-directory/StreamsInstallFiles/
```

3. Enter the following command to run the dependency checker script:

```
./dependency_checker.sh
```

Enter the following commands to run the dependency checker script after you install InfoSphere Streams:

```
cd product-installation-directory/bin
./dependency_checker.sh
```

The following command option is available:

**-v level**

This verbose option displays other messages. The following values for the *level* option are available:

- **1**: Includes other informational messages
- **2**: Includes other debug messages
- **-h**: Displays the help information

Fix any errors or warnings that are issued by the `dependency_checker` script. Continue to run and fix until all errors and warnings are fixed.

After the installation is complete, use the `streamtool checkhost` command to verify the requirements on single or multiple hosts. For more information, see the IBM Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

## Required Red Hat Package Manager for InfoSphere Streams

IBM InfoSphere Streams requires a set of prerequisite Red Hat Package Managers (RPMs).

**Tip:** You can install the RPMs with a package manager such as YUM. YUM resolves and installs other dependencies these RPMs have, which reduces or eliminates a common problem when Streams is installed. You can also use a package such as YUM to install the RPMs that are shipped in the installation bundle.

The following important points should be considered:

- ▶ Install all prerequisite RPMs before you install InfoSphere Streams, either with the installation utility or with a package, such as Yum. The installation utility checks which RPMs you need to install, but the utility does not install prerequisite RPMs. The installation utility detects if prerequisite RPMs are not installed on the current host or if installed RPMs are at an incompatible version level. If required RPMs are missing during the installation, the installation utility gives you the option of continuing the installation or stopping it. After the installation, the list of outstanding dependencies is recorded in the installation summary log.
- ▶ If you start running InfoSphere Streams and the prerequisite RPMs are not installed, the product does not run properly.

## RPM minimum versions and availability

The RPMs that are required for InfoSphere Streams can be grouped into the following categories. The InfoSphere Streams dependency checker script checks for Category 2 and 3 RPMs only, and identifies RPMs that must be installed or updated. This script does not check for Category 1 RPMs because the script assumes that you have installed the minimum required RPMs for your operating system:

### Category 1

RPMs that are included in the following RHEL and CentOS installations are not customized by actions such as removing packages from the base list:

- ▶ Default RHEL graphical mode installation
- ▶ CentOS Basic Server installation

Adding optional packages during the RHEL or CentOS installation does not affect InfoSphere Streams.

### Category 2

RPMs in this category are included in the operating system installation package but are not installed during a default RHEL graphical mode installation or CentOS Basic Server installation.

### Category 3

RPMs in this category are included in the InfoSphere Streams installation package.

**Important:** The information in Table A-3, Table A-4 , and Table A-5 , is subject to change and is provided as an example only. For the latest information, see the IBM Information Center at the following website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

Table A-3 Required RPMs and minimum versions for RHEL 5 and CentOS 5 on x86\_64

RPM name	Minimum RPM version	Operating system release	Availability
curl-devel	7.15.5-2.el5	5.3	Operating system package
	7.15.5-2.1.el5	5.4	
	7.15.5-9.el5	5.5, 5.6	
	7.15.5-9.el5_6.3	5.7	
	7.15.5-15.el5	5.8	
gcc-c++ <sup>a</sup>	4.1.2-44.el5	5.3	Operating system package
	4.1.2-46.el5	5.4	
	4.1.2-48.el5	5.5	
	4.1.2-50.el5	5.6	
	4.1.2-51.el5	5.7	
	4.1.2-52.el5	5.8	
ibm-java-x86_64-sdk	SR 11	5.3, or later	InfoSphere Streams package
perl-XML-Simple	2.14-4.fc6	5.3, or later	Operating system package
selinux-policy-devel <sup>b</sup>	2.4.6-255.el5	5.3	
selinux-policy-targeted <sup>b</sup>	In default operating system installation	5.4, or later	

a. The maximum version of the gcc-c++ RPM must be earlier than Version 4.2. Version 4.2, or later cannot be used with InfoSphere Streams.

b. If SELinux is running in enforcing or permissive mode, the selinux-policy-devel and selinux-policy-targeted RPMs are needed for installation. For version 5.3, the required SELinux RPM versions are available on your operating system update site. The preferred method for installing these RPMs is by using the **yum update** command. Before this command is used, you must set up a yum repository for your operating system update site. The required RPM versions are in the default operating system installation for version 5.4, or later.

Table A-4 Required RPMs and minimum versions for RHEL 6 and CentOS 6 on x86\_64

RPM name	Minimum RPM version	Operating system release	Availability
gcc-c++ <sup>a</sup>	4.4.5-6.el6	6.1	Operating system package
	4.4.6-3.el6	6.2	
ibm-java-x86_64-sdk	SR 11	6.1, 6.2	InfoSphere Streams package
libcurl-devel	7.19.7-26.el6	6.1	Operating system package
	7.19.7-26.el6_1.2	6.2	
perl-XML-Simple	2.18-6.el6	6.1, 6.2	InfoSphere Streams package
policycoreutils-python <sup>b</sup>	2.0.83-19.8.el6_0	6.1	Operating system package
	2.0.83-19.18.el6	6.2	
selinux-policy <sup>c</sup>	3.7.19-126.el6_2.10	6.2	Operating system package
selinux-policy-targeted <sup>c</sup>			

- a. The maximum version of the gcc-c++ RPM must be earlier than Version 4.5. Version 4.5, or later cannot be used with InfoSphere Streams.
- b. If SELinux is running in enforcing or permissive mode, the policycoreutils-python RPM is required. For more information about the use of SELinux with InfoSphere Streams, see this website: <http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>.
- c. The selinux-policy and selinux-policy-targeted RPMs are installed by default. However, if SELinux is running in enforcing or permissive mode, you must update these RPMs to Version 3.7.19-126.el6\_2.10, or later. Updates are available on your operating system update site. The preferred method for installing these RPMs is by using the **yum update** command. Before this command is used, you must set up a yum repository for your operating system update site.



Table A-5 Required RPMs and minimum versions for RHEL 6 on IBM POWER7 systems

RPM name	Minimum RPM version	Operating system release	Availability
advance-toolchain-at5.0-devel	5.0-7	6.1, 6.2	See Note b.
advance-toolchain-at5.0-runtime			
advance-toolchain-at5.0-selinux			
gcc-c++ <sup>a</sup>	4.4.5-6.el6	6.1	Operating system package
	4.4.6-3.el6	6.2	
ibm-java-x86_64-sdk	SR 11	6.1, 6.2	InfoSphere Streams package
ibm-power-at-repo <sup>b</sup>	1.0.0-1	6.1, 6.2	InfoSphere Streams package
libcurl-devel	7.19.7-26.el6	6.1	Operating system package
	7.19.7-26.el6_1.2	6.2	
perl-XML-Simple	2.18-6.el6	6.1, 6.2	InfoSphere Streams package
policycoreutils-python <sup>c</sup>	2.0.83-19.8.el6_0	6.1	Operating system package
	2.0.83-19.18.el6	6.2	
selinux-policy <sup>d</sup>	3.7.19-126.el6_2.10	6.2	Operating system package
selinux-policy-targeted <sup>d</sup>			
vacpp.rte	11.1.0.3-110614	6.1, 6.2	InfoSphere Streams package
xlsmplib.rte <sup>e</sup>	2.1.0.3-110611	6.1, 6.2	InfoSphere Streams package
xlsmplib.rte <sup>e</sup>			

- a. The maximum version of the gcc-c++ RPM must be earlier than Version 4.5. Version 4.5, or later cannot be used with InfoSphere Streams.
- b. The Advance Toolchain compiler is required for IBM POWER7 installations. To install the compiler RPMs, you must first install the ibm-power-at-repo RPM in the InfoSphere Streams installation package. Installing the ibm-power-at-repo RPM sets up a yum repository for accessing and installing the compiler RPMs. The advance-toolchain-at5.0-devel and advance-toolchain-at5.0-runtime RPMs are required for all installations. If SELinux is running in enforcing or permissive mode, the advance-toolchain-at5.0-selinux RPM is also required.
- c. If SELinux is running in enforcing or permissive mode, the policycoreutils-python RPM is required. For more information about using SELinux with InfoSphere Streams, see this website: <http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>.
- d. The selinux-policy and selinux-policy-targeted RPMs are installed by default. However, if SELinux is running in enforcing or permissive mode, you must update these RPMs to Version 3.7.19-126.el6\_2.10, or later. Updates are available on your operating system update site. The preferred method for installing these RPMs is by using the **yum update** command. Before using this command, you must set up a yum repository for your operating system update site.
- e. You must install the xlsmplib.rte RPM first, and then the xlsmplib.rte RPM.

# Installing InfoSphere Streams

Use the procedures that are described in this section for a new installation of IBM InfoSphere Streams. A new installation can be used for installing the product for the first time or installing with a different installation owner from the previous installation.

The following methods can be used to install InfoSphere Streams:

- ▶ Interactive GUI installation
- ▶ Interactive console installation
- ▶ Silent installation

In this section, we describe the installation process that is used for the Interactive GUI installation method.

## Before you begin

Ensure that you satisfy the prerequisites that were described in “Required Red Hat Package Manager for InfoSphere Streams” on page 253.

## Procedure

The following procedure assumes that you are running as user root. If SELinux is enabled (default system setting), you must install as user root because the installation program can install the Streams SELinux policy module and label the installation directory with the required file contexts:

1. To extract the contents of the InfoSphere Streams installation package, enter the following command:

```
tar -zxvf product-installation-package-name.tar.gz
```

The **tar** command creates the `StreamsInstallFiles` directory, which contains the self-extracting binary file `InfoSphereStreamsSetup.bin` and other installation files.

2. To start the installation, run the InfoSphere Streams self-extracting binary file by entering the following commands:

```
- cd product-installation-package-directory/StreamsInstallFiles/  
- ./InfoSphereStreamsSetup.bin
```

If an X Window System is installed, the installation starts in interactive GUI mode. If an X Window System is not installed, the installation starts in interactive console mode.

**Important:** Depending on the location of the system, you might see a language selection prompt. Select the language and click **OK**.

Figure A-1 shows the initialization of the InfoSphere Streams installation process.

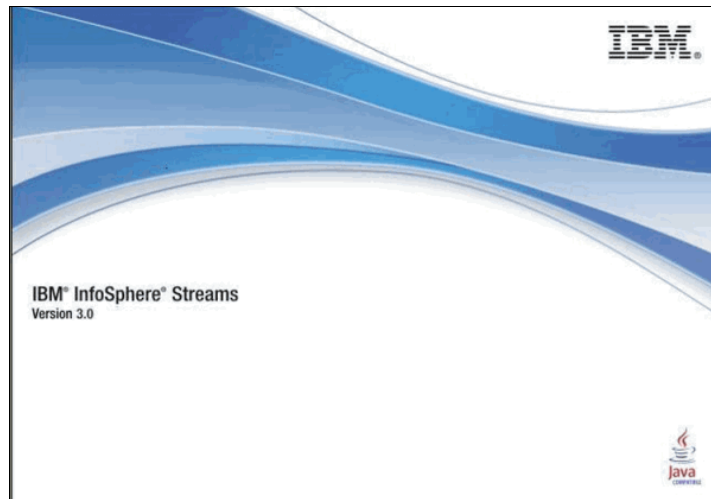


Figure A-1 InfoSphere Streams installation process

3. After you review the information about the Introduction page, click **Next** to continue. Figure A-2 shows the Introduction page.

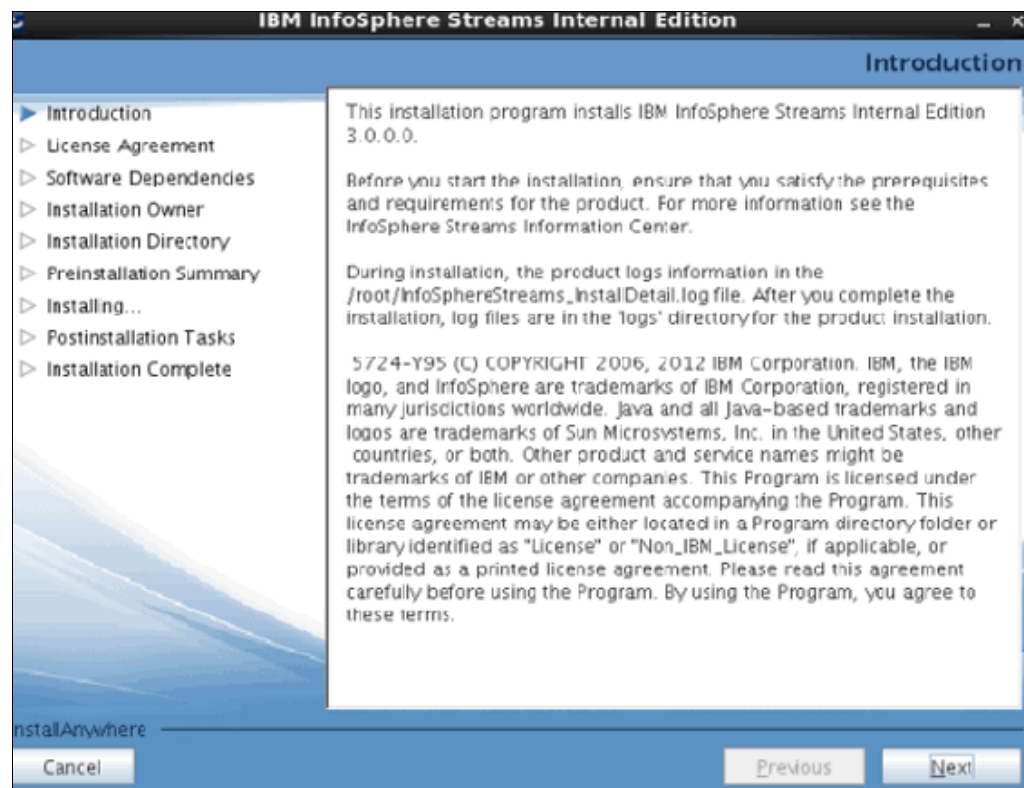


Figure A-2 Introduction

4. Review the license agreement as show in the Figure A-3 on page 260.

To continue the installation, click **I accept the terms in the license agreement** and then click **Next**.

If you do not accept the license agreement, click **I do not accept the terms in the license agreement** and then click **Cancel** to exit the installation utility.

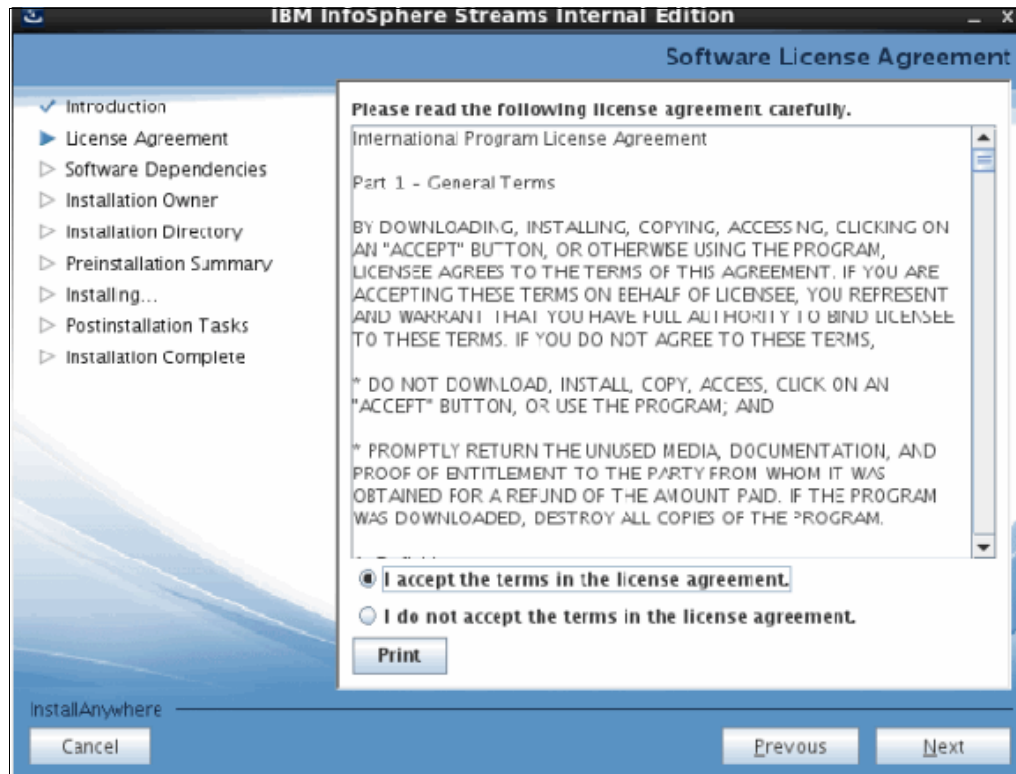


Figure A-3 License Agreement

**Important:** You must accept the license agreement to continue the installation.

The installation utility checks your system configuration for any incompatibilities with InfoSphere Streams and displays the results on the Software Dependencies page.

If an incompatibility is found, the installation utility displays the following error or warning message:

- If an error is displayed, you cannot continue the installation until the error is fixed. For example, if you try to install 64-bit InfoSphere Streams on a 32-bit system, the installation cannot continue.
- If a warning is displayed, you can continue or cancel the installation. If you continue, the installation completes successfully, but the incompatibilities must be corrected before you run any applications in an InfoSphere Streams instance.

The installation utility displays information about required RPMs and their status on your system. The utility also logs this information in the installation summary log. You can complete the installation without correcting any RPM dependencies, but you must correct any issues before you use InfoSphere Streams.

If the installation utility cannot find a host file on your system, it attempts to create a default host file for you in the `~installation-owner/.streams/config/` directory. If the utility has a problem resolving the host name and cannot create a default host file, a warning message is displayed and logged. You can continue or cancel the installation. If you continue, the utility does not generate a default host file. You must manually create a host file after you complete the installation.

5. To save the dependency check results in a file, click **Save result to file**. To continue the installation, click **Next.**, as shown in Figure A-4.

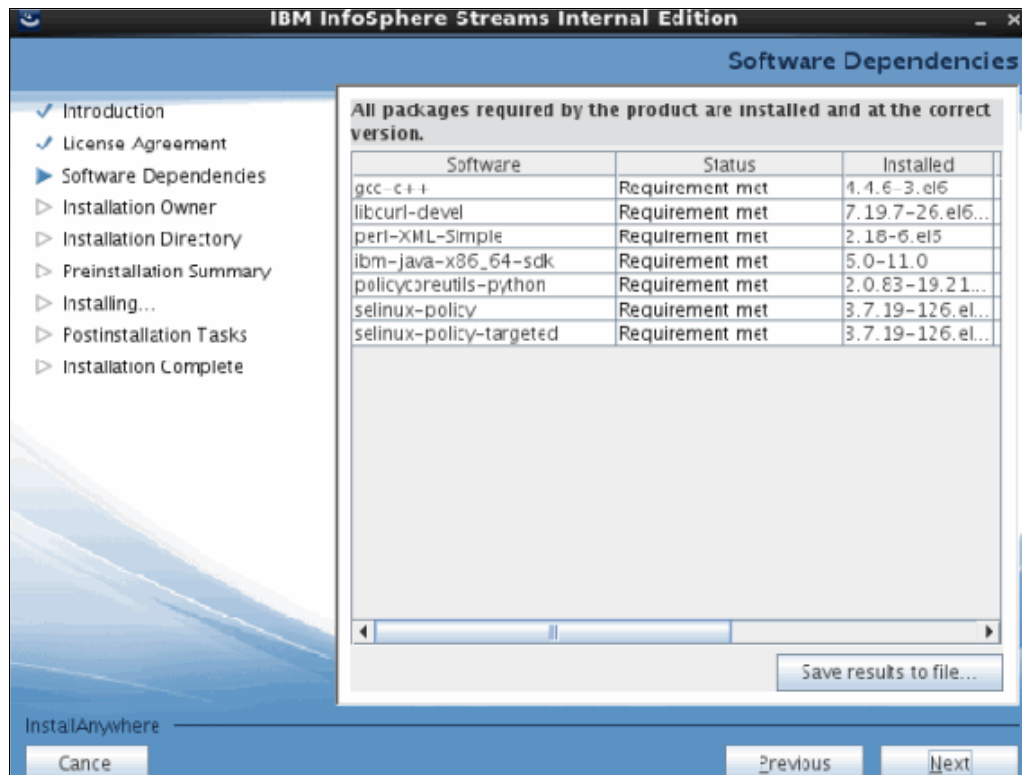


Figure A-4 Software Dependencies

If you are running the installation utility with root authority, the utility prompts you to specify the user and group that owns the installed files. The root user cannot own the installed files.

If you are running the installation utility as a non-root user, the Specify File Owner page is not displayed because you are the owner of the InfoSphere Streams installation.

Figure A-5 shows the details of the Installation Owner page.

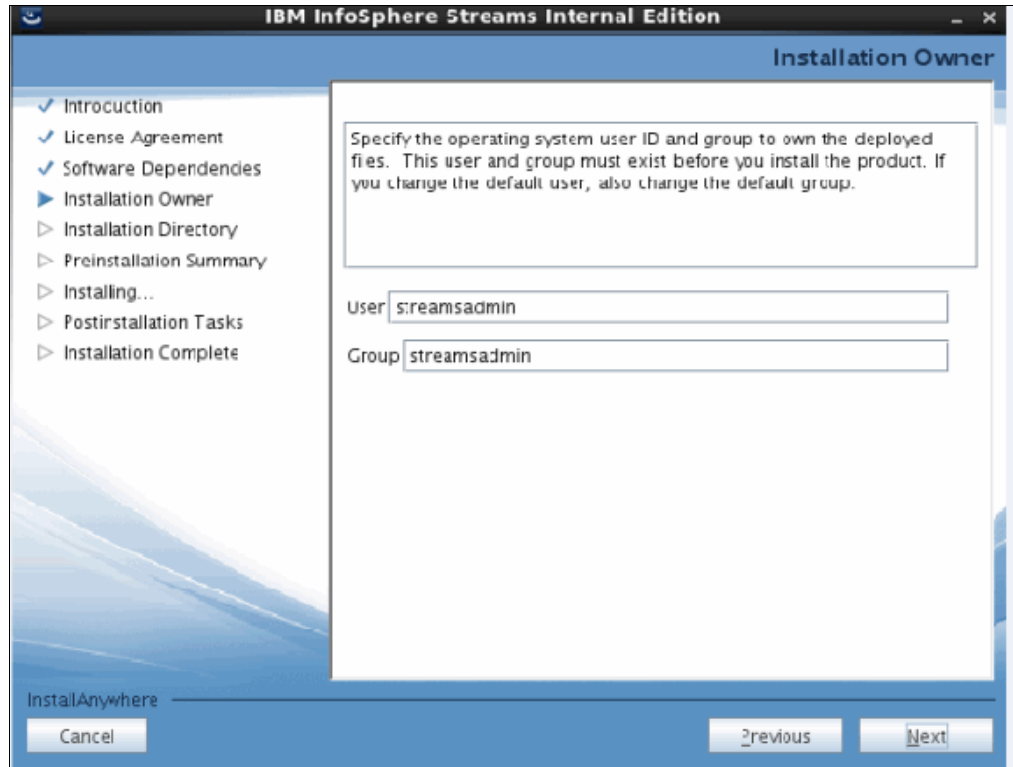


Figure A-5 Installation Owner

6. On the Installation Directory page, click **Next** to accept the default installation directory or enter the absolute path of another directory.

**Important:** The installation directory cannot contain spaces or the following characters: ' " ` \$ | & ? \* < > \. If you include spaces or non-supported characters, the installation utility displays an error message and requires you to enter a valid directory or cancel the installation.

If you intend to use the InfoSphere Streams installation in a multiple-host environment, you must install the product into a shared file system that is available to each host or install InfoSphere Streams in the same directory path on each host.

Figure A-6 shows the Installation Directory page.

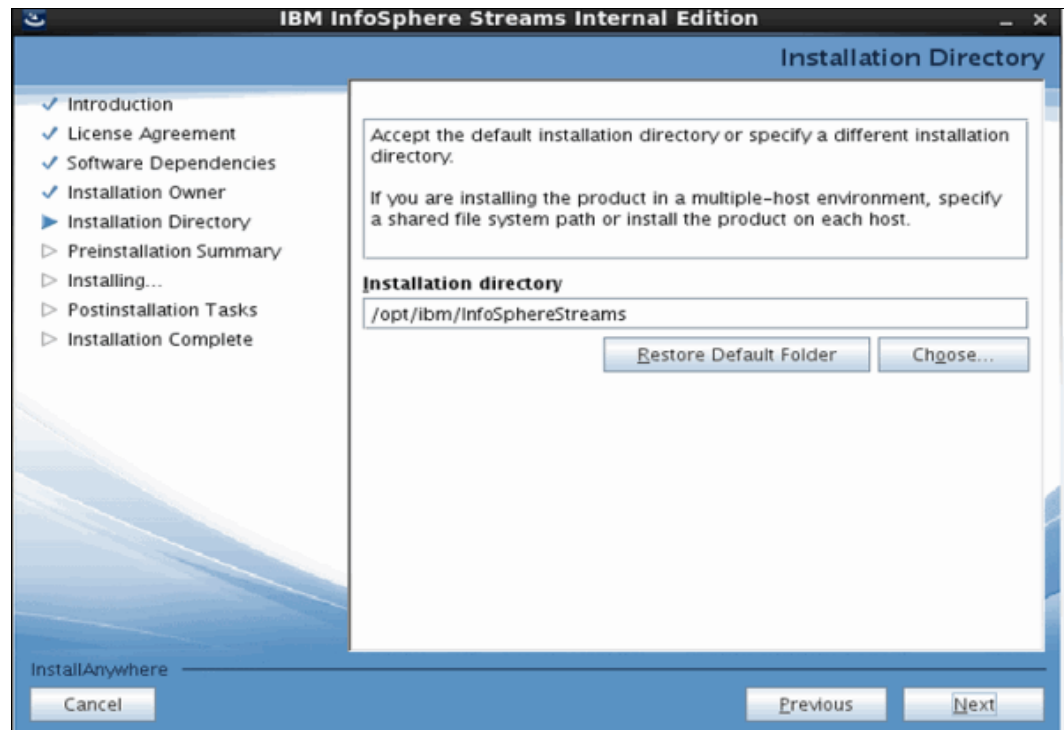


Figure A-6 Installation Directory

7. The installation utility displays the Preinstallation Summary page for your review before the installation is started. After reviewing this information, click **Install** to start the installation. The utility displays the status of the installation.
8. If InfoSphere Streams is already installed in the location that you specify, you can provide a new location, uninstall the existing installation, or stop the installation utility. If you choose to uninstall the existing installation, the utility removes the existing installation files and replaces them with the new installation files.

To cancel the installation and exit the utility, click **Cancel** and then **Exit**.

To specify a new location, click **Select new location**.

To uninstall the current installation and continue the installation, click **Uninstall** and then **Continue**.

If you choose to uninstall and continue, a Stop Running Instances Warning message is displayed.

**Important:** If instance owners do not stop all running applications and instances, some files for the previous installation might not be removed. If the new installation is installed in the same location, the presence of old files can result in unpredictable application behavior

To cancel the installation and exit the utility, click **Cancel** and then **Exit**.

To continue the installation, click **Continue**.

Figure A-7 shows the Installation Progress window.

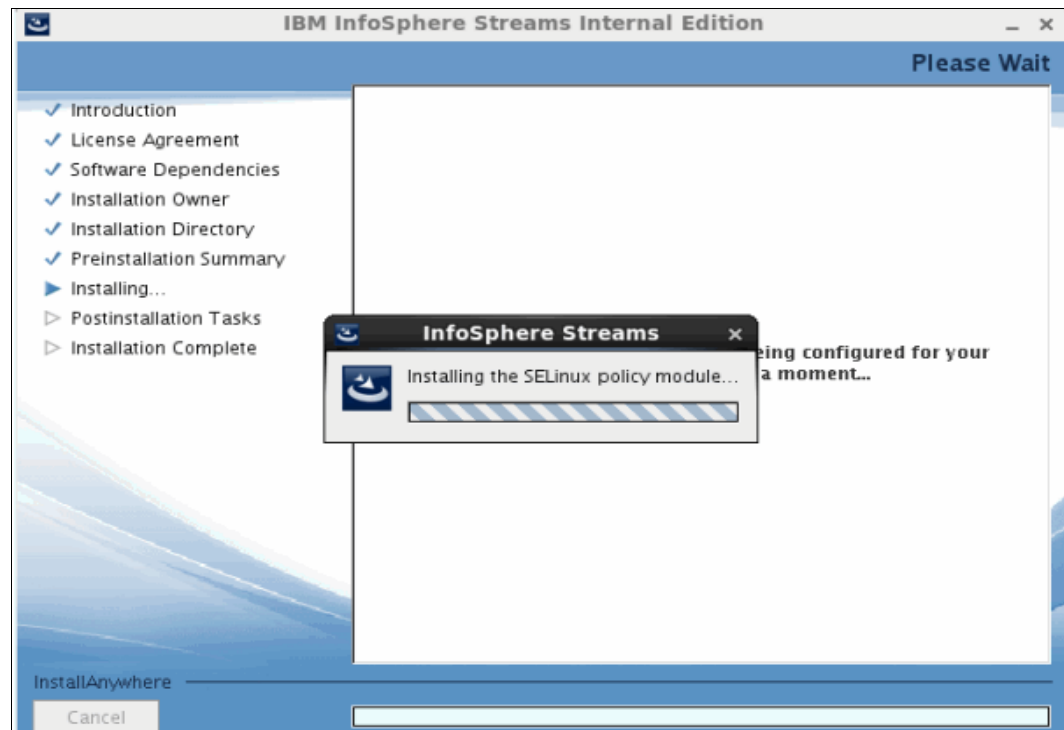


Figure A-7 Installation progress

9. Review the following options in the Postinstallation Tasks window:
  - To open a browser window that shows the Release Notes in HTML format, select **View release notes**.
  - By default, the **Launch First Steps** option is selected. The First Steps application helps you to complete the required post-installation tasks for InfoSphere Streams.
  - To continue, click **Next**.

The Installation Complete page is displayed, and the First Steps application opens in a new window. If you selected **View release notes**, the Release Notes are shown in a browser window.



Figure A-8 shows the Postinstallation Tasks window.

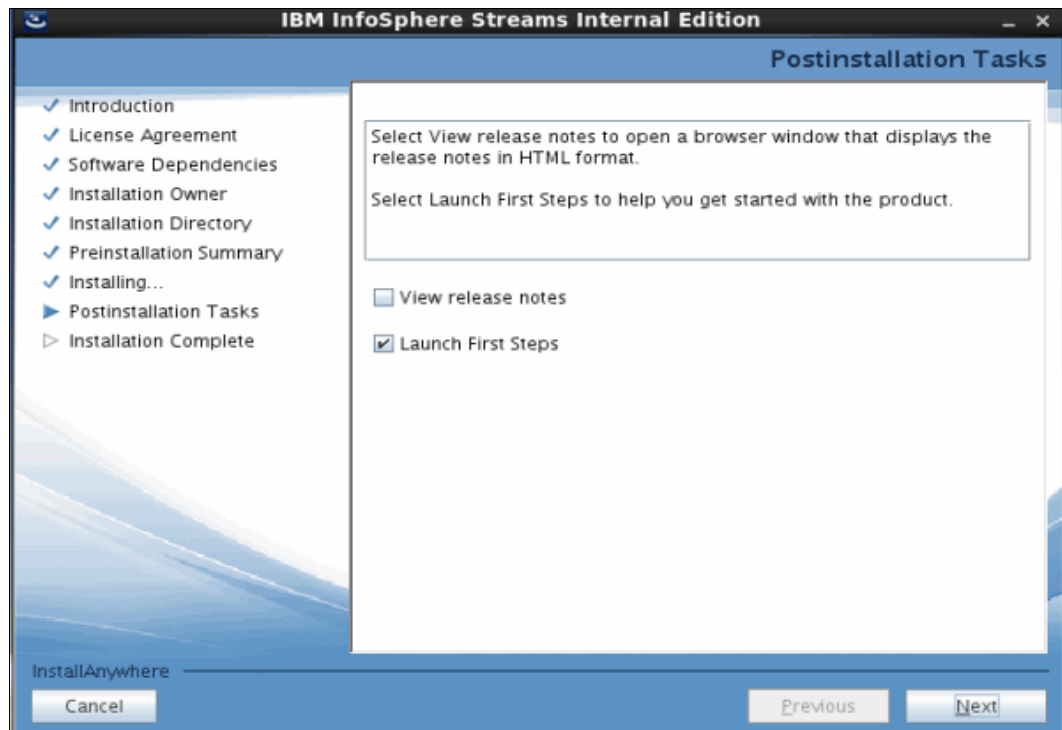


Figure A-8 Postinstallation Task

10. After reviewing the information on the Installation Complete page, click **Done** to exit the installation utility. Figure A-9 shows Installation Complete window.

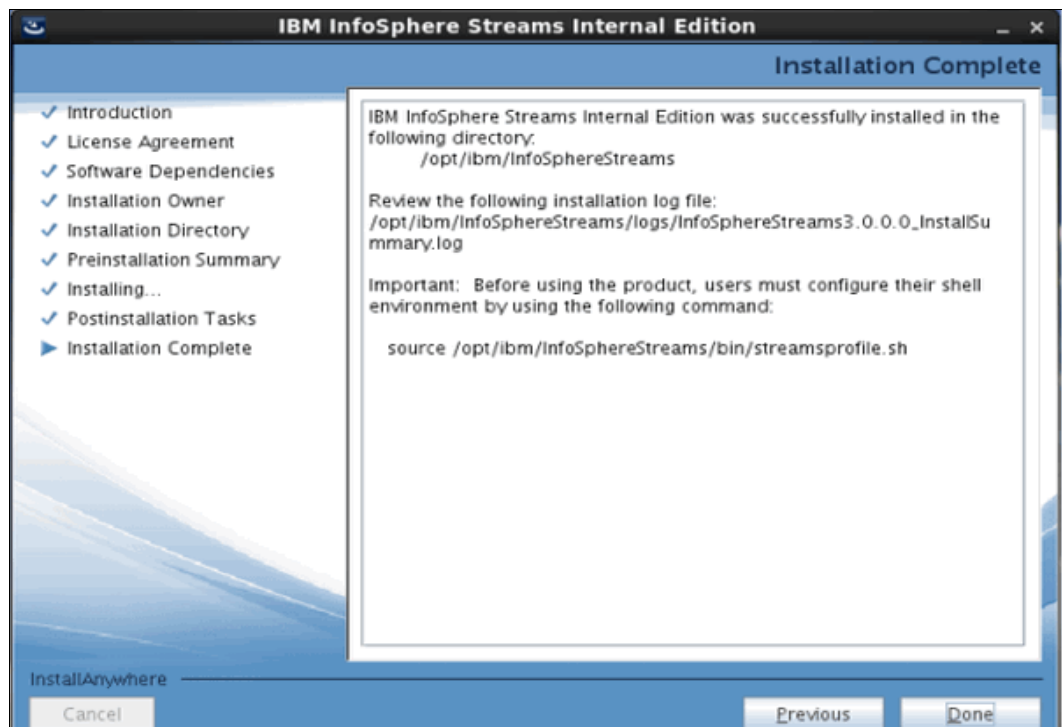


Figure A-9 Installation Complete window

11. If the installation utility showed warnings or errors during the installation, check the installation summary log file and resolve any issues before continuing. This log file is in the *product-installation-directory/logs* directory.

**Tips:** If the summary log file indicates that you need to run the `streamspostinstall.sh` script, see Appendix , “Postinstallation script for creating a default host file” on page 267 for more information.

After the Installations is completed successfully, the system automatically starts the Streams First Steps window for the configuration process and the users can set up the environment with the simple click of a button, as shown in Figure A-10. For more information about this configuration, see “IBM InfoSphere Streams First Steps configuration” on page 267.

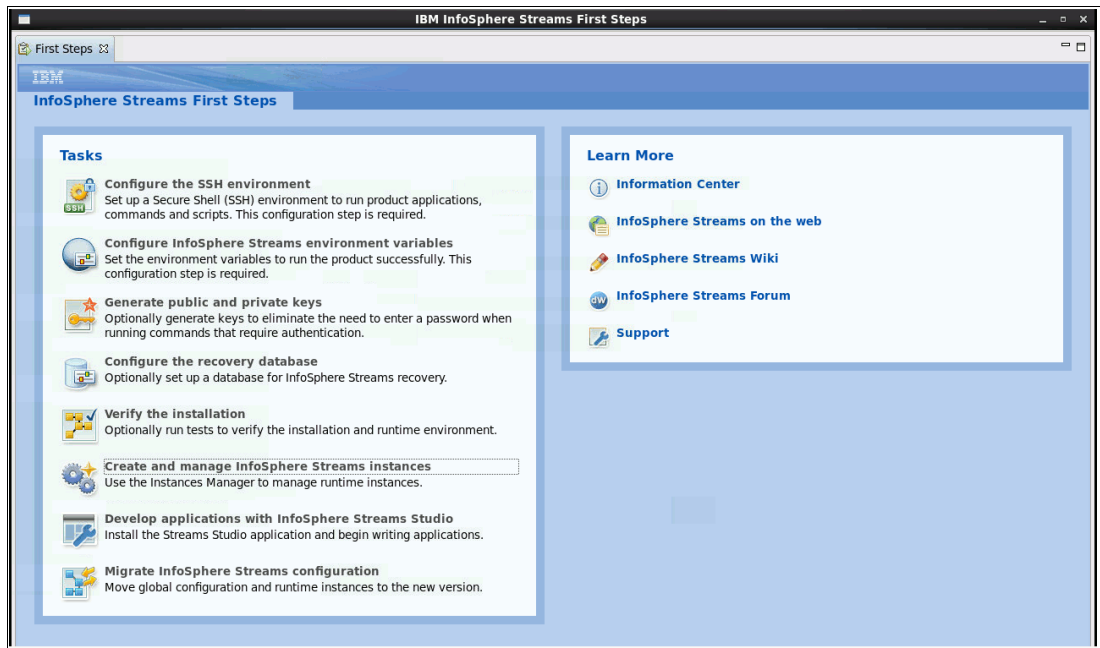


Figure A-10 InfoSphere Streams First Steps

## Postinstallation script for creating a default host file

If the InfoSphere Streams installation utility encounters a problem while creating a default host file in the home directory of the installation owner, running the postinstallation script might correct the problem. The summary log file in the *product-installation-directory/logs* directory indicates if the installation owner must run this script. If this problem occurs, the installation utility places the default host file in the product installation directory. This script copies the host file from the product installation directory to the *installation-owner/.streams/config/* directory.

To run the postinstallation script, enter the following commands:

```
cd product-installation-directory/bin
```

```
./streamspostinstall.sh
```

The following options are available for the commands:

- ▶ **-d** Default: Create a default host file in the *installation-owner/.streams/config/* directory.

**Tip:** This option creates a host file only if there are no issues resolving the host name. If the summary log file indicates that the installation utility had a problem resolving the host name, you must manually create a host file.

- ▶ **-v level** : Verbose option that displays more messages. The following values can be used for *level*:
  - Includes more informational messages.
  - Includes more debug messages.
- ▶ **-h**: Display the help information.

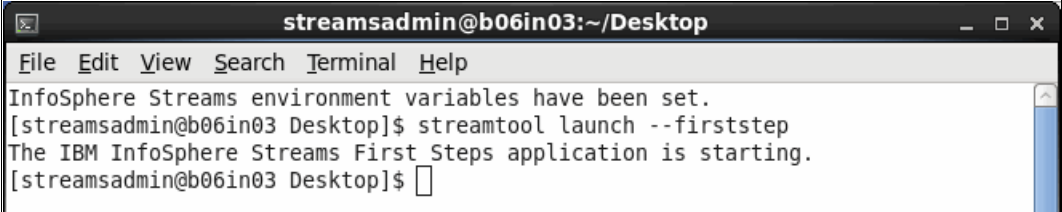
If the script completes successfully, the following message is displayed:

The *product-installation-directory/config/hostfile* default host file was copied.

## IBM InfoSphere Streams First Steps configuration

After the installation completes, all users are required to perform some configuring. IBM InfoSphere Streams (Streams) First Steps is included as a required new feature on Streams 3.0. You can configure the Streams environment ready-to-use by few mouse clicks.

After you finish your Streams installation Streams, First Steps automatically starts. You also can start First Steps at any time by using the **streamtool** command. The command to launch First Steps is shown in Figure A-11.



```
streamsadmin@b06in03:~/Desktop
File Edit View Search Terminal Help
InfoSphere Streams environment variables have been set.
[streamsadmin@b06in03 Desktop]$ streamtool launch --firststep
The IBM InfoSphere Streams First Steps application is starting.
[streamsadmin@b06in03 Desktop]$
```

Figure A-11 Starting First Steps

After running the script that is shown in Figure A-11, IBM InfoSphere Streams First Steps appears on your window, as shown in Figure A-12 on page 268.

**Important:** All Streams users are required to run First Steps to complete some set up tasks.

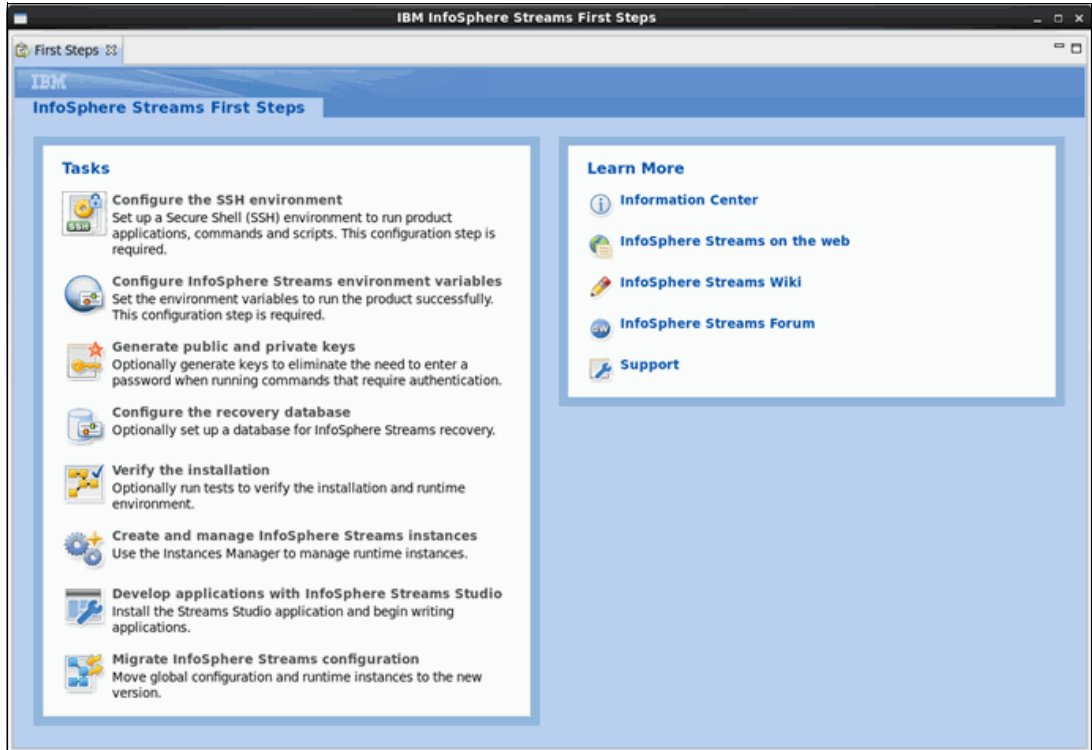


Figure A-12 IBM InfoSphere Streams First Steps window

We describe each item in the following section.

## Configure the Secure Shell environment

After installation, all users are required to configure the Secure Shell (SSH) environment. Streams 3.0 enables easy setup of SSH.

In IBM InfoSphere Streams First Steps window (as shown in Figure A-12), click **Configure the SSH environment**. The Configure the SSH environment window appears, as shown in Figure A-13. You can choose a DSA or RSA for SSH key types here. Click **OK**.



Figure A-13 Configure the SSH environment

## Configure InfoSphere Streams environment variables

You are required to configure Streams environment variables after installation. Click the **Configure InfoSphere Streams environment variables** menu on the First Steps window (as shown in Figure A-12 on page 268). You see a window (as shown in Figure A-14) that explains how to set Streams environment variables. Click **OK**.



Figure A-14 Configure InfoSphere Streams environment variables window

## Generate public and private keys

This setup can help you to avoid entering a password each time you run commands that require authentication. Click **Generate public and private keys** on IBM InfoSphere Streams First Steps window (see Figure A-12 on page 268). Figure A-15 shows the directory of public and private keys that are stored. Click **OK**. All users should consider completing this step if they use the default authentication and do not want to be prompted for a password.

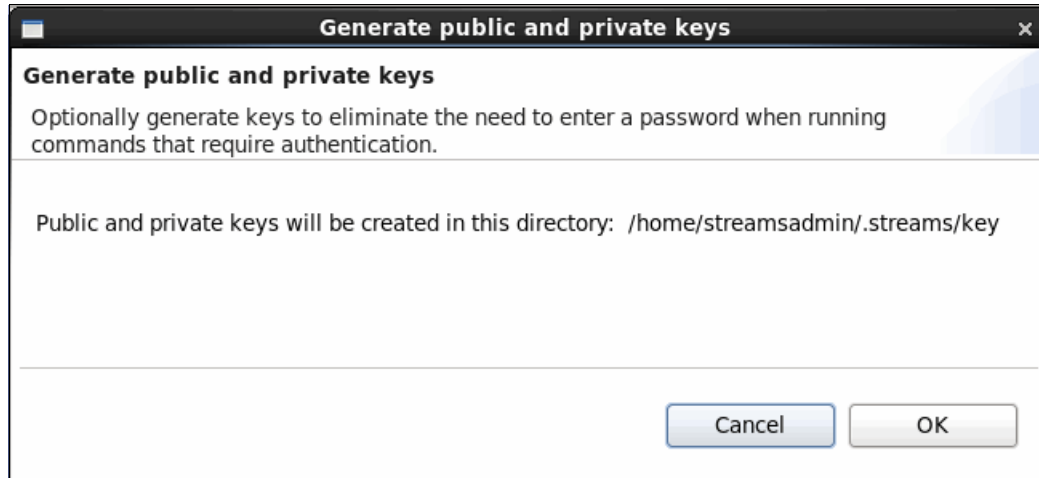


Figure A-15 Generate public and private keys window

## Configure the recovery database

By using Streams 3.0, you can perform failover recovery by using an IBM DB2 database. Only the installation owner can complete setting up the recovery database. The DB2 database must be created by using the DB2 interfaces. Also, there must be connectivity to the DB2 server. Click **Configure the recovery database** in IBM InfoSphere Streams First Steps window (see Figure A-12 on page 268), and Figure A-16 is shown. Enter the required information, including the database name, host, port, user ID, and password. Click **OK**.



**Configure the recovery database**

Configure an existing database to be used for InfoSphere Streams recovery.

This connection information will be stored in the runtime database configuration file.

Database

Name:

Host:

Port: 50000

User ID:

Password:

?

Cancel OK

Figure A-16 Configure the recovery database window

By using the Configure the recovery database window, you can customize the DB2 database for streams, set up a database configuration file, and test the connectivity to the database. You can also use this window to change the database configuration. For example, you can change the database name. This window customizes the new database for streams recovery, updates the database configuration file, and tests the connectivity to the database.

If you need to change the user's password, first change it on the DB2 server. Then use this window to update the database configuration file. If setting up a new database or changing any of the prompts under Database, you are prompted for the DB2 administrator ID and password. This information is used to customize the database.

If you are not the installation owner and a recovery database is set up, you can test the connectivity by using this window.

## Verify the installation

By using this menu, you can perform some of the test and verification tasks of your Streams environment. Select **Verify the installation** option on IBM InfoSphere Streams First Steps window (see Figure A-12 on page 268). A new window (see Figure A-17) is displayed that shows you what this option does to test your environment. Click **OK**.

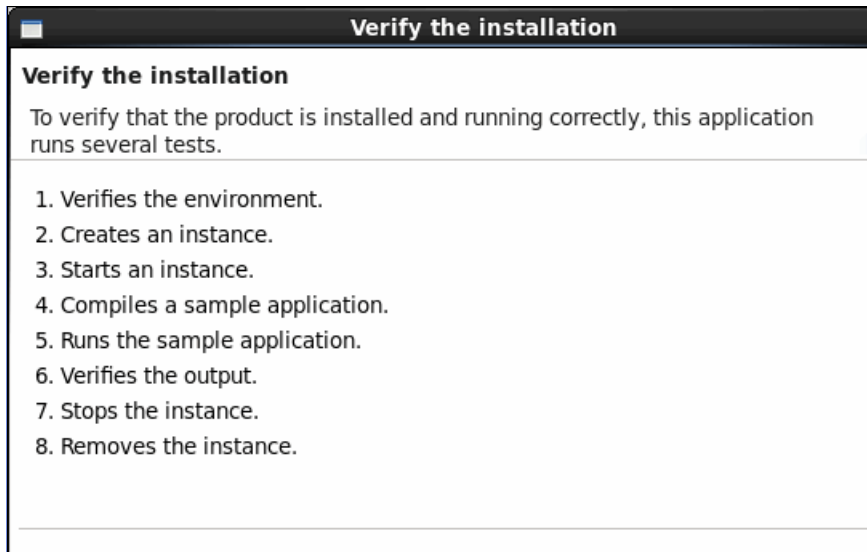


Figure A-17 Verify the installation window

The Progress Information window appears, as shown in Figure A-18.

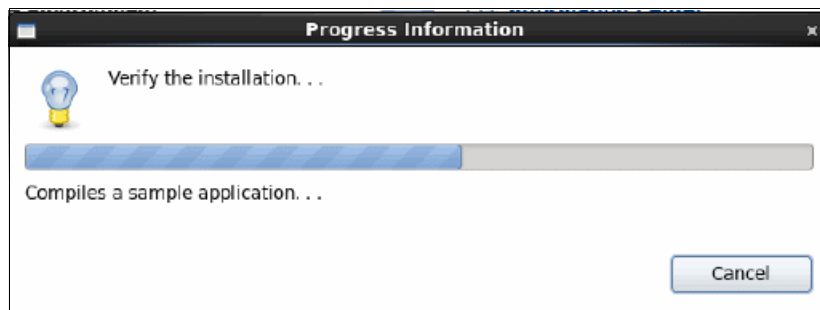


Figure A-18 Progress Information window

Figure A-19 shows that your verification tests completed successfully.

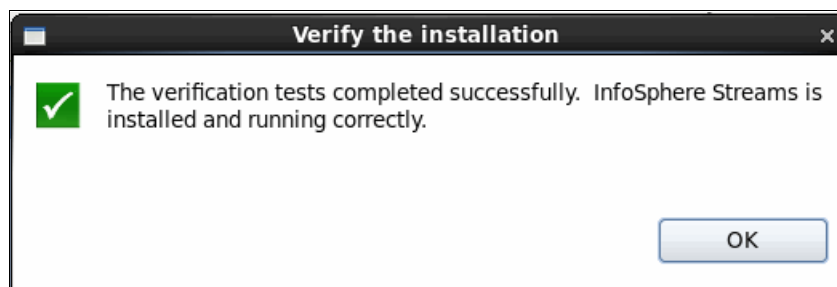


Figure A-19 Verification tests completed



## Creating and managing InfoSphere Streams instances

For more information about creating and managing InfoSphere Streams instances, see 8.1.1, “Instances Manager” on page 201.

## Developing applications with InfoSphere Streams Studio

This option installs Streams Studio on your environment on your first run. After the installation completes, the process directs you to the installed Studio instance. Click **Develop applications with InfoSphere Streams Studio** on IBM InfoSphere Streams First Steps window (see Figure A-12 on page 268). You can then choose the installation directory, as shown in Figure A-20. Click **OK**.

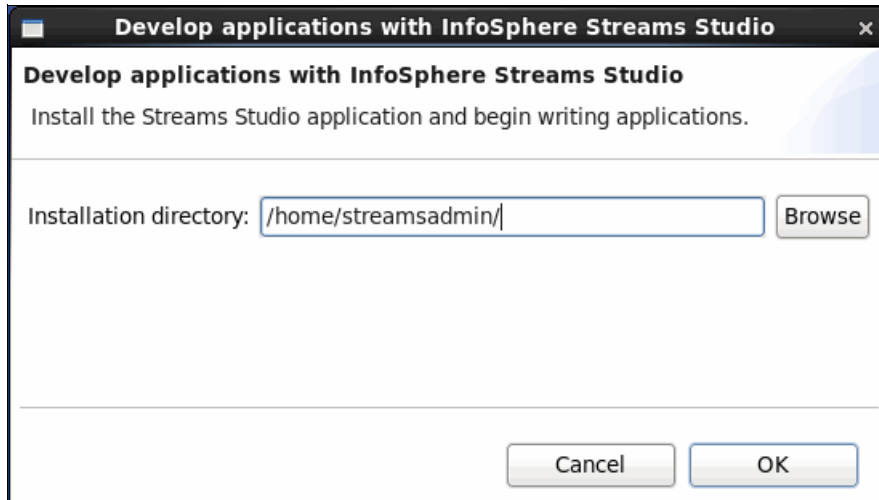


Figure A-20 Streams Studio installation

The Progress Information window appears, as shown in Figure A-21. Wait until the installation completes before you proceed.

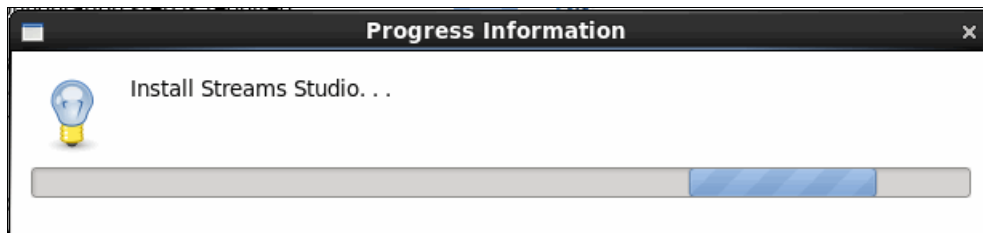


Figure A-21 Progress Information Streams Studio installation window

After the Streams Studio installation completes, you see the Workspace launcher window (as shown in Figure A-22). You can then modify your workspace location. Click **OK**.

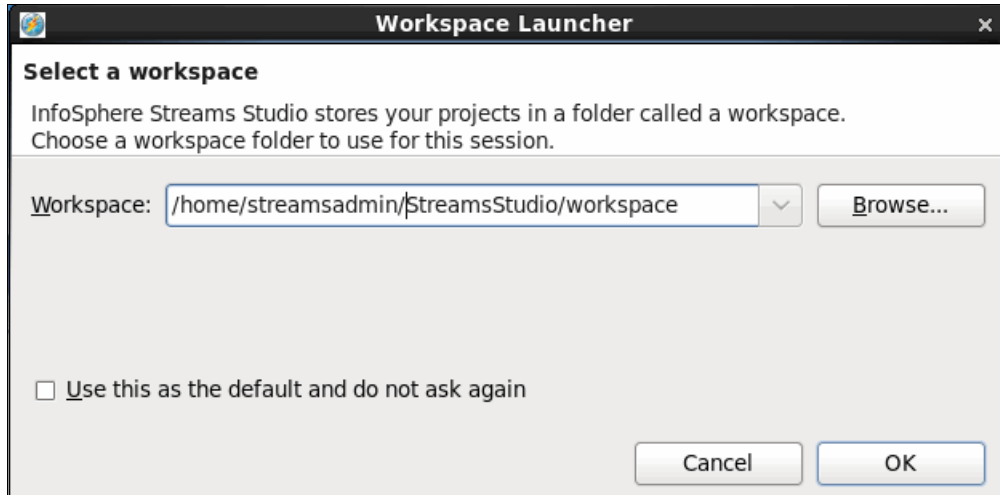


Figure A-22 Workspace launcher

You see the Streams Studio window that features the Task Launcher for Big Data tab, as shown in Figure A-23. You are now ready to develop applications by using IBM InfoSphere Streams.

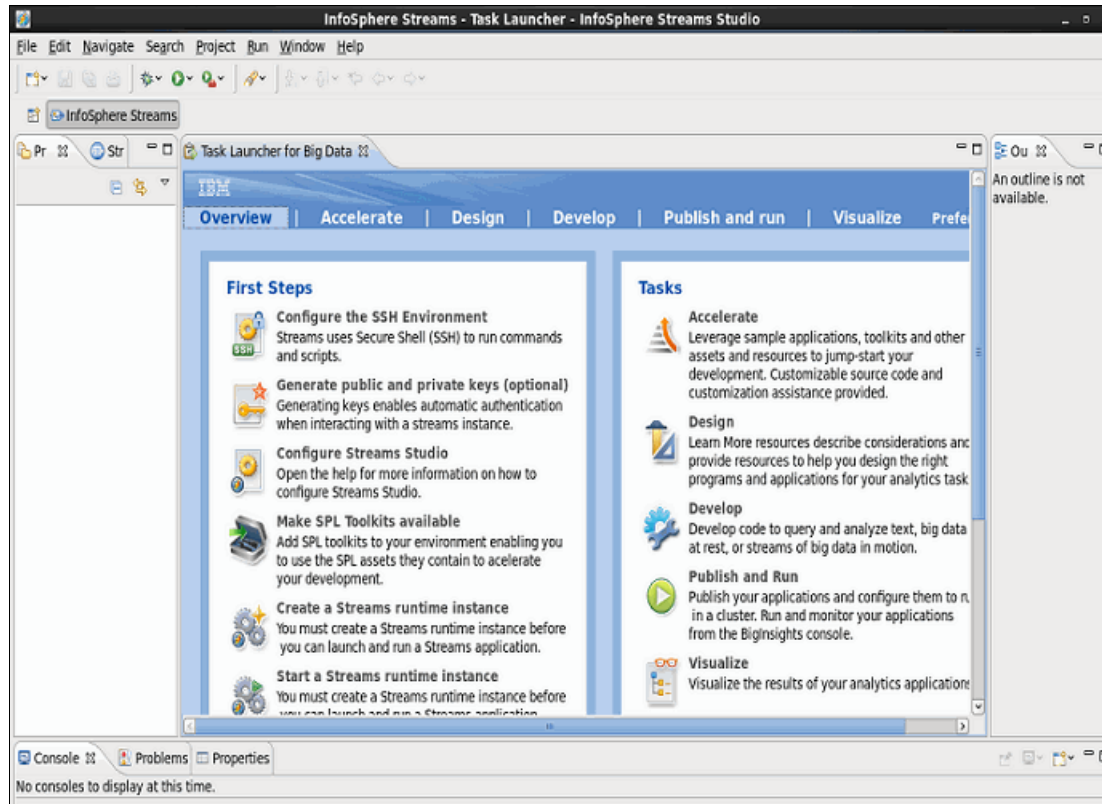


Figure A-23 Streams Studio window

## Migrate InfoSphere Streams configuration

For more information about the migrate InfoSphere Streams option, see *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888, or the Information Center at this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

**Important:** You must run this option if you are upgrading from a previous Streams version to Streams 3.0.





# **IBM InfoSphere Streams security considerations**

In this appendix, we describe the security aspects of Streams. We cover Security-Enhanced Linux (SELinux) to control and manage internal system with external system. We also include authentication features that are provided by Streams when Streams is configured to work with Lightweight Directory Access Protocol (LDAP) and Pluggable Authentication Modules (PAM). We also describe the Access Control List (ACL) and the audit log file.

# Security-Enhanced Linux for Streams

SELinux is one of the Red Hat Package Manager (RPM) packages that is required before you install Streams (for more information, see “Required Red Hat Package Manager for InfoSphere Streams” on page 253).

By using SELinux, you can control and restrict the operating system resource that processing elements (PEs) can access. You also can limit the ability of the PEs to connect with other external systems. The following levels of containment of the PEs are supported by SELinux:

- ▶ Unconfined
- ▶ Confined
- ▶ Confined and vetted

## Unconfined PEs

All PEs run in an unconfined domain, which means PEs can transfer data and connect over any network. However, there are some restrictions; for example, they cannot change Streams installation files to avoid accidental or malicious harm to these files.

## Confined PEs

PEs run in a confined domain that is defined by Streams policy module. You must define which external data or non-Streams resources you want to interact with PEs. Figure B-1 shows the interaction between PEs and the external data system.

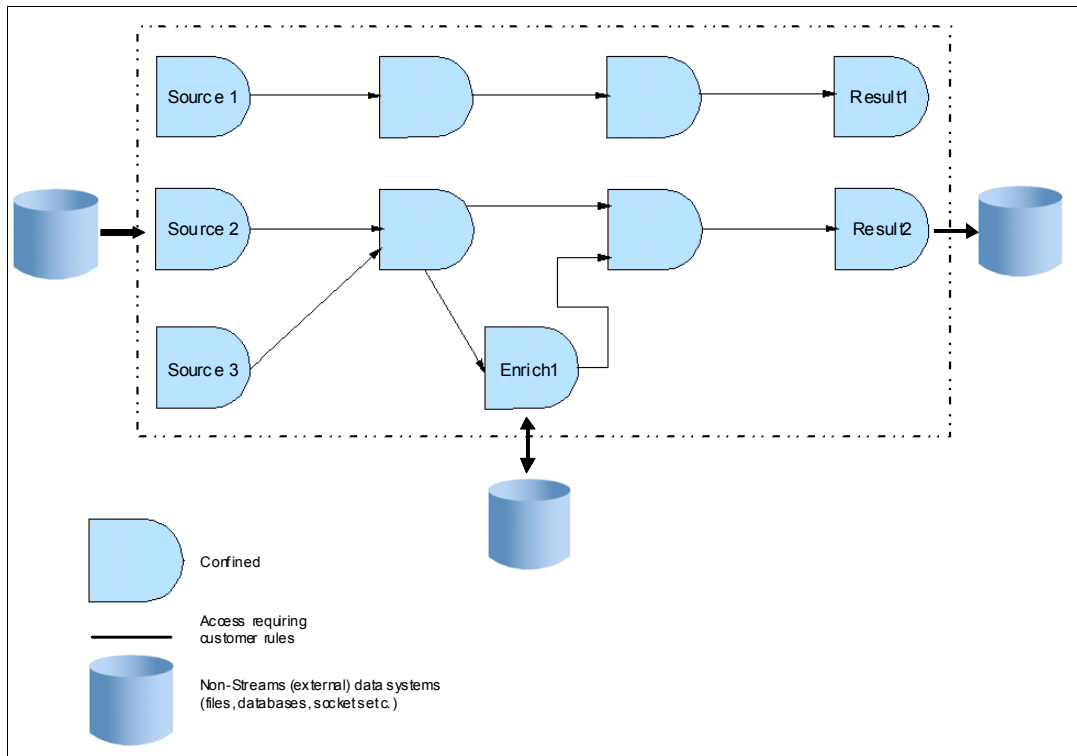


Figure B-1 Streams confined domain

## Confined and vetted PEs

As with the confined domain, you can limit the ability of the PEs to interact with the external system by using the confined and vetted containment. The difference is, you can mark trusted PEs as vetted in a vetted domain so that PEs that are labeled as vetted can interact with restricted external data system.

Figure B-2 shows confined and vetted PEs with the external system. In a vetted domain often has more rights because it can communicate with the external system.

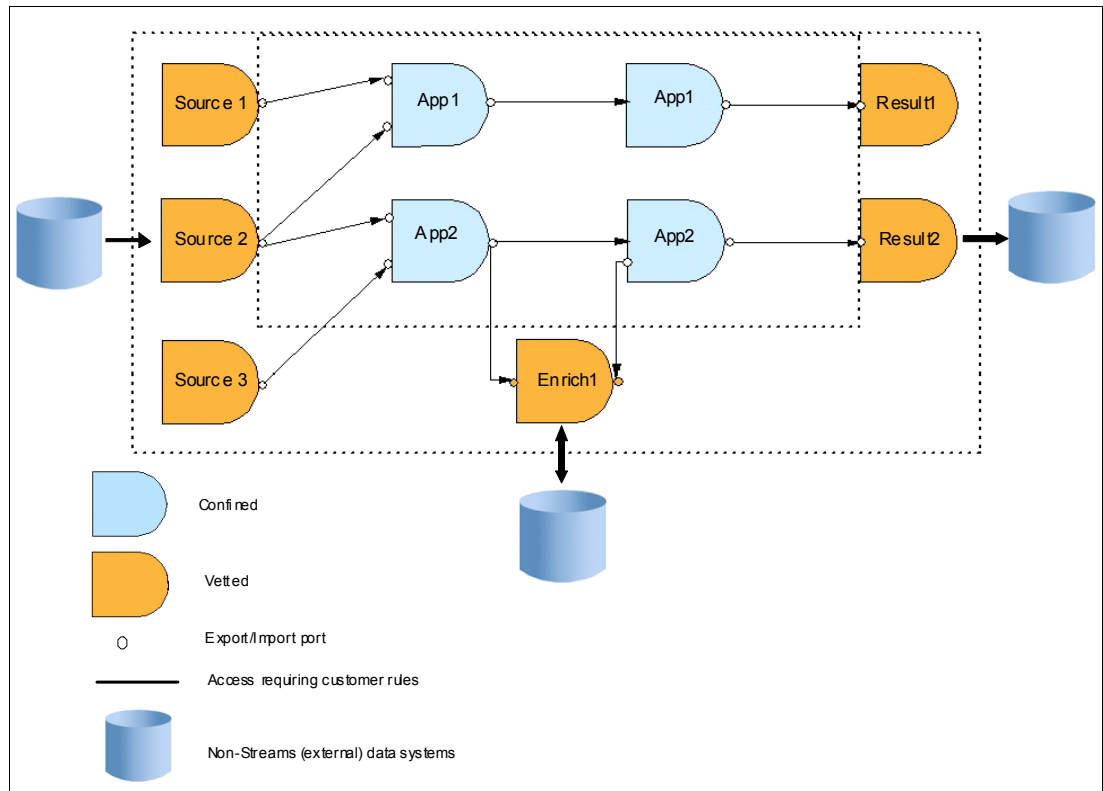


Figure B-2 Streams confined and vetted domain

## SELinux advance setting

To enable these PEs containments, you must define security policies for Streams applications. For more information about enabling SELinux on your Streams environment, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

# User authentication

## PAM authentication service

By default, the InfoSphere Streams security templates use the PAM authentication service, and instances are created with RSA authentication enabled. This option eliminates the need for users to enter a password when InfoSphere Streams commands that require authentication are run.

The PAM authentication service uses the PAMs for Linux to define users and groups for Streams. An instance is defined to use a PAM service that supports user name and password authentication, defaulting to the login service. The system administrator can define a PAM service that is specific to Streams or uses an existing service for their environment. By using the same or a related PAM service that is used for operating system access, users can access Streams by using their operating system password.

If your organization requires a more refined authentication setup, you might need to define new PAM services. You can extend PAM by defining new services and writing new modules. For more information about configuring PAM, see the PAM documentation.

## LDAP authentication service

LDAP authentication can be used if an enterprise-wide LDAP server is available and the use of PAM authentication is not desirable in this context. For example, LDAP is a good option if Streams users are not allowed to connect directly to the Streams cluster.

If LDAP is the authentication back end for PAM, it is recommended that you use the PAM authentication service instead. For LDAP, users must enter a password when they use **streamtool** commands. If the user login is different from the LDAP login, the user must also specify a user name on the **streamtool** command by using the **--User** option.

For more information about configuring LDAP authentication for Streams, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

# User authorization

InfoSphere Streams uses access control lists (ACLs) for authorization. An ACL is composed of the type of instance object to secure and the actions that a group or user is authorized to perform against the object. You can choose a private or shared security template when an instance is created.

The following predefined authorization policies are available:

- ▶ Private: Use for Instances with a single user (the Instance owner)
- ▶ Shared: For Instances with multiple users

The following Streams objects are controlled by the authorization policy:

- ▶ Instance: The overall administrative scope
- ▶ Config: The Instance configuration
- ▶ Hosts: The set of hosts for the Instance, and services on those hosts
- ▶ jobs: The list of jobs that are submitted for the Instance
- ▶ job\_<x>: A specific job and its Processing Elements (PEs).
- ▶ jobs-override: Controls the ability to override Host load protection when jobs are submitted



Each of these Streams objects include their own ACL permission. For more information about ACLs, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>

## **Audit log file for Streams**

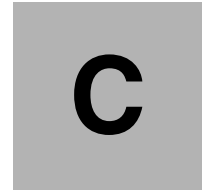
Streams can create an audit log file to capture the security events for an instance. By default, audit logging is not enabled. If audit logging is enabled, Streams creates one audit log file for each instance.

Streams creates the audit log file when the instance is started, and adds one or more log entries for every security event. If the audit log file exists, Streams appends to the file. Streams does not truncate or delete the audit log file.

For more information about audit log files for Streams, see this website:

<http://pic.dhe.ibm.com/infocenter/streams/v3r0/index.jsp>





# Commodity purchasing application demonstration

This appendix provides a sample commodity purchasing application, including an application overview. We also describe the process that is used to import and run the application in your environment.

## Application overview

The CommodityPurchasing application is provided for the purposes of demonstration and education. It shows a stream processing application in action, and how it is constructed. In addition, running and inspecting the sample application shows you to the various features of the Streams product.

A fictitious commodity, the Infoberry, is used as an example in the application. Imagine that your company is a producer of the world's finest Infoberry juice. This fact makes the Infoberry a critical commodity for your business. The success of your product relies heavily upon the quality and availability of Infoberries. The better the berry, the better the juice that can be made, and a steady supply of these berries is needed to support juice production. Infoberries feature the following unique properties:

- ▶ They grow year-round in almost any climate. They are the fruit of a special evergreen plant that produces berries in all seasons.
- ▶ The fragile nature of the Infoberry requires that it is shipped immediately after harvesting. This fact means that Infoberries are harvested only after an order for them is placed.

The quality of the Infoberry is predominantly affected by the weather. The following weather factors can affect the quality of the Infoberry:

- ▶ Temperature variation

The most critical factor that affects Infoberry quality is the variation in temperature in the 18 hours before harvest. Ideally, the 18-hour temperature should be as close to constant as possible.

- ▶ Relative Humidity

Another major factor that affects Infoberry quality is the relative humidity at the time of harvest. If it is too dry, the skin of the berry might thicken to an undesirable level. If it is too humid, the berry might over-ripen in transit. The ideal humidity level is 45%.

- ▶ Average Temperature

A less important factor is the average temperature during the 18 hours before harvest. While not as important as the temperature variation, an 18-hour average temperature around 65 F degrees is best for Infoberries.

The factors that affect Infoberry quality are all related to the weather. Two are values that are based on the recent temperature history, which are temperature variation and average temperature, while the other value, relative humidity, is a current reading. Thus, the latest conditions and the recent history of conditions both must be considered when Infoberry quality is estimated.

In addition to buying high-quality Infoberries, some considerations must be made to manage risk that is related to the Infoberry supply. In the past, shipping problems hindered juice production. An investigation into these problems provided the following conclusions:

- ▶ Non-weather-related problems at a supplier sometimes adversely affected shipments. In these cases, we were not aware of precisely what problems a particular supplier experienced, but they resulted in delayed or missed shipments.
- ▶ Severe weather at the supplier location often was the cause of delayed or canceled shipments.

When Infoberries are purchased, we must minimize the risk of unanticipated problems at one supplier that can significantly affect on our supply. We also must avoid ordering from suppliers at locations where severe weather is predicted. By carefully choosing which suppliers to buy from, we can minimize our exposure to supply problems.

## High-level application design

The CommodityPurchasing application uses data sources from flat files and a weather feed website that is provided by National Oceanic and Atmospheric Administration (NOAA). A Stored Procedure Language (SPL) application performs some data analysis (which is shown as a yellow box in Figure C-1) and view the data sink (the violet box on the right in Figure C-1) by using browser-based user interface. Figure C-1 shows a high-level application design for the CommodityPurchasing application.

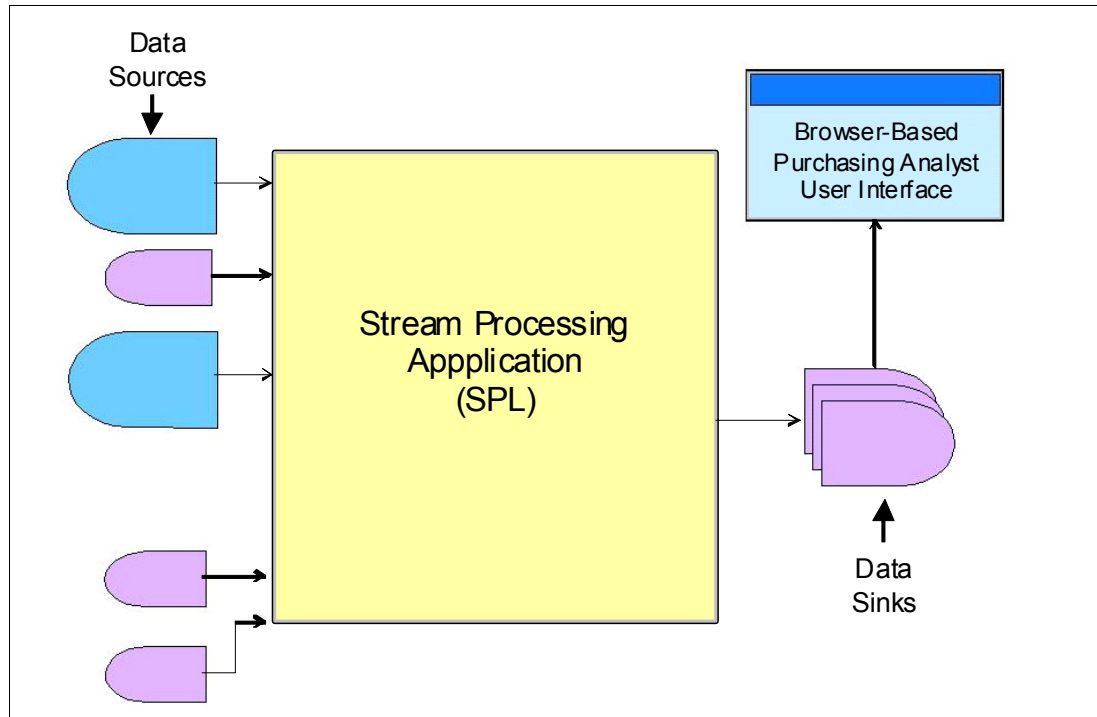


Figure C-1 High-level commodity application design

## Detail application design

This section describes the application design in greater detail. As shown in Figure C-2, the following sources are used for the CommodityPurchasing application:

- ▶ Internet sources are shown in blue
- ▶ Flat file sources are shown in violet
- ▶ Manual input from running a command-line interface is shown in green

File yellow boxes represent separate SPL applications that produce data sinks (which are represented by a violet box on the right) for user interface view.

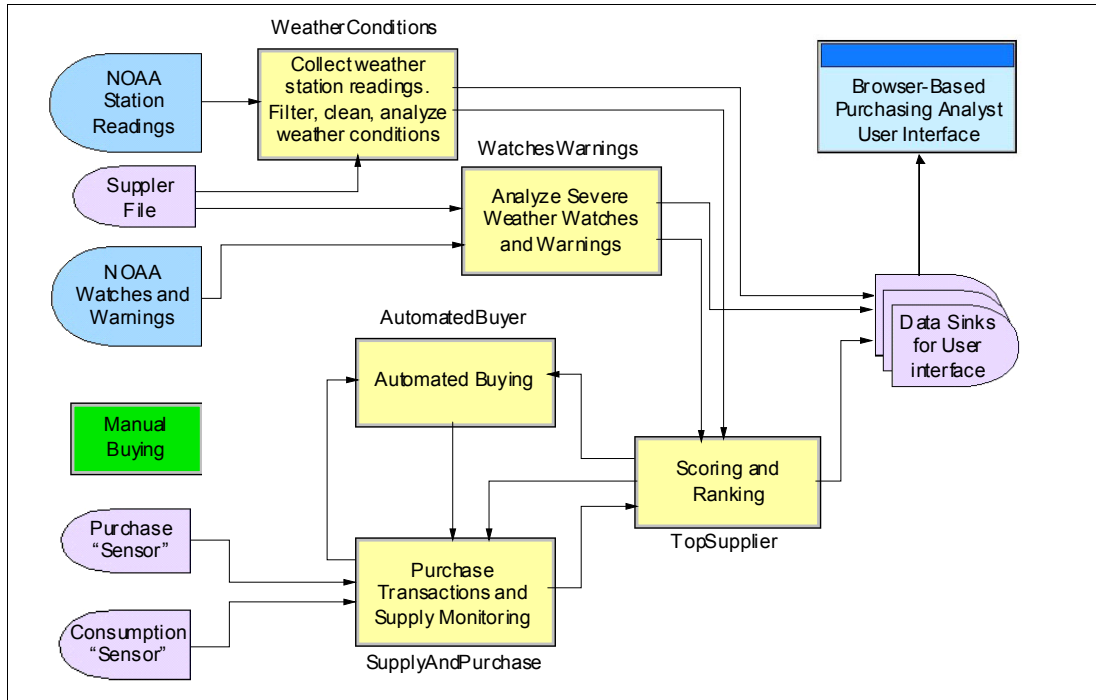


Figure C-2 Detail commodity application design

### AutomatedBuyer

Figure C-3 shows the AutomatedBuyer application operators. Combine the operator import supplier stream that is produced by TopSupplier and the current stock stream that is produced by SupplyAndPurchase into a single stream. The functor then analyzes the data and determines whether an automatic purchase should be made. If the condition meets the criteria to make an automatic purchase, an export operator exports the stream that contains any purchases to be processed by the SupplyAndPurchase application.

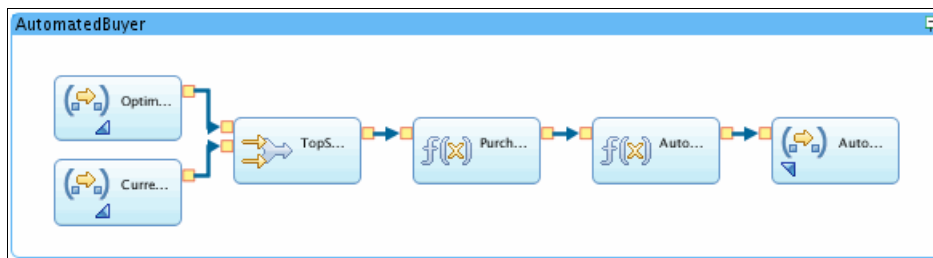


Figure C-3 AutomatedBuyer application

## TopSupplier

The TopSupplier job uses the analysis that was performed by the other jobs to determine the optimal supplier to purchase from at any point. Figure C-4 shows the TopSupplier application on Streams Studio.

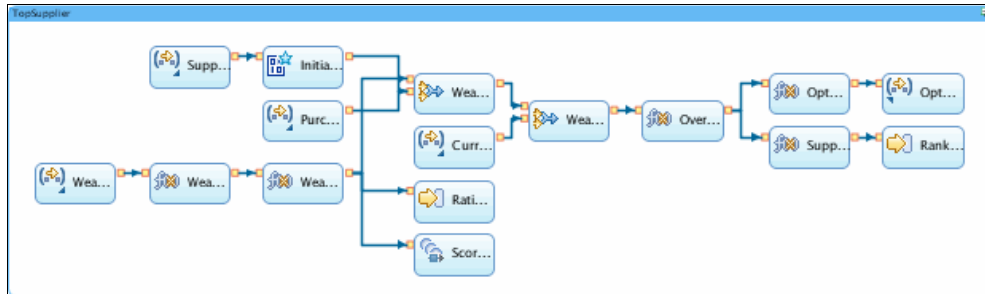


Figure C-4 TopSupplier application

The following data streams come to the TopSupplier application:

- ▶ The SupplierData stream, which is produced by the WeatherConditions job.
- ▶ The WeatherSummary stream, which is produced by the WeatherConditions job.
- ▶ The PurchaseTransactions stream, which is produced by the SupplyAndPurchase job.
- ▶ The CurrentAlerts stream, which is produced by the WatchesWarnings job.

These attributes determine the scoring and ranking of suppliers before the purchase is made.

## WeatherConditions

The WeatherConditions application monitors the 24 URLs from NOAA that report weather conditions from weather stations throughout the world. The WeatherConditions application maps location data from NOAA with the suppliers' location, which is defined on a flat file. The output of this application is the suppliers' location that is viewed on the user interface and the WeatherSummary stream that is used for scoring and ranking on the TopSupplier application. The WeatherConditions application design is shown in Figure C-5.

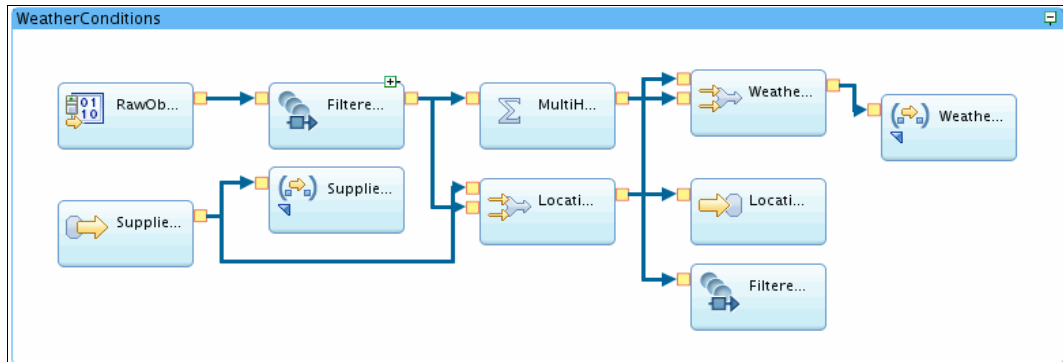


Figure C-5 WeatherConditions application

## SupplyAndPurchase

The SupplyAndPurchase application is responsible for the following tasks:

- ▶ Tracks the number of bushels of Infoberries in Projected Stock at any time.
- ▶ Monitors a consumption sensor that records the rate at which our factory is using Infoberries.
- ▶ Monitors the UI that facilitates user-initiated purchases, and feeds (as an addition) that information to the function that tracks the projected supply and to the function that matches purchases with whichever supplier was top-ranked at time of purchase.
- ▶ Maintains a running average of supplier scores that were used in any type of purchase.

Figure C-6 shows SupplyAndPurchase application in Streams Studio.

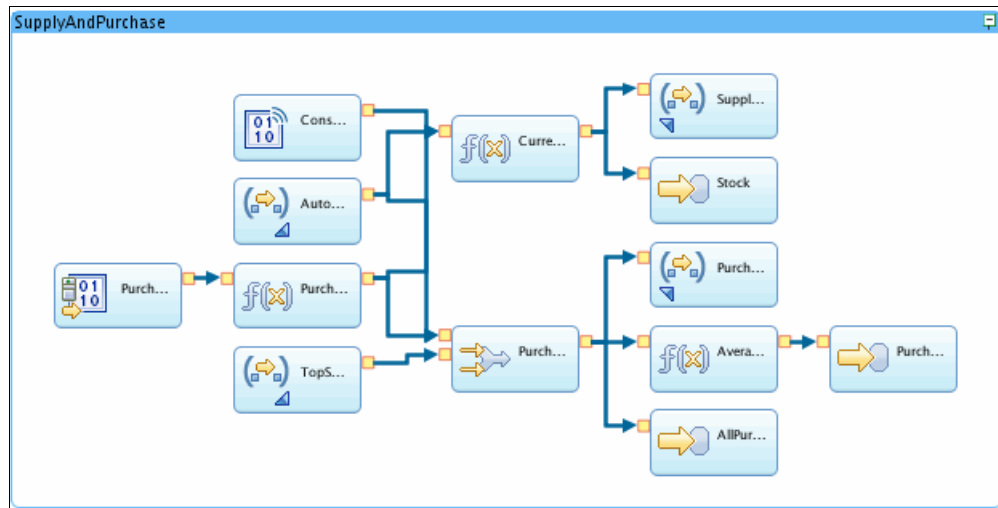


Figure C-6 SupplyAndPurchase application

## WatchesWarnings

The WatchesWarnings job is designed to use NOAA RSS feeds that contain watches, warnings, and advisories for the US. The incoming feeds are scanned to determine whether the county of any supplier of Infoberries is affected by a severe weather alert. If a county of one of our suppliers is affected by a weather alert, our intention is to avoid purchasing Infoberries from that supplier until the weather alert is no longer in effect. Figure C-7 shows the WatchesWarnings application flow in Streams Studio.

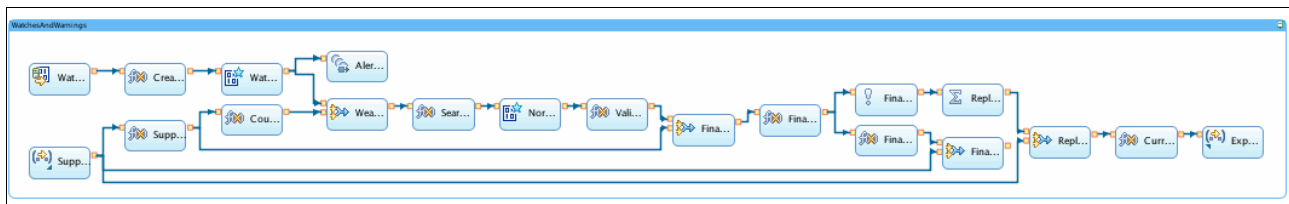


Figure C-7 WatchesWarnings application



## Application demonstration

Complete the following steps to import the application:

1. Select **File** → **Import** in Streams Studio.

The Import windows opens, as shown in Figure C-8.

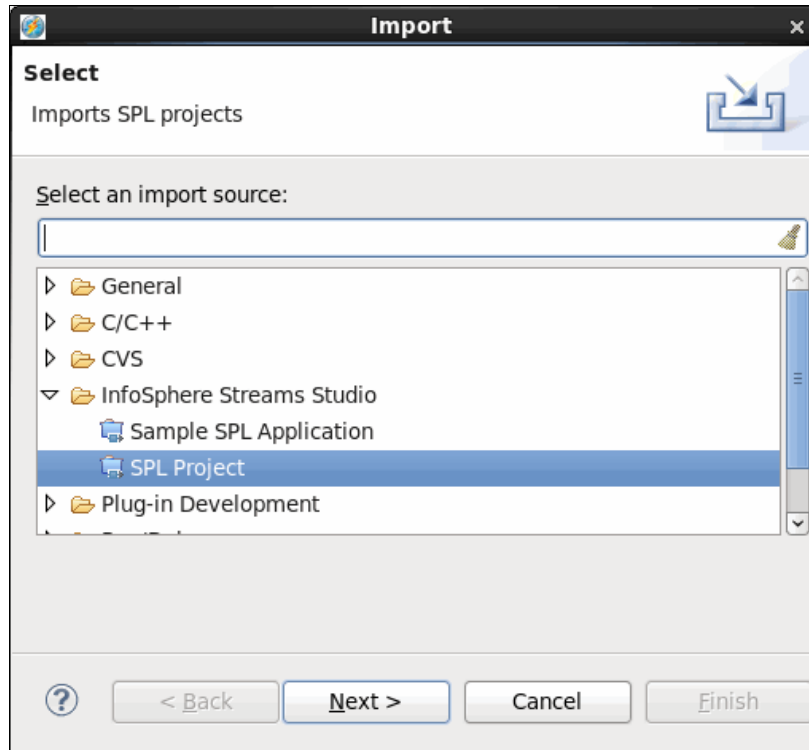


Figure C-8 Import SPL Projects Import window

2. Select **SPL Project** and then click **Next**.

3. Browse the CommodityPurchasing project on your Streams installation directory: \$STREAMS\_INSTALL/samples/sp1/demo/CommodityPurchasing, as shown in Figure C-9. Click **Finish**.

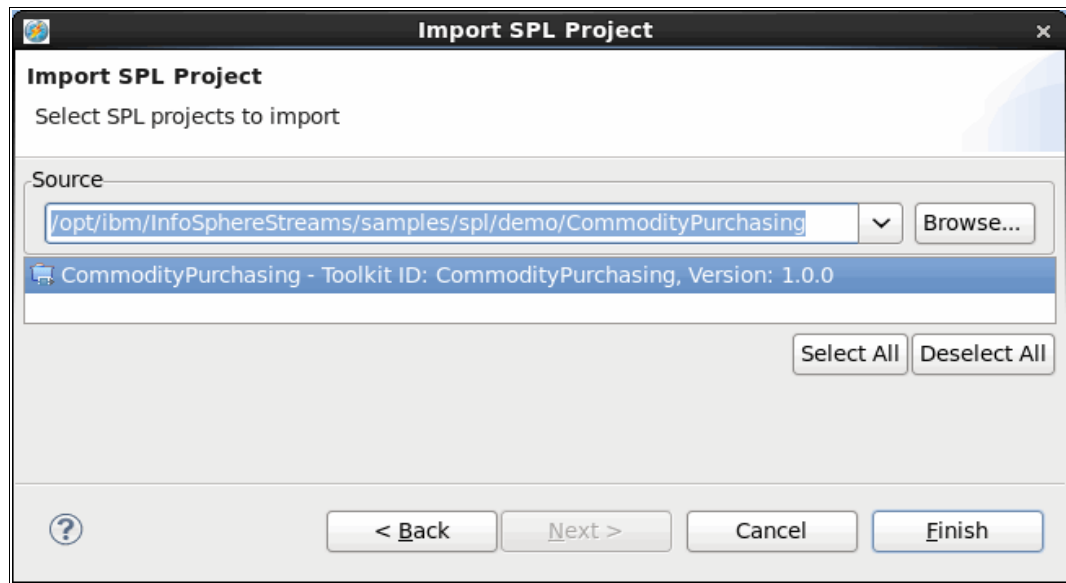


Figure C-9 Import CommodityPurchasing application

4. You find CommodityPurchasing application on your project explorer window, as shown in Figure C-10.

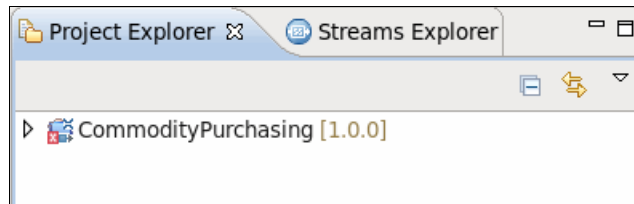


Figure C-10 CommodityPurchasing application on Project Explorer

## Adding missing toolkit

After the CommodityPurchasing application is imported, and error still exists in your project explorer. To fix the error, you must add a missing toolkit on this application.

Complete the following steps to add the Internet toolkit to the application:

1. Click the **Streams Explorer** tab. Right-click **Toolkit Locations**, click **Add Toolkit Location**. Figure C-11 shows the Streams Explorer tab view.

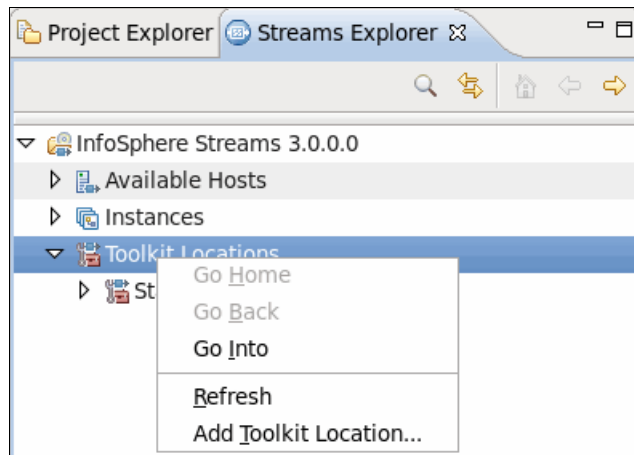


Figure C-11 Streams Explorer tab

2. The **Add toolkit location** (as shown in Figure C-12) window opens. Click **Directory**.

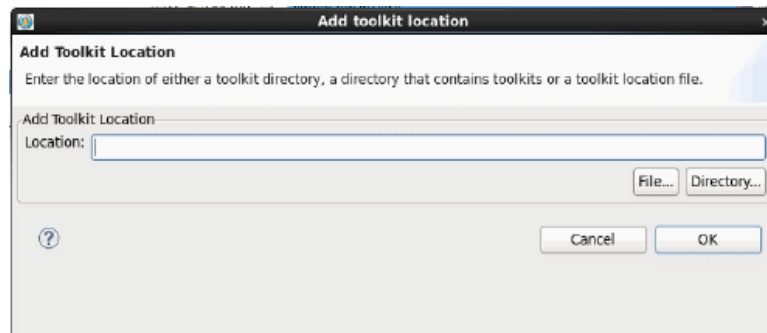


Figure C-12 Add toolkit location window

3. Browse the Internet toolkit on your \$STREAMS\_INSTALL/toolkits directory. Select **com.ibm.streams.inet** in the Select the Toolkit location window, as shown in Figure C-13. Click **OK**.

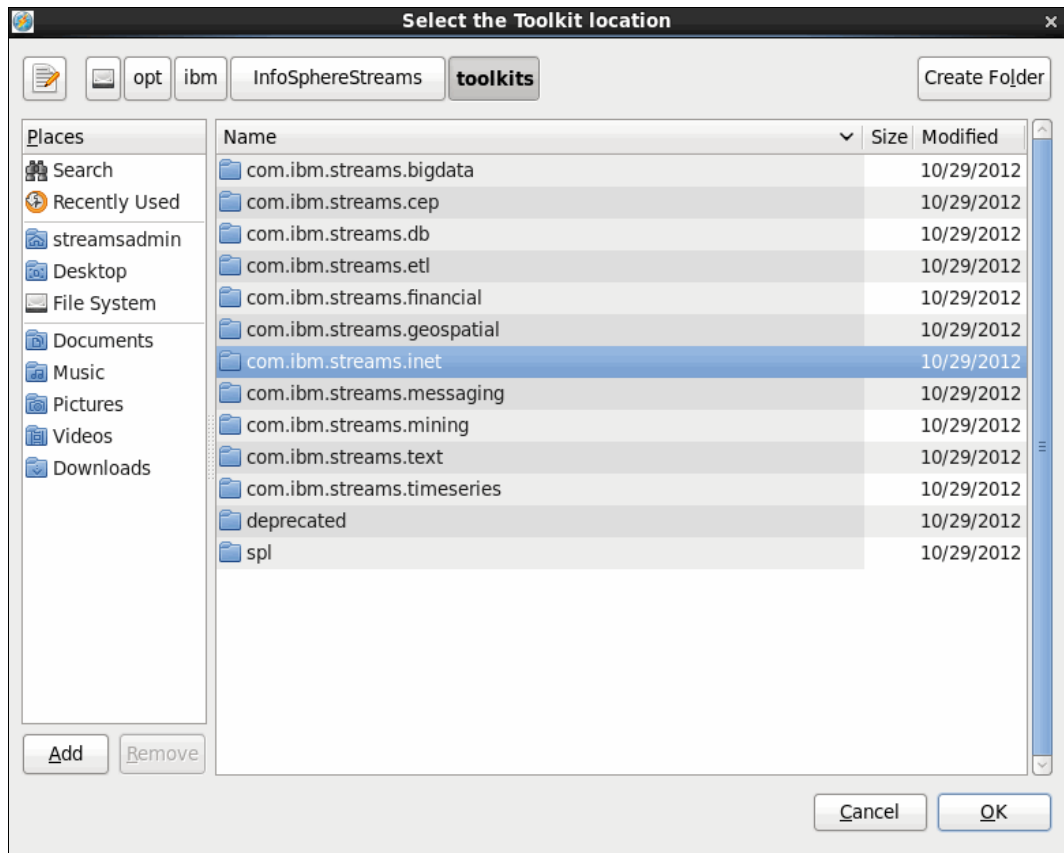


Figure C-13 Select toolkit location window

4. Click **OK** in the Add toolkit location window, as shown in Figure C-14).

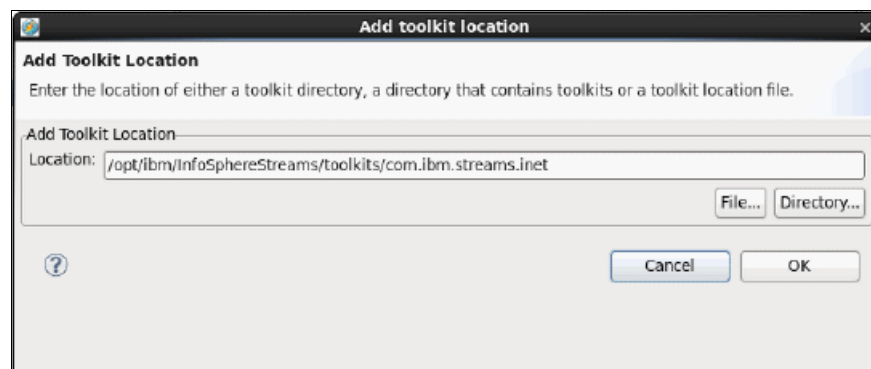


Figure C-14 Add toolkit location window

5. Inspect the Project Explorer tab (shown in Figure C-15). The toolkit dependencies problem is fixed.

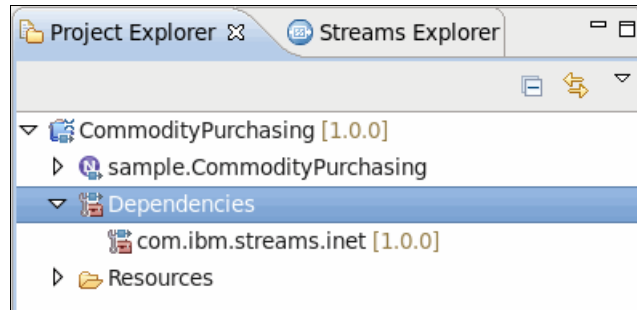


Figure C-15 Project Explorer tab

## Running CommodityPurchasing application

Before an application is run, complete the following steps:

1. Start your instance to deploy the CommodityPurchasing application. For more information about managing your instance by using Streams Studio, see “Instances folder” on page 111. If you want to use Streams console, see 8.1, “InfoSphere Streams Instance Management” on page 200.
2. Install xterm into your Red Hat environment.
3. Confirm that there is an Internet connection on your environment because the CommodityPurchasing application uses live data that is obtained online.

Complete the following steps to run the application:

1. To run a terminal, change the directory to your CommodityPurchasing application inside your Streams Studio workspace directory. Run the **startApp.sh** command, as shown in Figure C-16.

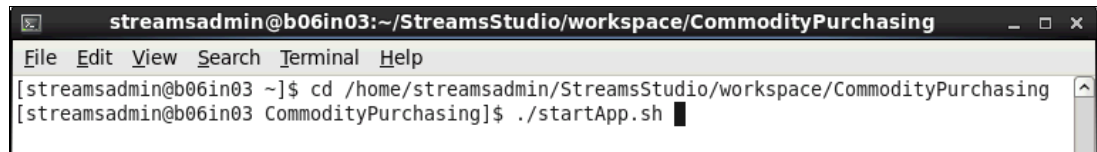


Figure C-16 Running CommodityPurchasing application

Windows appear as shown on Figure C-17. On the upper right side, there is an xterm User-Initiated Purchasing window that is used for manual buying. On the lower right side, you see the CommodityPurchasing User Interface Helper window. One browser window consists of two tabs, Infoberry Purchasing Analysis and Application Metrics that show you the data that is processed by the application.

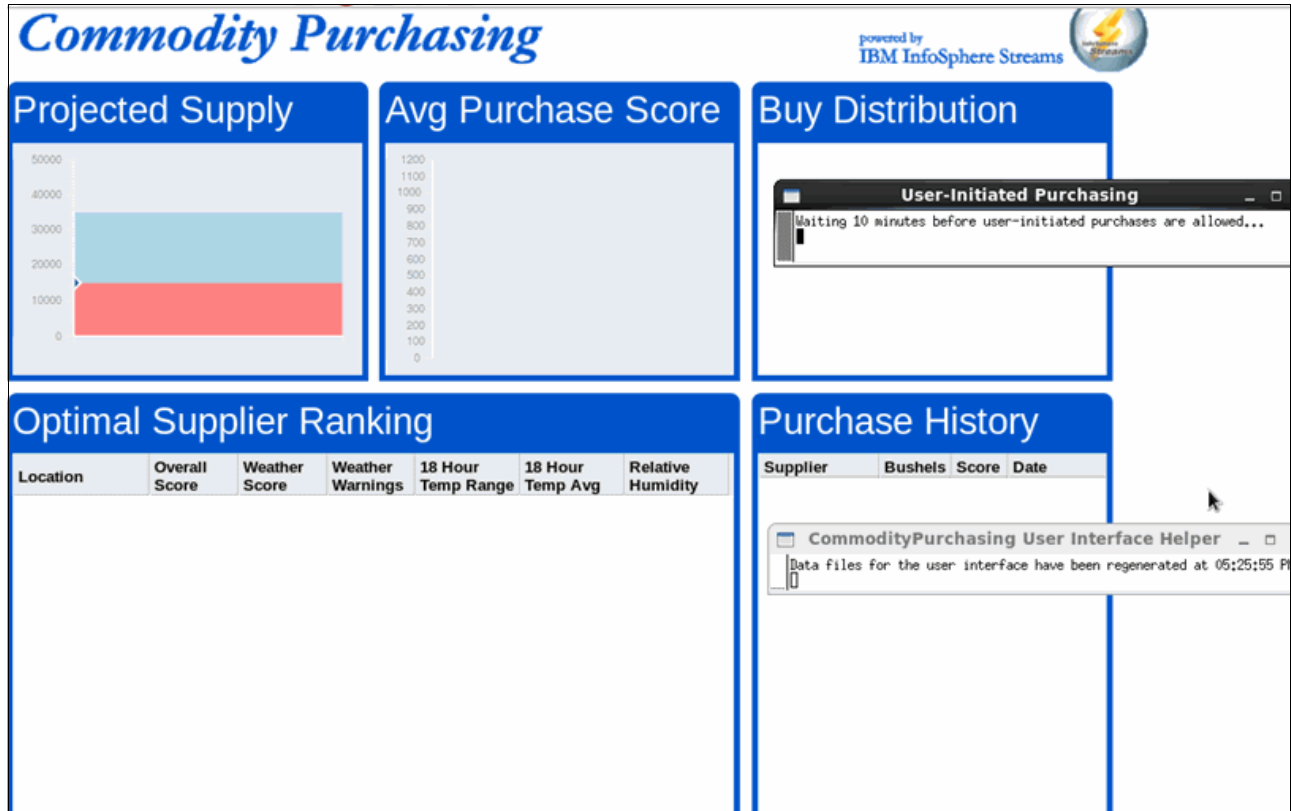


Figure C-17 Running CommodityPurchasing application

## Dashboard

The CommodityPurchasing application dashboard is shown in Figure C-18.

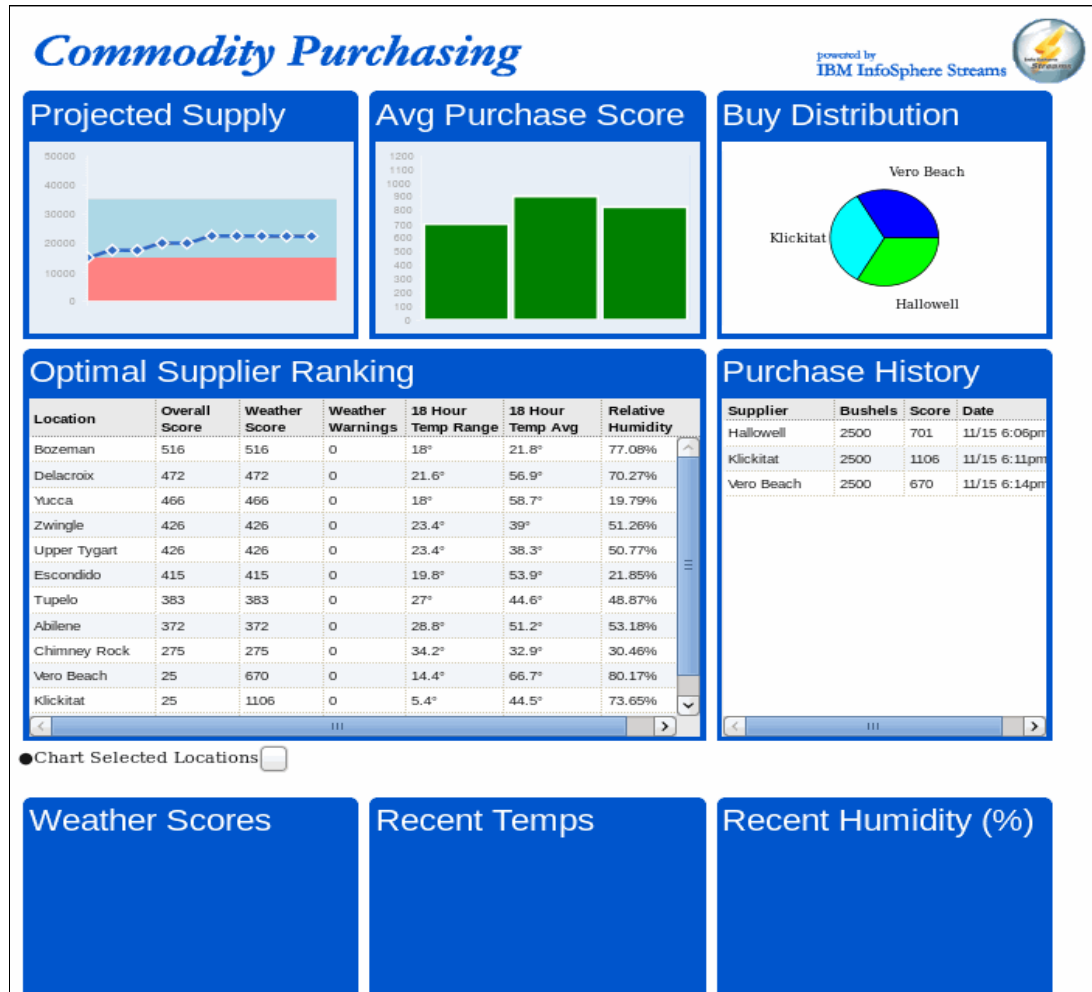


Figure C-18 CommodityPurchasing application dashboard

The dashboard includes the following components:

► **Projected Supply**

This panel carries information about the current Infoberry supply. After there is consumption and buying activity, the updated value is shown on the graph. You can see the detail of each value by hovering to the dot node on the graph and the bottom of the dashboard. There are a few sections of the graph where the background is shaded to indicate the following important supply thresholds:

- The pink region of the graph means that the supply level is dangerously low. Dropping below this level could negatively affect juice production.
- The blue region of the graph indicates supply levels are where we should buy more Infoberreries. The need for more supply is not urgent, but ideally the supply levels should be kept close to, or above, the top of the blue region.
- The white (or light blue) region of the graph indicates a good supply of Infoberreries.
- The top of the graph indicates a supply level that should not be exceeded under normal circumstances. The projected Infoberry supply should never exceed 50,000 bushels.

► Optimal Supplier Ranking

This area shows the ranks of each supplier according to an overall score that is affected by some factors as described in “Application overview” on page 284. The top-ranked supplier is the best supplier to purchase from at any time. When purchases are made, they are automatically selected from the top-ranked supplier. In addition to the overall score, the table shows the following values that are associated with each supplier:

- Weather Score rates each supplier in terms of the weather conditions and their affect on Infoberry quality.
- Weather Warnings indicates the number of severe weather watches and warnings at the time for the supplier location.
- 18-Hour Temp Range shows the magnitude of the temperature variation over the last 18 hours in degrees Fahrenheit.
- 18-Hour Temp Avg shows the average temperature over the last 18 hours in degrees Fahrenheit.
- Relative Humidity shows the current relative humidity.

► Avg Purchase Score

This bar chart shows the cumulative average of the weather score when a purchase is made. This chart is an indicator of how successful the purchases were at obtaining high-quality Infoberries.

► Buy Distribution

This pie chart shows the distribution of purchases across the supplier locations. Over time, the chart shows the suppliers that we buy from the most often, and the suppliers that we rarely buy from. You can fly over any portion of the pie chart to see the details for a particular supplier.

► Purchase History

This table shows a listing of the purchase history. The last 40 purchase transactions are shown.

► Weather Scores, Recent Temps, and Recent Humidity

These tabs are a set of recent history graphs. To view data in these graphs, select one or more suppliers from the Optimal Supplier Ranking table, and click the **Chart Selected Locations** below the table. You can mouse over the points in the graphs to view the details of the history.



## Automatic purchase

This section describes a scenario in which the CommodityApplication performs an automatic purchase. The Projected Supply panel that is shown in Figure C-19 shows that at first, the supply includes approximately 15,000 bushels. After several minutes, any potential supplier that has an overall score of more than 700 is detected. The application then issues the automatic purchase for Hollowell supplier because, at that time, Hollowell has the highest overall score.

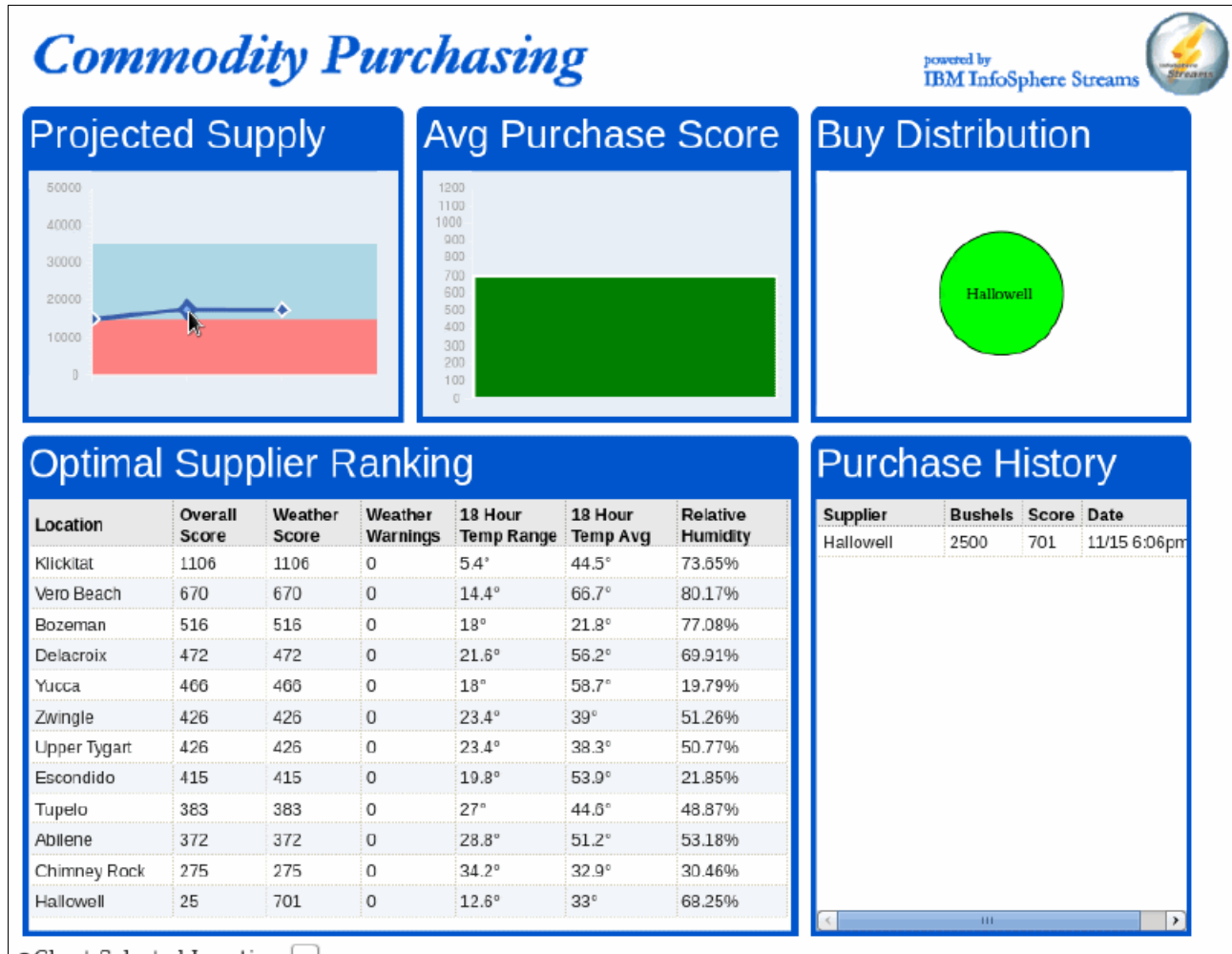


Figure C-19 Automatic purchase

To avoid purchasing from the same place continuously, the application sets the overall score of the purchased supplier to 25 for at least 12 hours. You can see on the Optimal Supplier Ranking panel in Figure C-19 that the Hollowell ranking is placed at the bottom of the supplier with an overall score set to 25. The Avg Purchase Score, Buy Distribution, and Purchase History panels were updated immediately after the automatic purchase.

## Manual purchase

Instead of waiting for an automatic purchase to be made by the application, you also can manually make a purchase. After you start the application, you see the xterm User-Initiated Purchasing window, as shown in Figure C-20. You must wait for 10 minutes before you can manually make a purchase because the application takes some time to identify the optimal supplier.

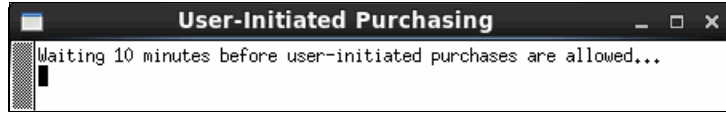


Figure C-20 User-Initiated Purchasing

After 10 minutes, the xterm should resemble Figure C-21. Now you can complete the manual purchase by pressing Enter.

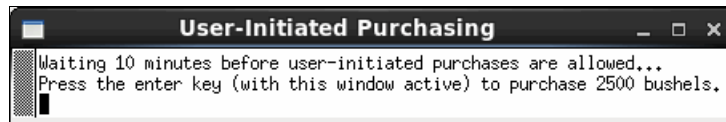


Figure C-21 User-Initiated Purchasing that allowed

Figure C-22 shows you the CommodityPurchasing application dashboard before you complete the manual purchase. The highest overall score in the Optimal Supplier Ranking panel goes to the Vero Beach supplier.

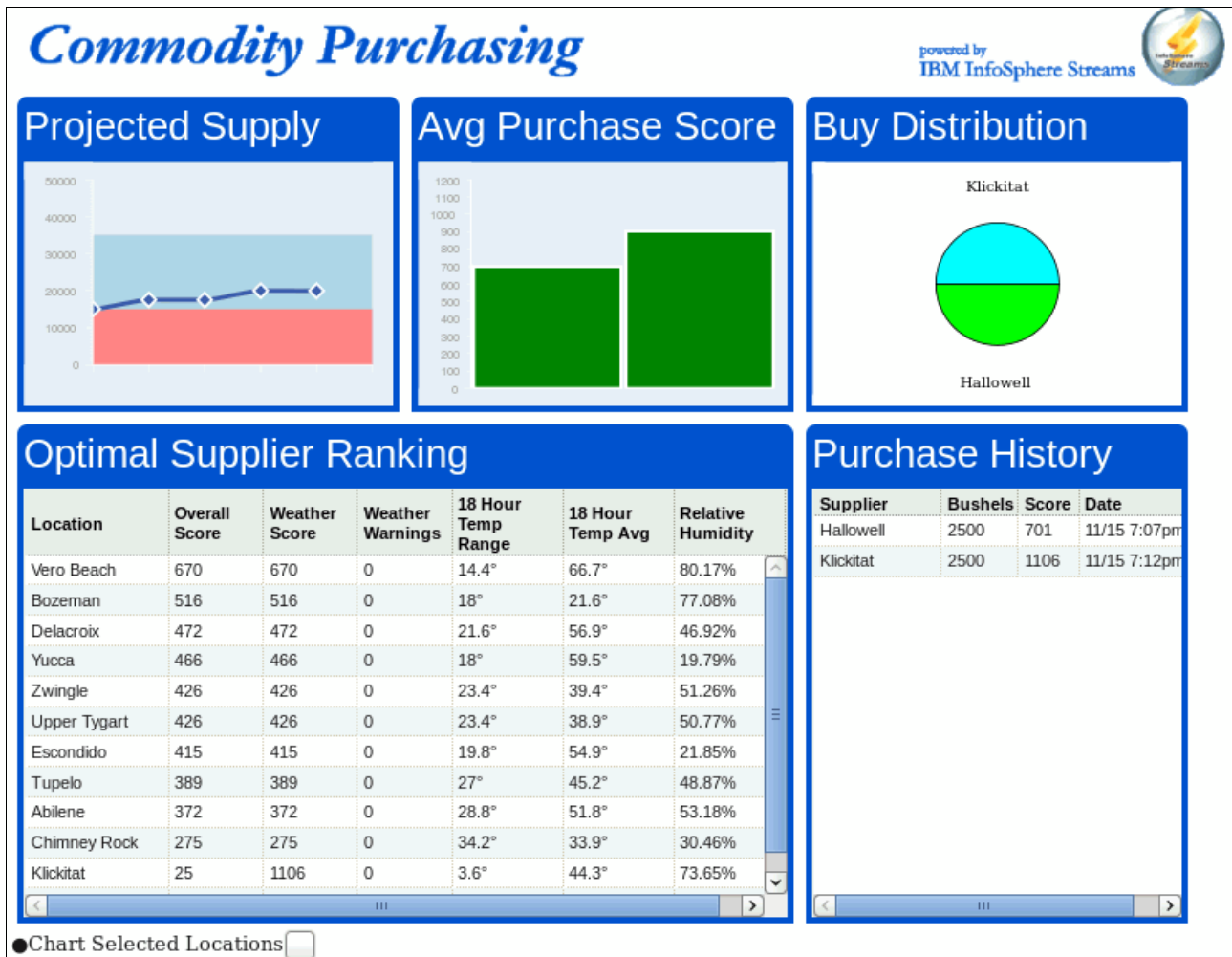


Figure C-22 CommodityPurchasing dashboard before manual purchase is issued

In the User-Initiated Purchasing window, press Enter, as shown in Figure C-23.

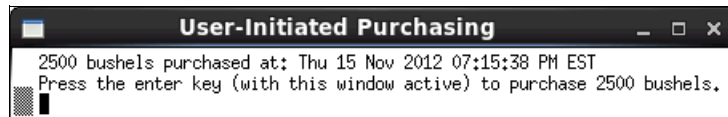


Figure C-23 User-Initiated Purchasing window after manual purchase

Review the CommodityPurchasing application dashboard that is shown in Figure C-24. Vero Beach already is in the Purchase History and Buy Distribution panel and its overall score that is shown in the Optimal Supplier Ranking panel was reduced to 25 for at least 12 hours. Also, review the Project Supply graphics, which shows an increasing number of bushels.

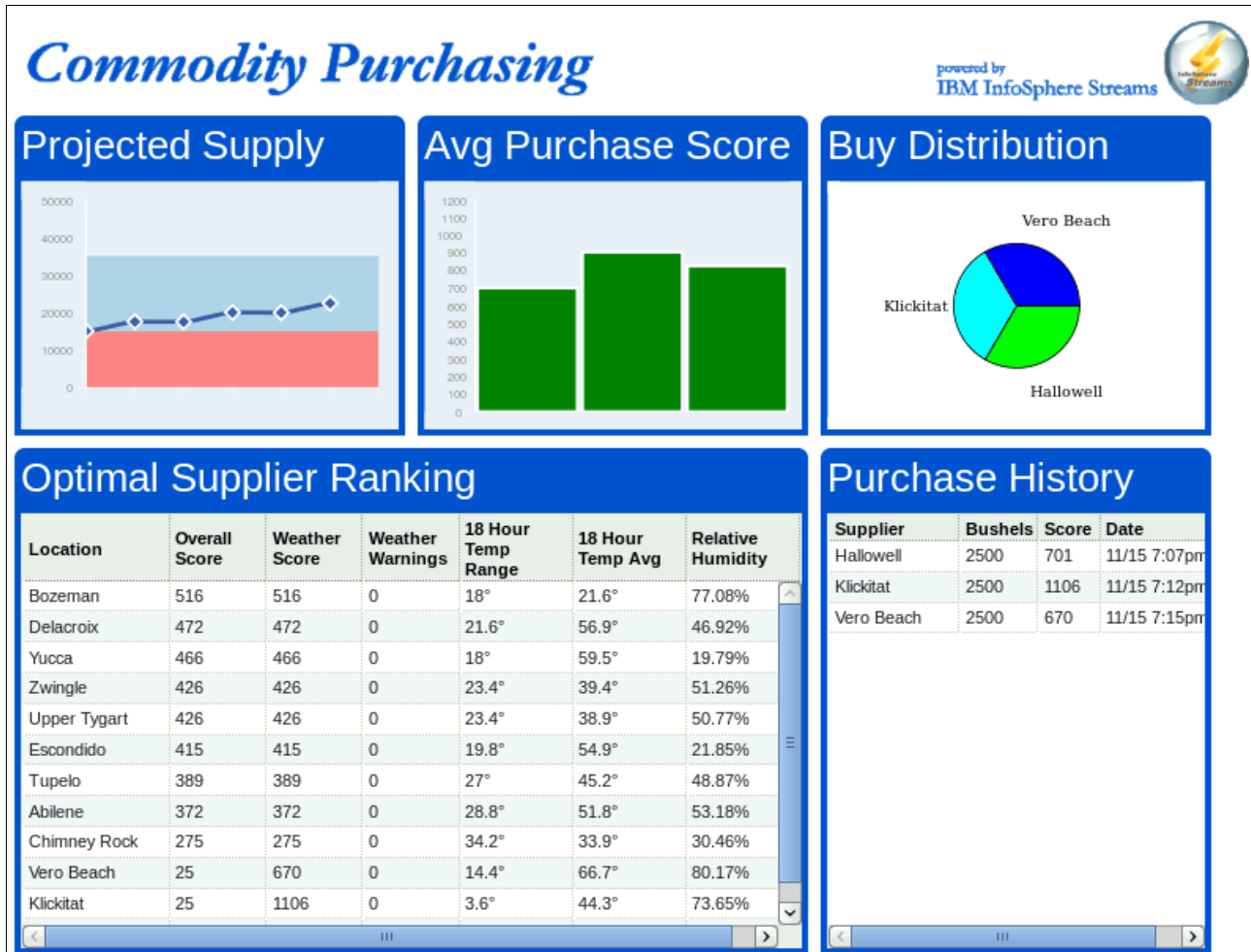


Figure C-24 CommodityPurchasing dashboard after manual purchase is issued

## Supplier identification

Now we want to show why some areas feature a high overall score. You can see in Figure C-25 that Klickitat has a high overall score in the Purchase History panel. Examine the recent history graph. Recall from section “Application overview” on page 284, that constancy of temperature over the past 18 hours is the largest factor in the score. The Recent Temps graph shows that the temperature barely changed over the period that is shown in the graph. You can also see the 18 Hour Temp Range in the Optimal Supplier Ranking, which is only a 5.4 degree movement.

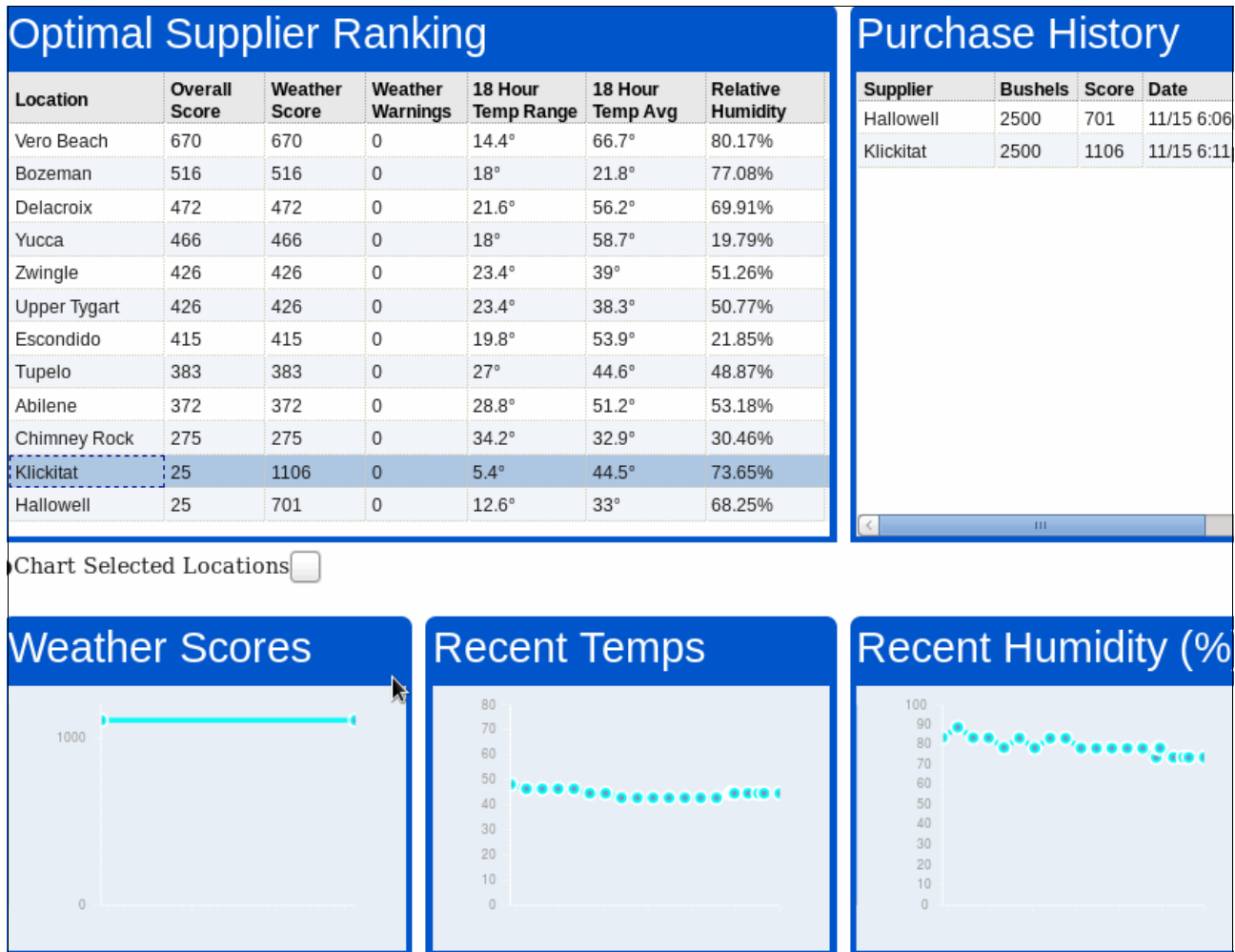


Figure C-25 Recent history graph

## Monitoring the application

For more information about how to monitor the Streams application by using Streams Studio, see 6.2.5, “Monitor your application” on page 131.

For more information about monitoring your application by using Streams Console, see Chapter 8, “IBM InfoSphere Streams administration” on page 199.



# Glossary

**access control list (ACL)** The list of principals that have explicit permission (to publish, to subscribe to, and to request persistent delivery of a publication message) against a topic in the topic tree. The ACLs define the implementation of topic-based security.

**analytic** An application or capability that analyzes a set of data.

**application programming interface** An interface that is provided by a software product that enables programs to request services.

**asynchronous messaging** A method of communication between programs in which a program places a message on a message queue, then proceeds with its own processing without waiting for a reply to its message.

**configuration** The collection of brokers, their execution groups, the message flows and sets that are assigned to them, and the topics and associated access control specifications.

**data mining** A mode of data analysis that focuses on the discovery of new information, such as unknown facts, data relationships, or data patterns.

**deploy** Make operational the configuration and topology of the broker domain.

**engine** A program that performs a core or essential function for other programs. A database engine performs database functions on behalf of the database user programs.

**instance** A particular realization of a computer process. Relative to database, the realization of a complete database environment.

**metadata** Typically called data (or information) about data. It describes or defines data elements.

**multitasking** Operating system capability that allows multiple tasks to run concurrently, which take turns by using the resources of the computer.

**multithreading** Operating system capability that enables multiple concurrent users to use the same program. This ability reduces the effect of initiating the program multiple times.

**optimization** The capability to enable a process to run and perform in such a way as to maximize performance, minimize resource usage, and minimize the process run response time that is experienced by the user.

**pool** Set of hosts that are used on colocation and exlocation constraints.

**process** An instance of a program that is running in a computer.

**program** A specific set of ordered operations for a computer to perform.

**roll-up** Iterative analysis, exploring facts at a higher level of summarization.

**sample** An SPL application that is made available with the product. It might be part of the product, or made available with a toolkit. A sample is usually simple, and meant to illustrate a single feature or functionality.

**server** A computer program that provides services to other computer programs (and their users) in the same or other computers. However, the computer that a server program runs in is also frequently referred to as a server.

**SPL application** Refers to an SPL Main composite operator. When compiled, a Main composite operator can be executed as a distributed or stand-alone application. An SPL source file can have zero or more Main composites.

**SPL application set project** Refers to a project that references one or more SPL applications or SPL mixed-mode applications that are developed or run together.

**SPL mixed mode source file** A file that contains a mix of Perl and SPL code. When pre-processed, the SPL mixed-mode source file yields an SPL source file. SPL mixed-mode source files always feature a file extension of `.splmm`.

**SPL project** Refers to an SPL project. An SPL project can contain an SPL application, which has an `.spl` file extension; an SPL mixed-mode application, which has an `.splmm` file extension; SPL native functions; SPL primitive operators, and so on.

To use Streams Studio, you must create a new SPL project or import an existing SPL toolkit or application. When you import an existing SPL toolkit or application, Streams Studio creates an SPL project for you.

**SPL source file** A file that contains SPL code. The code in a source file implements SPL functions and composite operators. SPL source files always include a file extension of `.spl`.

**stand-alone application** Refers to a stand-alone executable file that was compiled so that it can be run without a Streams instance.

A stand-alone application runs as an executable file and does not use the Streams run time or job management facilities. This feature can be helpful when you are testing or debugging applications. The SPL compiler generates this type of executable file when you set the Executable type option on the SPL Compiler Options preference page to Standalone application. Applications that run on a Streams instance are called distributed applications.

**stream** An infinite sequence of tuples. Each time an operator invocation receives a tuple on one of its input streams, it fires and produces some number of tuples on its output streams.

**task** The basic unit of programming that an operating system controls. Also see multitasking.

**thread** The placeholder information that is associated with a single use of a program that can handle multiple concurrent users. Also see multithreading.

**toolkit** A set of SPL artifacts, which are organized into a package. The main purpose of a toolkit is to make functions (SPL or native) and operators (primitive or composite) reusable across different applications. A toolkit provides one or more namespaces, which contain the functions and operators that are packaged as part of the toolkit, and makes them available to use in applications that can access the toolkit.

**toolkit model editor** Refers to one of the following editors of toolkit model documents:

- ▶ Information model editor
- ▶ Operator model editor
- ▶ Function model editor

**white box applications** Sample applications that are modular in design with pluggable and replaceable components so that they can be modified and extended. Sample applications made available with the Streams toolkits are written as white box applications for easy extensibility.

**zettabyte** A trillion gigabytes.



# Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this book.

## IBM Redbooks

The following IBM Redbooks publications provide more information about the topics in this book. Some publications that are referenced in this list might be available in softcopy only:

- ▶ *IBM InfoSphere Streams Harnessing Data in Motion*, SG24-7865 (Streams V1)
- ▶ *IBM InfoSphere Streams: Assembling Continuous Insight in the Information Revolution*, SG24-7970 (Streams V2)
- ▶ *Analytics in a Big Data Environment*, REDP-4877
- ▶ *Implementing IBM InfoSphere BigInsights on System x*, SG24-8077

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft, and other materials at the following website:

<http://www.ibm.com/redbooks>

## Other publications

The following publications also are relevant as further information sources:

- ▶ *IBM InfoSphere Streams: Installation and Administration Guide V3.0*, SC19-3888
- ▶ *IBM InfoSphere Streams: Studio Installation and User's Guide V3.0*, SC19-3889
- ▶ *IBM InfoSphere Streams: Getting Started with the Streams Trial V3.0*, SC19-3890

## Online resources

The following websites also are relevant as further information sources:

- ▶ Bringing Big Data to the Enterprise  
<http://www-01.ibm.com/software/data/bigdata/>
- ▶ developerWorks Big Data site  
<http://www.ibm.com/developerworks/data/products/bigdata/index.html/>
- ▶ InfoSphere Streams  
<http://www-01.ibm.com/software/data/infosphere/streams/>
- ▶ developerWorks Streams site  
<https://www.ibm.com/developerworks/mydeveloperworks/wikis/home/wiki/Informix%20smart%20meter%20central/page/InfoSphere%20Streams?lang=en>

## Help from IBM

IBM Support and downloads

<http://www.ibm.com/support>

IBM Global Services

<http://www.ibm.com/services>

## Addressing Data Volume, Velocity, and Variety with IBM InfoSphere Streams V3.0

(0.2"spine)  
0.17"->0.473"  
90->249 pages







# Addressing Data Volume, Velocity, and Variety with IBM InfoSphere Streams V3.0



**Performing real-time analysis on big data by using System x**

**Developing a drag and drop Streams application**

**Monitoring and administering Streams**

There are multiple uses for big data in every industry, from analyzing larger volumes of data than was previously possible to driving more precise answers, to analyzing data at rest and data in motion to capture opportunities that were previously lost. A big data platform enables your organization to tackle complex problems that previously could not be solved by using traditional infrastructure.

As the amount of data that is available to enterprises and other organizations dramatically increases, more companies are looking to turn this data into actionable information and intelligence in real time. Addressing these requirements requires applications that can analyze potentially enormous volumes and varieties of continuous data streams to provide decision makers with critical information almost instantaneously.

IBM InfoSphere Streams provides a development platform and runtime environment where you can develop applications that ingest, filter, analyze, and correlate potentially massive volumes of continuous data streams based on defined, proven, and analytical rules that alert you to take appropriate action, all within an appropriate time frame for your organization.

This IBM Redbooks publication is written for decision-makers, consultants, IT architects, and IT professionals who are implementing a solution with IBM InfoSphere Streams.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:  
[ibm.com/redbooks](http://ibm.com/redbooks)**

SG24-8108-00

ISBN 0738437808