



CORAL8, INC.

Comprehensive Guide to Evaluating Event Stream Processing Engines



Copyright © 2006 Coral8, Inc. All rights reserved worldwide.

Worldwide Headquarters:

Coral8, Inc.
82 Pioneer Way, Suite 106
Mountain View, CA 94041
Phone: 650-210-3810
Fax: 650-210-3811

Visit us on the web at www.coral8.com or e-mail us at info@coral8.com.

Table of Contents

Introduction: Evaluating ESP Engines	1
1 What Products to Evaluate	2
1.1 ESP or CEP?	2
1.2 An Application or an Engine?.....	2
1.3 So an ESP Engine is...?	2
2 Programming Model: Basics Questions	3
2.1 Does the Programming Model Look Familiar?.....	3
2.2 Is the Primary Programming Model Language-Based or GUI-based?.....	3
2.3 Are Continuous Incremental Queries Supported?	3
2.4 Are Windows Supported? What Kind?	3
2.5 Are There Rich Features For Joining or Correlating Multiple Streams?.....	4
2.6 Is Interfacing with Stored Data Supported?.....	4
2.7 Are Event Patterns Supported?.....	4
3 Programming Model: Advanced Questions	5
3.1 Are XML Messages Supported?	5
3.2 Are User-Defined Functions, Aggregators, and Datatypes Supported?	5
3.3 How Are Complex Applications Built?.....	5
3.4 What Testing and Debugging Capabilities Are Available?	5
4 Ease of Integration and Deployment Questions	6
4.1 How Does Data Get Into the Engine?.....	6
4.2 How Are Data Anomalies Handled?	6
4.3 How Does the Engine Produce Output (Data, Alerts, and Actions)?	6
4.4 Is There a Publish/Subscribe Transport Layer Included with the Engine?	6
4.5 How Do Queries Get Registered With the Engine?	7
4.6 Do I Need to Restart the Engine to Accomplish...?	7
4.7 How Easy is the Engine to Install?	7
4.8 How Modular is the Engine's Design?.....	7
4.9 What Platforms Are Supported?.....	7
5 Enterprise Features Questions	8
5.1 Is There Support for State Persistence?	8
5.2 Is There Support for Clustering for High Availability?	8
5.3 Is There Support for Clustering for High Performance?.....	8
5.4 What Security Features Are Available?.....	8
5.5 What Monitoring and Management Capabilities are Available?	9
6 Performance Questions	10
6.1 What Kind of Event Processing Latency Does the Engine Exhibit?	10
6.2 What is the Throughput of the Engine?	10
6.3 How Many Streams and Queries Can the Engine Handle?	10
6.4 How is Performance Affected by Persistence, High Availability, Guaranteed Deliver, Security, and other Options?	10
7 Conclusion: Start a Hands-On Evaluation!	11

Introduction: Evaluating ESP Engines

This purpose of this white paper is to assist prospective customers who are planning to evaluate one or more Event Stream Processing Engines, also known as Complex Event Processing Engines. Even though this paper is produced by a vendor (Coral8, Inc.), every effort has been made to keep the material objective and vendor-neutral. The word “Coral8” does not appear anywhere in the body of this document, except in this paragraph. Coral8, Inc. welcomes feedback and comments from customers, partners, and other vendors in this space.

This white paper poses a set of questions that somebody evaluating Event Stream Processing Engines may wish to answer. One way to answer these questions is to ask the vendors, and we encourage all prospective customers to do so. Asking the vendors is easy, but we strongly believe it is more productive to arrive at the answers through **hands-on evaluations**. These evaluations are by far the best way to learn if the Engine meets the project requirements, is easy to use, and exhibits the right level of performance in real-world applications from a particular domain.

While some ESP Engines can be downloaded directly from the vendor’s web site, others may require you to contact a vendor’s sales representative. Regardless, most vendors are very happy to assist customers with product evaluations, and many will work with customers to build one or more application prototypes. Building a prototype is the most reliable way to answer the questions raised in this document.

The rest of the document consists of the following sections:

- **What to Evaluate** suggests how to choose the right products to evaluate.
- **Programming Model (Basic Questions)** contains questions about basic engine features available to developers.
- **Programming Model (Advanced Questions)** covers some advanced features necessary for building complex applications.
- **Ease of Integration and Deployment Questions** discusses the features that make an ESP Engine easy to integrate and deploy in production environments.
- **Enterprise Features Questions** discusses questions about clustering, high-availability, persistence, security, and other features required for large distributed deployments of ESP Engines in enterprises.
- **Performance Questions** covers performance questions, and explains why performance questions are typically more complex than first suspected.

1 What Products to Evaluate

1.1 ESP or CEP?

Choosing the right products to include in the evaluation is half the job, but the process can be very confusing. The best places to start are web sites that list the vendors and the products, such as www.complexevents.com and www.eventstreamprocessing.com. However, many of the products on these web sites are quite different, and comparing some of them to each other is not unlike comparing apples to oranges. Part of the problem is the lack of common agreed-upon terminology. For example, some vendors speak of their products as solving Event Stream Processing (ESP) problems, while others talk of Complex Event Processing (CEP). For the purpose of this document, however, we consider the two to be **one and the same**. To most customers, what the products do or do not do is much more important than the moniker used to describe them.

1.2 An Application or an Engine?

Another more serious confusion is that some products on the market are Event Stream Processing **applications** for solving *specific* problems, or classes of problems, such as program trading, risk management, network monitoring, intrusion detection, click stream analysis, etc. Business people, IT administrators, or other non-developer users typically use these applications or solutions to solve specific problems. These products are **not** dealt with here.

Products covered by this white paper are **general purpose** Event Stream Processing Engines (or platforms) used for **building** a wide variety of Event Stream Processing applications. These Engines are typically used by IT developers, Independent Software Vendors and System Integrators. It is important not to confuse these Engines with Event Stream Processing applications.

1.3 So an ESP Engine is...?

It is not the purpose of this document to define precisely what an Event Stream Processing Engine is or is not. The products on the market vary, and no one approach is clearly the best for all situations. Therefore, this document will adopt a simple high-level definition:

An Event Stream Processing Engine is a software product used to create and deploy applications that process large volumes of incoming messages or events, analyze those messages or events in various ways, and respond to conditions of interest in real-time.

2 Programming Model: Basics Questions

The programming model describes how developers build their applications. It includes the set of abstractions, language features, and APIs that developers need to learn to use an ESP Engine. Some of the basic programming model questions are discussed in this section.

2.1 Does the Programming Model Look Familiar?

This is often the single most important question developers want answered. One alternative to an ESP Engine is custom coding an application using a language such as C, C++, Java, C#, or VB.NET, probably coupled with the use of a database (SQL). These technologies are very familiar to most developers. If the ESP Engine's programming model does not look familiar and does not appear very easy to learn, developers will resist using it.

The best way to evaluate the programming model is to look at as many examples of queries and sample applications as possible, and to create some examples of your own. When reviewing the answers to other questions, it is a good idea to ask, "Can I see an example?"

2.2 Is the Primary Programming Model Language-Based or GUI-based?

This is an area where products may differ greatly. Some products are primarily controlled via a programming language, whether an SQL-like language or a different language, while other products are controlled via a "drag-and-drop" graphical user interface (GUI). Some products provide both, but one approach usually predominates. Since many developers traditionally have very strong feelings on language-based vs. drag-and-drop programming environments, it is important to clearly understand which options are supported by a given Engine.

2.3 Are Continuous Incremental Queries Supported?

Continuous incremental queries appear in one form or another in most ESP Engines on the market. In fact, the ability to execute continuous queries clearly separates ESP Engines from other similar technologies, such as in-memory databases. These queries reside in memory, and continuously compute and produce the requested outputs in response to events. One may also want to ask:

- Are incremental aggregators (Count, Sum, Avg, Min, Max, StdDev) supported? Aggregators benefit greatly from incremental processing. For example, to compute Sum over a window, one keeps a running total, and adjusts it as messages enter and leave the window.
- Are incremental User-Defined Aggregators supported?
- Are incremental Joins supported?

2.4 Are Windows Supported? What Kind?

Windows define a subset of data from a data stream. While initial applications may need just a few basic window types, additional types of windows will typically become relevant as applications evolve. The following types of windows are often important:

- Time-based windows (e.g., keep 10 seconds of data)
- Count-based windows (e.g., keep 100 events)
- Count-based windows partitioned by an attribute (e.g., the last 100 trades for each stock)
- Sliding windows (e.g., a window that continually keeps the last 10 seconds of data)
- Jumping (or tumbling) windows (e.g., a window that resets every 10 seconds)
- Top-k / Bottom-k windows (e.g., the top 10 stocks by trade volume)

- Complex windows (e.g. the last 10 readings for each reader that are less than 1 minute old)
- Explicitly-controlled windows (e.g., allows rows to be arbitrarily inserted or deleted)

2.5 Are There Rich Features For Joining or Correlating Multiple Streams?

The power of ESP Engines comes from their ability to join or correlate messages from multiple streams or windows. This is similar to joining multiple tables in a relational database. Here are some kinds of joins or join-related issues to consider:

- Is it possible to join a stream against a window? Against multiple windows?
- Is it possible to join multiple windows against each other?
- Are outer joins supported?
- Are joins with complex join conditions supported? How complex?
- How is indexing done for joins? What kinds of indexes are supported?
- Are joins with stored data supported (see next question)?

2.6 Is Interfacing with Stored Data Supported?

Interfacing with data stored in relational databases and other storage systems is very important, as real-time messages often need to be efficiently stored in a database, and may require processing in the context of stored messages. Here are some questions to ask:

- Is it possible and convenient to write data (both raw data and processed data) into a standard relational database (Oracle, Microsoft SQL Server, MySQL, etc.)?
- Is it possible to read data from a database and join incoming data streams with this data?
- What is the granularity of data access, i.e., is it possible to read a single row of data on demand? For example, if there is a huge table that stores properties for each RFID tag, is it possible to read just one row from the database when the tag appears in a stream, rather than reading the whole table?
- Is caching supported? Without caching, database access may be required for every incoming message resulting in performance limitations. Strong data caching support is of paramount importance if high performance is a requirement.

2.7 Are Event Patterns Supported?

Event patterns arise in many areas applications requiring the matching of a combination or sequence of events. For example, a typical pattern may be "Over 10 minutes, event A happens, then B and C happen in either order, then D or E happen in either order, and then F does not happen, with restrictions and relationships on all the events specified." Supporting event patterns directly is extremely useful in many types of applications, but not all products have this capability. Here are some questions to ask about those that do:

- Are sequence operators supported (A then B)?
- Is AND supported? (A and B in either order)
- Is OR supported? (A or B in either order)
- Are non-events supported? (A does NOT happen). (This is a very important question to ask. Naive implementations trigger actions only when certain events occur. It is also very important to be able to do something when an event does NOT occur.)
- What kind of conditions can be specified for the events in the pattern?

3 Programming Model: Advanced Questions

3.1 Are XML Messages Supported?

XML messages are very important to some applications, less to others, and not important to the rest. If XML messages are important, it is necessary to understand how the ESP Engine treats them. There are several models to consider:

- The Engine itself does not understand anything about XML. An adapter of some sort needs to parse the XML message into a simpler format understood by an Engine
- The Engine has a limited understanding of XML messages. For example, XSLT may be supported for transformations, or XPath can be supported to extract individual fields.
- The Engine may know how to do advanced operations with one or more complex XML messages, similar to what SQL/XML and XQuery can do.

3.2 Are User-Defined Functions, Aggregators, and Datatypes Supported?

No matter how much functionality is provided by the Engine's programming model, sometimes the developer will want to code some logic in a "traditional" programming language. Relevant questions to ask include:

- Are user-defined functions supported?
- What languages can be used for writing user-defined functions?
- Are user-defined aggregators over windows or other stateful functions supported?
- Are custom data types supported?

3.3 How Are Complex Applications Built?

This is an open-ended question, but it may be interesting to know if the Engine supports:

- Combining queries into complex processing topologies, where data flows through multiple layers of queries, perhaps residing on different servers.
- Encapsulating sets of queries into shareable parameterized modules.
- Creating complex state management application logic, such as Finite State Machines.
- Applications where queries and subscriptions can be instantiated dynamically at run time

3.4 What Testing and Debugging Capabilities Are Available?

Event Stream Processing applications, just like other applications, need to be tested and debugged before deployment. Developers who develop applications in traditional language expect a high level of debugging support. In particular, they may expect:

- An ability to quickly switch between live and recorded data feeds for testing and algorithm back-testing.
- An ability to visualize the data flow across streams and queries.
- An ability to easily examine every stream in the system, whether input, output, or intermediate
- An ability to examine the contents of every window, and any other state held by the system.
- An ability to "step-through" through time slices, forward and backward, and observe how the behavior of the system changes.

4 Ease of Integration and Deployment Questions

4.1 How Does Data Get Into the Engine?

Most Engines come with some sort of a framework for creating agents or adapters that transform external data into a format understood by the Engine. Here are some relevant questions:

- What adapters or agents come with the Engine?
- Are there adapters that support common standards in the relevant domain? For example, if your application uses JMS, you should find out if a JMS adapter comes with the Engine.
- How easy is it to create new input adapters? What languages are supported?
- Can an adapter run on the same machine as the Engine? In the same process? Can it run on a different machine?

4.2 How Are Data Anomalies Handled?

Some Engines assume that incoming messages are never delayed, are never out of order and are never lost. In real world, this is not the case. Some very important questions to ask include:

- Does the Engine handle delayed messages?
- Can the Engine synchronize messages across multiple data sources?
- Does the Engine handle out-of-order data?
- Can the Engine process messages according to the message timestamp?
- Can the Engine process messages according to the time of their arrival at the Engine?
- Are there guaranteed delivery options for situations when messages must not get lost?

4.3 How Does the Engine Produce Output (Data, Alerts, and Actions)?

Some ESP Engines provide fewer options on the output side than on the input side. Some make an assumption that the volume of output messages is lower than the volume of input messages, which is sometimes true and sometimes false. Here are some questions to consider:

- What output adapters are supported out of the box? Are there adapters to send an email, issue a SOAP command, store data into a database, publish into a JMS bus, etc.?
- How easy is it to create new output adapters? What languages are supported?
- Are output adapters capable of processing the same volume of data as input adapters?

4.4 Is There a Publish/Subscribe Transport Layer Included with the Engine?

Publish/Subscribe is a very common paradigm for building loosely coupled applications, where multiple publishers and subscriptions can be added dynamically, using a set of subjects. The questions to ask include.

- Is there support for publish/subscribe in the engine?
- Is it built-in, or does it rely on third-party products?
- Is there support for location-transparency, i.e., is it possible to connect to a stream without knowing its exact location?

4.5 How Do Queries Get Registered With the Engine?

Even though ESP Engines support continuous queries, it is very important for many applications to be able to register the queries dynamically, without restarting the Engine or affecting other queries. Some questions to ask are:

- What interfaces are supported for registering queries dynamically? Is SOAP one of them?
- In addition to continuous queries, are request-response queries supported?
- Are parameterized subscription-style queries supported? For example, if a query checks for a certain stock to reach a certain threshold, can a user dynamically instantiate several versions of this query, each with a different stock symbol and threshold?

4.6 Do I Need to Restart the Engine to Accomplish...?

This is a very important question! Some products require you to restart the Engine to accomplish simple tasks, such as:

- Adding a query
- Adding a data source
- Adding a new subscriber to a stream
- Adding a new publisher to a stream
- Adding a new action or alert

If it is important for you to run the Engine continuously, make sure to ask the ESP Engine vendor whether you need to restart the server for any of these reasons.

4.7 How Easy is the Engine to Install?

Ease of installation is a major factor for many applications. Ideally, the Engine should be easy to install and deploy in a number of configurations. Since any serious evaluation begins with product installation, this should be a good indicator of how easy the Engine will be to install and manage in production.

4.8 How Modular is the Engine's Design?

This is important to many customers, especially ISVs who build their own products. Some of the important questions include:

- One may want to use the ESP Engine's runtime, but not the development environment. Are the two easily separable? Is there a well-defined interface between the two?
- Can queries be created and managed via command line tools, or must the Engine's GUI be always used?
- Are input and output adapters clearly separable from the core engine?

4.9 What Platforms Are Supported?

This set of questions is very straightforward:

- What platforms are currently supported by the Engine's runtime and development environment?
- If the platforms you prefer are not currently supported, how hard is it to get a build for your platform?
- Are 64-bit processors supported? Even if you do not need this immediately, you may find that the CPU and memory requirements of your project call for this later.

5 Enterprise Features Questions

Enterprise-level features are required to run an ESP Engine in clustered, mission-critical environments. The features you require depend on the scale you need. A number of important enterprise feature issues are covered in this section.

5.1 Is There Support for State Persistence?

Many customers often think of ESP Engines as in-memory systems, not unlike in-memory databases. This was indeed true of early ESP Engines, but it is no longer true of the best ones. If losing the state of the computation (windows, in-memory tables, aggregators, etc.) is not an option, make sure to ask your ESP vendor whether a state persistence option is available. Some additional questions include:

- Where is the data stored?
- What is the granularity of the persistence option? Can it be enabled for some queries but not others?
- How is the system structured to minimize the performance penalty of the persistence option? What is the performance penalty?

5.2 Is There Support for Clustering for High Availability?

It is often not enough to simply save and restore the state in case of a failure. In mission critical applications, it is necessary to move computation to a new server in case of hardware or software failures. Some questions to ask:

- Is High-Availability supported?
- Is it supported in 1-to-1 Primary-Secondary configuration, or in many-to-many (e.g., can one deploy a cluster with 10 primary servers and 3 secondary)?
- How long does it take the system to detect a failure and to recover from it?

5.3 Is There Support for Clustering for High Performance?

Event Stream Processing applications are often very CPU and memory intensive, so clustering may be needed for high performance. Some questions to ask include:

- Is clustering for performance supported?
- How easy is it to set up?
- Is it easy to split one stream across multiple servers?
- Is it easy to split queries across multiple CPUs? Multiple servers? Multiple layers of servers?
- Is automatic load balancing supported?

5.4 What Security Features Are Available?

Security is important for many applications running in enterprise environments. If security is important to you, the relevant questions include:

- What is the security model supported by the Engine?
- Is transport encryption (SSL) supported?
- Is it possible to limit who can connect to the Engine? What forms of authentication are supported? Is SSL authentication available?

- Is there an access control model for queries, streams, adapters, messages, and other entities?

5.5 What Monitoring and Management Capabilities are Available?

Monitoring and Management is important for every enterprise. Some specific questions to ask include:

- What monitoring capabilities are provided? Can one monitor CPU usage, memory utilization, I/O, and other parameters for each application hosted by an Engine?
- Can one get this information via a GUI? Via command line tools? Via SOAP? Via a stream? Via SNMP? Via any other interfaces?
- What management capabilities are supported? Can one start and stop the server via command line tools? Can one configure the server both via a GUI and via configuration files? What configuration options are present?

6 Performance Questions

Performance is often the first thing customers ask about, and this is easy to understand: everybody is curious whether a general purpose Engine can match the performance of custom-coded applications. Unfortunately, comparing the performance of different Engines can be a complex task. There are many variables, such as the hardware used, the message format and content, and the actual query functions, which require normalization before different Engines can be meaningfully compared. Even then, subtle differences in query semantics may be unavoidable. With this caveat, here are some questions to ask:

6.1 What Kind of Event Processing Latency Does the Engine Exhibit?

Latency measures how long it takes the Engine to process a given event, typically measured in milliseconds or fractions of a millisecond. Specific latency-related questions might include:

- What is the end-to-end latency (end-to-end latency should include message processing time as well as transformations to and from the internal formats)?
- What is the pure processing latency for typical queries?

6.2 What is the Throughput of the Engine?

Throughput is measured in messages per second *for a given query or set of queries*. Speaking about throughput without specifying the queries is meaningless! Some simple types of queries to consider are:

- Pass-through: The query that simply passes data from input to output.
- Basic filter: A filter based on one attribute.
- Complex filter: A filter based on a combination of attributes
- An aggregator over a window: Count, Sum, Max, Min, Avg, etc.
- A join between a stream and a window.
- A join between multiple windows
- Event pattern detection, etc.

6.3 How Many Streams and Queries Can the Engine Handle?

While the preceding questions deal with one query and one or two streams, often it is much more interesting to know how well the Engine can handle multiple streams and queries. If this is important, ask the vendor about algorithms and techniques used to optimize CPU optimization, memory usage, context switching, etc.

6.4 How is Performance Affected by Persistence, High Availability, Guaranteed Deliver, Security, and other Options?

Vendor literature often discusses performance for a basic engine configuration, without specifying how performance is affected by the above options. Yet these options, especially persistence, introduce overhead that may affect performance significantly. If these options are important to you, make sure to ask the vendor about the impact of these features on performance.

7 Conclusion: Start a Hands-On Evaluation!

As we have seen, a comprehensive evaluation of an Event Stream Processing Engine is a complex matter. There are many questions to ask, and there are many possible answers to any given question. Do not be surprised if the answer from a vendor is usually “it depends”. The answers to many questions, especially questions about performance, depend on a number of assumptions you make.

Therefore, we would like to conclude with reiterating our recommendation to *try* ESP Engines, rather than just relying on the answers from the vendors. Performing *a hands-on evaluation* is by far the best way to figure out if the Engine satisfies all of the requirements, is easy to use, and exhibits acceptable levels of performance. You can do the evaluation completely by yourself, or you can ask the vendor to help you install the product, create some sample use cases for you, and get you started using the ESP Engine on your own. Whatever option you choose, a hands-on evaluation will quickly help you determine which, if any, ESP Engine is the right choice for your project.