Oracle CEP 11gR1 11.1.1.3 (PS2) has just being released and it has several noteworthy improvements and features:

Integration of CQL with Java

CQL (or any other event processing language) allows the authoring of event processing applications at a higher level of abstraction, making them less suitable for dealing with low-level tasks, such as String manipulation, and other programming-in-the-small problems; and lack the richness of other programming language libraries (e.g. Java), which have been built over several years of usage.

In this new release of Oracle CEP, we solve this problem by fully integrating the Java programming language into CQL. This is done at the type-system level, rather than through User-Defined Functions or call-outs, allowing the usage of Java classes (e.g. constructors, methods, fields) directly in CQL in a blended form.

```
SELECT "BusArrival", bus.busId,
       Date().toString()
FROM BusPosStream[NOW] as bus,
     BusStop as busStop
WHERE CONTAINS(spatial(busStop.geom, bus.geom, 0.0d) = true
```

In this example, we make use of the Java class `Date`, by invoking its constructor, and then we invoke the instance method `toString()` on the new object.

The JDK has several useful utility classes, such as `Date`, `RegExp`, and `String`, making it a perfect choice for CQL.

Integration of CQL and Spatial

Location tracking and CEP go hand-in-hand. One example of a spatial-related CEP application is automobile traffic monitoring, where the automobile location is received as a continuous event stream.

Oracle CEP now supports the direct usage of spatial types (e.g. Geometry) and spatial functions in CQL, as shown by the next example, which verifies if "the current location of a bus is contained within a pre-determined arrival location".
One very important aspect of this integration is that indexing of the spatial types (e.g. Geometry) is also being handled in the appropriate form. In other words, not only a user is able to leverage the spatial package, but also OCEP takes care of using the right indexing mechanism for the spatial data, such as an R-tree instead of a hash-based index.

**High-Availability Adapters**

CEP applications are characterized by their quick response-time. This is also applicable for high-available CEP applications, hence a common approach for achieving high availability in CEP systems is to use an active/active architecture. In the previous release of OCEP, several APIs were made available for OCEP developers to create their active/active HA CEP solutions.

In this new release, we take a step further and provide several built-in adapters to aide in the creation of HA OCEP applications. Amongst these, there are HA adapters that synchronize time in the upstream portions of the EPN, and synchronize the events in the downstream portions of the EPN, as illustrated in the previous figure.
Other new features include...

There is much more, please check the documentation for the full set of features and their details, but here are other examples:

• Visualizer's Event Injection and Trace, which allows a user to easily and dynamically send and receive events into and from a running application without having to write any code

• Manage the revision history of a OCEP configuration

• Deploy OCEP application libraries, facilitating re-use of adapters, JDBC drivers, and event-types

• Support for a WITHIN clause in CQL's pattern matching to limit the amount of time to wait for a pattern to be detected

• Create aliases in CQL, facilitating the creation and management of complex queries

• Support for TABLE functions in CQL, thus allowing a query to invoke a function that returns a full set of rows (e.g. table)

For detailed documentation on Oracle Complex Event Processing:
http://download.oracle.com/docs/cd/E14571_01/soa.htm#ocep